LAJC Wage Theft Data

September 18, 2019

1 About

- Author: Raf Alvarado, Data Science Institute, UVA
- Date: 2019-09-17

2 Prepare the data

```
In [2]: import pandas as pd
        import sqlite3
        import re
        import plotly_express as px

In [3]: %matplotlib inline
```

2.1 Import CSV

```
In [4]: df = pd.read_csv('../Raw data/lajc_wage_claim.csv')

In [5]: N = df.shape[0]

In [6]: N

Out[6]: 3948
```

2.2 Import codebook

We do this because we may want to use these details in visualizations and presentations later, e.g. when column names are used as labels, when can use descriptions as hover text.

```
In [6]: codebook_raw = """CLAIM NO         A unique ID for each official wage-theft complaint
        COMPLAINT         Whether or not this row represents an official complaint         X i
        ROUTINE                          Missing everywhere, safe to ignore
        EMPLOYER NAME         The name of the employer against whom the wage-theft complaint is
        EMPLOYER CITY         The employer's city         Character
        ST         The employer's state         Character
        ZIP         The employer's ZIP code         Character
        gender         The predicted gender of the person issuing the complaint based on first
```

hispanic	Whether the complainant is predicted to be Hispanic based on the last	
asian	Whether the complainant is predicted to be Asian based on the last name	
CLAIM AMT	The amount of money claimed to have been illegally withheld	
CASE OPEN/RE-OPEN	The date the case was opened	Character
CLAIM RECEIVED	The date the claim was received by DOLI	Character
Valid	Whether the claim was deemed by DOLI to be valid	Character: X if
Informal Resolution	Whether the claim was resulted in an informal resolution be	
Bankrupt	Whether the claim is being made against a bankrupt employer	Cl
Invalid	Whether DOLI disqualified the claim for one of the following reasons:	
Fringe Benefits	Whether DOLI disqualified the claim because it deals with conti	
Independent Agent	Whether DOLI disqualified the claim because it involves a cor	
Subcontractor	Whether DOLI disqualified the claim because it involves a subcont	
False Claim	Whether DOLI determined that the claim is false	Character:
Other	Whether DOLI determined that the claim is invalid for another reason	
CLAIM INVAL OTHER DESCRIPTION	The stated reason DOLI found a claim invalid if '	
Claim Validity	Whether an undetermined claim is valid	Character: X if t
Employer left State	Whether a claim is undetermined because the employer left t	
Employer Cannot be Located	Whether a claim is undetermined because the employer	
Complainant Cannot be Located	Whether a claim is undetermined because the compl	
Complainant Dropped Claim	Whether a claim is undetermined because the complaina	
Paid Prior to Investigation	Whether a claim is undetermined because the employe	
Business is Closed	Whether a claim is undetermined because the employer's busi	
Other_1	Whether a claim is undetermined for another reason	Character: X
CLAIM UNDETERMINED OTHER DESCRIPTION	The stated reason for being undetermined i	
VERIFIED CLAIM AMT	The total amount claimed after verification by DOLI	
CASE CLOSE/RECLOSE DATE	Date case was officially closed	Character
Employer Contested Valid Determination	Whether the employer contested a claim t	
1st Response Investigation	Whether only a 1st response was conducted	Cl
Formal Investigation	Whether a formal investigation was conducted	Chara
Request Settlement Conference	All missing	No such requests in t
Request Informal Fact Finding	All missing	No such requests in t
Request Formal Fact Finding	All missing	No such requests in the
Wage Order	Whether a legal order is given for the employer to pay lost wages	
Informal Conference	All missing	No such actions in the data
Civil Action for Wages/Penalties	All missing	No such actions in
Other_2	All missing	No such actions in the data
OTHER DISPOSITION DESCRIPTION	All missing	
CASE CLOSED for the REPRESENTATIVE	Date the DOLI labor law representative ended	
JUDGMENT	All missing	
DISMISSED	All missing	
NON-SUITED	All missing	
JUDGMENT_1	All missing	
DISMISSED_1	All missing	
NON-SUITED_1	All missing	
CIVCOURTDTE FOR WAGES/PENALTY	All missing	
TOT AMT	Total amount recovered for the complainant	Character (but easil
WAGE AMT	Amount recovered in lost wages for the complainant	Character
INTEREST AMT	Amount recovered in interest on lost wages for the complainant	

```
ATTY FEES         Amount recovered in reimbursed attorney fees for the complainant
DATE FOR WAGES         Date wages were returned to the complainant         Character
WAGES APPEALED                  All missing
WAGES APPEAL CIRCUIT COURT DATE                 All missing
ASSESSED         Whether the DOLI investigation concludes the employer owes a civil mo
AMT         Amount of the CMP owed by the employer         Character (but easily conve
ATTY FEES_1         Attorney fees owed by the employer as part of the CMP         Chara
ASSESSED DATE         Date CMP totals were assessed         Character         See above
TOTAL CMP AMT         CMP + attorney fees         Character (but easily converted to nu
CMP APPEALED                  All missing         See above
CMP APPEAL CIRCUIT COURT DATE                 All missing         See above
FINES & COURT COSTS         Other fines and court costs associated with the CMP
TOTAL WAGES         Total wages that the DOLI says have been actually collected
TOTAL INTEREST AMT         Total interest collected         Character (but easily conve
TOT WAGES & INTEREST         Wages + interest collected         Character (but easily
GARNISHED AMT                  All $0
JUDGMENT WAGE AMT         We think these are additional fees for repeat offenders
JUDGEMENT PENALTY AMT         We think these are additional fees for repeat offenders
DOCKETED/SENT FOR COLLECTION DATE                  All missing
ACTION TAKEN                  All missing
COURT DATE                 All missing
CONVICTED - DISPOSITION                 All missing
DISMISSED - DISPOSITION                 All missing
NOL Prossed                  All missing
CONFINEMENT                  All missing
SUSPENSION                 All missing
BOTH - C&F                 All missing
JUDGMENT_2                 All missing
DISMISSED_2                  All missing
NON-SUITED_2                  All missing
JUDGMENT_3                 All missing
DISMISSED_3                  All missing
NON-SUITED_3                  All missing         """.split('\n')
```

```
In [7]: codebook = pd.DataFrame([row.split('\t') for row in codebook_raw], columns=['Variable'
```

```
In [8]: codebook.head()
```

Out[8]:

	Variable	Description	Values	Notes
0	CLAIM NO	A unique ID for each official wage-theft compl...		
1	COMPLAINT	Whether or not this row represents an official...	X if yes	All rows are complaints, so safe to ignore thi...
2	ROUTINE			
3	EMPLOYER NAME	The name of the employer against whom the wage...		
4	EMPLOYER CITY	The employer's city		

```
2                              Missing everywhere, safe to ignore
3  Character
4  Character
```

2.3 Normalize column names

It's always a good idea to remove spaces and special characters from column names, especially if we plan to import the CSV file into a database.

```
In [9]: cols_orig =  df.columns
        cols =  [re.sub(r'[/\s-]+', '_', col.lower()) for col in cols_orig]
        df.columns = cols
        codebook['newcol'] = cols

In [10]: codebook = codebook.set_index('newcol')
```

2.4 Convert date columns

```
In [11]: date_cols = codebook[codebook.Description.str.contains('date', case=False)].index.toli

In [12]: date_cols

Out[12]: ['case_open_re_open',
          'claim_received',
          'case_close_reclose_date',
          'case_closed_for_the_representative',
          'date_for_wages',
          'assessed_date']

In [13]: for old_col in date_cols:
             new_col = old_col + "_DATE"
             df[new_col] = df[old_col].str.split('-')
             nas = df[new_col].isna()
             df.loc[~nas, new_col] = df.loc[~nas, new_col].apply(lambda x: "20{}-{}-{}".format
             df[new_col] = pd.to_datetime(df[new_col])

In [14]: df[[col+'_DATE' for col in date_cols]].head()

Out[14]:   case_open_re_open_DATE claim_received_DATE case_close_reclose_date_DATE  \
         0             2014-08-22          2014-08-19                   2018-03-21
         1             2014-09-08          2014-09-04                   2015-01-22
         2             2015-03-06          2015-03-05                   2017-03-17
         3             2015-07-31          2015-07-25                   2016-05-23
         4             2015-08-04          2015-07-28                   2016-01-26

           case_closed_for_the_representative_DATE date_for_wages_DATE  \
         0                             2014-10-20                 NaT
         1                             2014-10-17          2015-01-28
         2                             2015-04-14                 NaT
```

```
3                              2015-10-08                 NaT
4                              2015-10-28                 NaT

  assessed_date_DATE
0                NaT
1         2015-01-28
2                NaT
3                NaT
4                NaT
```

2.5 Convert money columns

```
In [15]: money_cols = codebook[codebook.Description.str.contains('amount', case=False)].index.t

In [16]: money_cols

Out[16]: ['claim_amt',
          'verified_claim_amt',
          'tot_amt',
          'wage_amt',
          'interest_amt',
          'atty_fees',
          'amt']

In [17]: for old_col in money_cols:
             new_col = old_col + "_MONEY"
             df[new_col] = df[old_col].str.replace('[$.,]', '')
             nas = df[new_col].isna()
             df[new_col] = df[new_col].astype('int')

In [18]: new_money_cols = [col+'_MONEY' for col in money_cols]
         df[new_money_cols].head()

Out[18]:    claim_amt_MONEY  verified_claim_amt_MONEY  tot_amt_MONEY  wage_amt_MONEY  \
         0            34000                     34000          45333           34000
         1           127550                    127299         164467          123350
         2                0                     54788              0               0
         3             5000                      5400              0               0
         4           222950                    128613              0               0

            interest_amt_MONEY  atty_fees_MONEY  amt_MONEY
         0                   0            11333      25000
         1                   0            41117      70000
         2                   0                0          0
         3                   0                0          0
         4                   0                0          0
```

2.6 Save data to database

```
In [19]: with sqlite3.connect('lajc.db') as db:
             df.to_sql('wage_claim', db, if_exists='replace')
             codebook.to_sql('codebook', db, if_exists='replace')
```

3 Look at things

3.1 Amounts of money

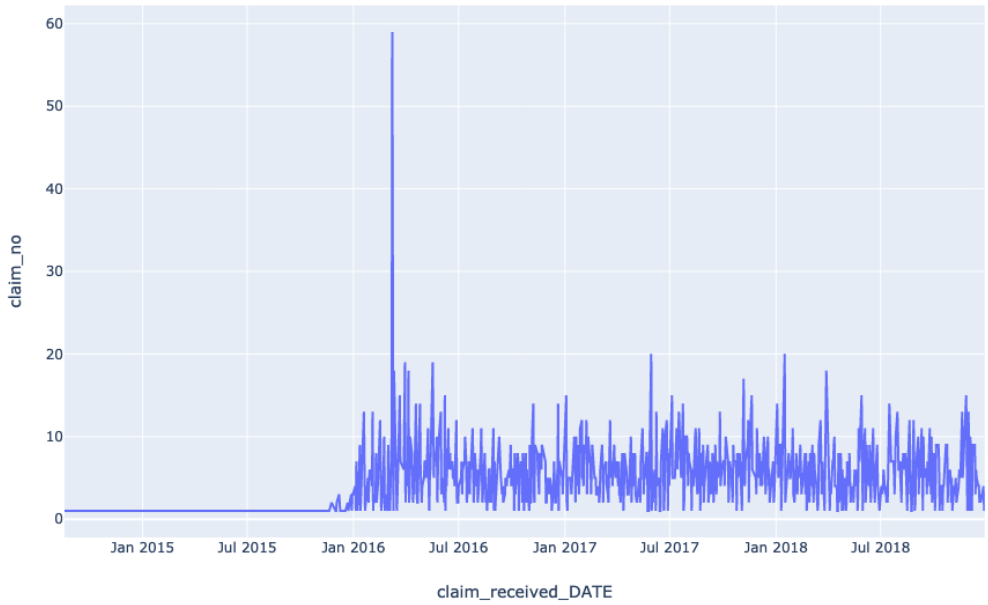
```
In [20]: df[new_money_cols].sum().to_frame().reset_index()\
             .rename(columns={'index':'item', 0:'amt_total'}).sort_values('amt_total', asce
```

Out[20]:

	item	amt_total
0	claim_amt_MONEY	1108830110
1	verified_claim_amt_MONEY	64983670
2	tot_amt_MONEY	1208029
3	wage_amt_MONEY	906022
6	amt_MONEY	550000
5	atty_fees_MONEY	302007
4	interest_amt_MONEY	0

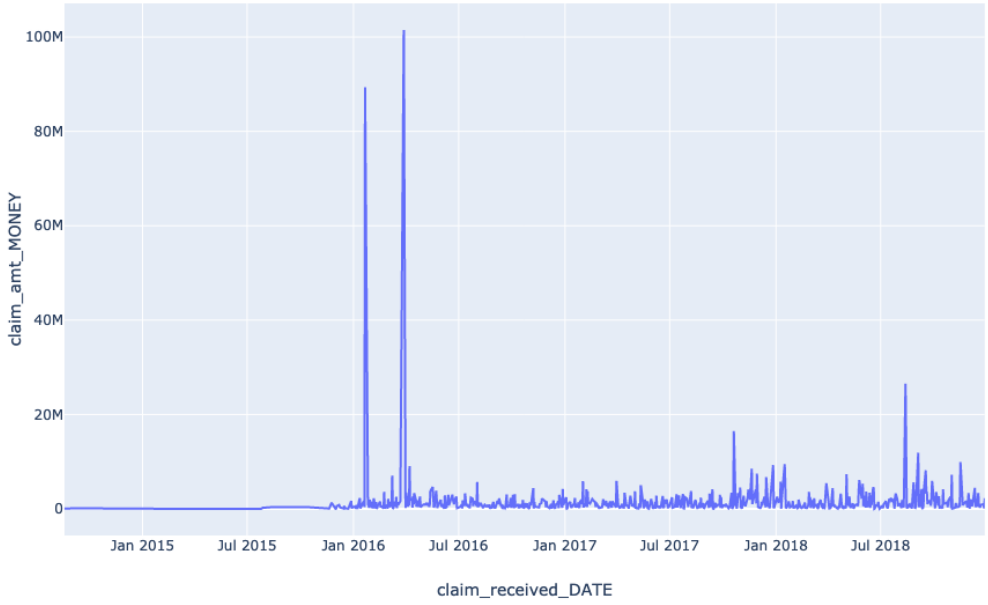
3.2 Number of claims over time

```
In [21]: px.line(df.groupby('claim_received_DATE')['claim_no'].count()\
             .to_frame().reset_index(),
             x='claim_received_DATE', y='claim_no')
```



3.3 Amount of claims over time (aggregate)

```
In [22]: px.line(df.groupby('claim_received_DATE')['claim_amt_MONEY'].sum().reset_index(),
                 x='claim_received_DATE',  y='claim_amt_MONEY')
```



4 Questions

4.1 Question 1: Determine what % of claims get wages ordered to be returned

```
In [23]: codebook.loc['wage_order', 'Description']
```

```
Out[23]: 'Whether a legal order is given for the employer to pay lost wages '
```

```
In [24]: q1 =  round((df.wage_order.value_counts().values[0] / N) * 100, 2)
```

```
In [25]: q1
```

```
Out[25]: 0.23
```

4.2 Question 2: Determine what % of claims get 1st response and formal investigations opened

```
In [26]: q2v1 = '1st_response_investigation'
         q2v2 = 'formal_investigation'

In [27]: # df[q2v1].value_counts()[0]

In [28]: # df[q2v2].value_counts()[0]

In [29]: q2 = df[(df[q2v1] == 'X')  & (df[q2v2]  == 'X')].values

In [30]: len(q2)

Out[30]: 0
```

4.3 Question 3: Determine what % of claim are found to be valid, invalid, and undetermined

```
In [31]: q3v1 = 'valid'
         q3v2 = 'invalid'
         q3_a = df[df[q3v1] == 'X'].shape[0]
         q3_b =  df[df[q3v2] == 'X'].shape[0]
         q3_c = N - (q3_a + q3_b)

In [32]: q3_a, q3_b, q3_c

Out[32]: (173, 1038, 2737)
```

4.4 Question 4: How long does it take to conduct an investigation?

```
In [33]: start_date_col = 'case_open_re_open'
         end_date_col = 'case_close_reclose_date'
         df['how_long_investigation'] = df[end_date_col+'_DATE'] - df[start_date_col+'_DATE']

In [34]: df.how_long_investigation.describe()

Out[34]: count                        3948
         mean      16 days 19:35:11.854103
         std       39 days 03:46:07.942834
         min                0 days 00:00:00
         25%                0 days 00:00:00
         50%                1 days 00:00:00
         75%               22 days 00:00:00
         max             1307 days 00:00:00
         Name: how_long_investigation, dtype: object
```

4.5 Question 5: How long does it take to get a wage order?

```
In [35]: start_date_col = 'claim_received'
         end_date_col = 'date_for_wages'
         df['how_long_get_wage_order'] = df[end_date_col+'_DATE'] - df[start_date_col+'_DATE']

In [36]: df.how_long_get_wage_order.describe()

Out[36]: count                          4
         mean           118 days 06:00:00
         std      35 days 10:52:20.976187
         min             71 days 00:00:00
         25%            101 days 00:00:00
         50%            128 days 00:00:00
         75%            145 days 06:00:00
         max            146 days 00:00:00
         Name: how_long_get_wage_order, dtype: object
```

4.6 Question 6: If wages are ordered to be returned, what's the distribution of amounts?

```
In [37]: df['tot_amt_MONEY'].value_counts()

Out[37]: 0         3939
         8000         1
         140921       1
         228800       1
         7200         1
         216000       1
         164467       1
         318000       1
         45333        1
         79308        1
         Name: tot_amt_MONEY, dtype: int64
```

4.7 Question 7: How do these outcomes listed above depend on ethnicity, gender, and industrial category?

4.7.1 Ethnicity

```
In [38]: round((df['hispanic'].sum() / N)  * 100, 2)

Out[38]: 8.56

In [39]: round((df['asian'].sum() / N)  * 100, 2)

Out[39]: 2.89
```

4.7.2 Gender

```
In [40]: round((df['gender'].value_counts() / N) * 100, 2)
```

```
Out[40]: male      49.72
         female    37.99
         Name: gender, dtype: float64
```

4.7.3 Industry

4.7.4 20 Most offending cities

```
In [41]: df.employer_city.value_counts().to_frame().head(20)
```

```
Out[41]:                   employer_city
         RICHMOND                    369
         VIRGINIA BEACH              274
         CHESAPEAKE                  161
         WOODBRIDGE                  129
         NORFOLK                     124
         NEWPORT NEWS                108
         FREDERICKSBURG              101
         MANASSAS                     95
         ALEXANDRIA                   82
         HAMPTON                      78
         STERLING                     73
         ROANOKE                      70
         CHANTILLY                    67
         FAIRFAX                      58
         ARLINGTON                    55
         PORTSMOUTH                   54
         HENRICO                      51
         VIENNA                       48
         FALLS CHURCH                 41
         CHARLOTTESVILLE              41
```

4.7.5 20 Most offending zip codes

```
In [42]: df.groupby(['employer_city', 'zip']).claim_no.count()\
             .to_frame().sort_values('claim_no', ascending=False).head(20)
```

```
Out[42]:                          claim_no
         employer_city  zip
         WOODBRIDGE     22192.0        94
         CHESAPEAKE     23320.0        77
         VIRGINIA BEACH 23462.0        62
         CHANTILLY      20151.0        56
         STERLING       20166.0        53
         VIRGINIA BEACH 23452.0        50
         MANASSAS       20109.0        43
```

```
RICHMOND       23230.0        39
VIRGINIA BEACH 23454.0        38
HAMPTON        23666.0        38
RICHMOND       23235.0        37
               23224.0        36
MANASSAS       20110.0        33
NEWPORT NEWS   23606.0        32
VIRGINIA BEACH 23455.0        32
VIENNA         22182.0        32
NORFOLK        23510.0        30
RICHMOND       23220.0        29
MIDLOTHIAN     23112.0        29
VIRGINIA BEACH 23451.0        28
```

4.7.6 20 Most offending employers

```
In [43]: df.groupby(['employer_name','employer_city', 'zip']).claim_no.count()\
             .to_frame().sort_values('claim_no', ascending=False).head(20)
```

Out[43]:

employer_name	employer_city	zip	claim_no
FIRST TRANSIT	WOODBRIDGE	22192.0	71
CHESAPEAKE SERVICE SYSTEMS	CHESAPEAKE	23320.0	11
ANCHOR BAR	RICHMOND	23235.0	11
A.C.I. DRYWALL CIA	SHADY SIDE	20764.0	7
CHESAPEAKE SERVICE SYSTEMS, INC.	CHESAPEAKE	23320.0	7
BRAVEN PAINTING LLC	VIRGINIA BEACH	23453.0	6
PUBLIC PARTNERSHIP LLC	GLEN ALLEN	23060.0	5
THOMAS SWANSTON	VIRGINIA BEACH	23454.0	5
MINISTERING ANGELS LLC	WINCHESTER	22601.0	5
CARLISLE LIVING LLC	VIRGINIA BEACH	23464.0	5
ROYAL CLEANING SERVICES	NORTH CHESTERFIELD	23236.0	5
EPC BUILDERS LLC	ROCKVILLE	23146.0	5
TRUSTIFY	ARLINGTON	22202.0	4
BODY & SOL TANNING LLC	YORKTOWN	23693.0	4
GREATER WASHINGTON ENDODONTICS	FAIRFAX	22031.0	4
MARATHON RESOURCE MANAGEMENT GROUP	ASHLAND	23005.0	4
SMYTH COUNTY AMBULANCE SERVICE	MARION	24354.0	4
THE TREAT SHOP	RICHMOND	23225.0	4
UNITED SERVICE 333 LLC	HERNDON	20171.0	4
RAGSDALE COMMERCIAL CLEANING	STANARDSVILLE	22973.0	4