Predicting the Directional Change in the Number of Social Security Filers

Henry Johnson

April 2021

Table of Contents

1 Abstract

This study forecasts the directional change in social security filers by utilizing financial data from Yahoo Finance and the Federal Reserve Bank of St. Louis. The goal of this research is to as-accurately-as-possible predict the change in social security solely by financial data. No attention is paid to omitted variable bias; but I still grapple with questions of how changes in financial data will affect the overall number of social security filers.

2 Introduction

Many Americans rely on Social Security benefits for their retirement. Due to this, it's important for policymakers to have realistic expectations of the total amount of social security filers in order to better-allocate resources for the future.

Through using readily-available financial data on the Dow Jones Index, S&P 500, federal funds rate, the labor force participation rate of individuals at least 55 years old, and social security benefits, I estimate the number of social security filers in the United States.

3 Literature Review

Since the change from defined benefit (pensions provided by a company for a worker's retirement for life) to defined contribution (retirement accounts contributed to by either the company or individuals or both) plans, people near retirement age have been more exposed to stock market fluctuations (Goda, Shoven, Slavov, 2012). The prediction that changes in wealth will result in changes in labor supply and retirement intentions is called the lifecycle model.

In 2006, 15.2% of wealth held by the 53-58 year old age bracket was in IRAs or direct stock holdings (Gustman, Steinmeier, and Tabatabai, 2010). With the percentage of wealth held in the stock market is small, the literature I read found little effect and little statistical significance from stock market fluctuations on retirement intentions. But these findings seem highly dependent on the assumption that, despite the recent trend in moving from defined benefit to defined contribution, retirement plans will not have much stock market exposure.

In Goda, Shoven, and Slavov (2012), they examine labor market intentions using a more restrictive dataset from the Health and Retirement Study (HRS) that provides county-level and age-specific data that allows them to look closely at the relationship between labor force participation and retirement and stock market fluctuations by matching respondents to county-level unemployment rates and house price indices. They look at SP 500 changes in the past month and year and try to weed out whether those have a statistically significant effect on the intentions to retire. Though they found a negative correlation between stock market fluctuations and retirement intentions, they didn't find the coefficients to be consistently statistically significant.

Similarly to the Goda, Shoven, and Slavov (2012) study, Coile and Levine (2004) used HRS data and added Current Population Survey (CPS) and Consumer Finances (SCF) datasets to conduct a double quasi-experiment to investigate whether retirement and labor force patterns are affected by stock market fluctuations. Similarly, they find no evidence that the stock market affects aggregate labor supply and retirement.

The literature I looked at seems to not find a statistically significant relationship between stock market fluctuations and retirement. However, there is consistently an inverse correlation between stock market changes and retirement numbers.

4 Data

I collected monthly data on social security filers from the Social Security Administration, monthly financial data on the Consumer Price Index and Federal Funds Rates from the Federal Reserve Bank of St. Louis, monthly financial data from Yahoo Finance for the S/P 500 and Dow Jones Index, and monthly Bureau of Labor Statistics Data for the labor force participation rate of individuals at least 55 years old.

5 Methods

The original model I decided on was a linear regression model using the total number of social security filers as the dependent variable and regressors for the average amount paid by social security in real terms, a synthetic variable generated from the S&P 500 and the Dow Jones Index (DJI) in real terms, the federal funds rate, and the labor force participation rate for the population over 55 years old.

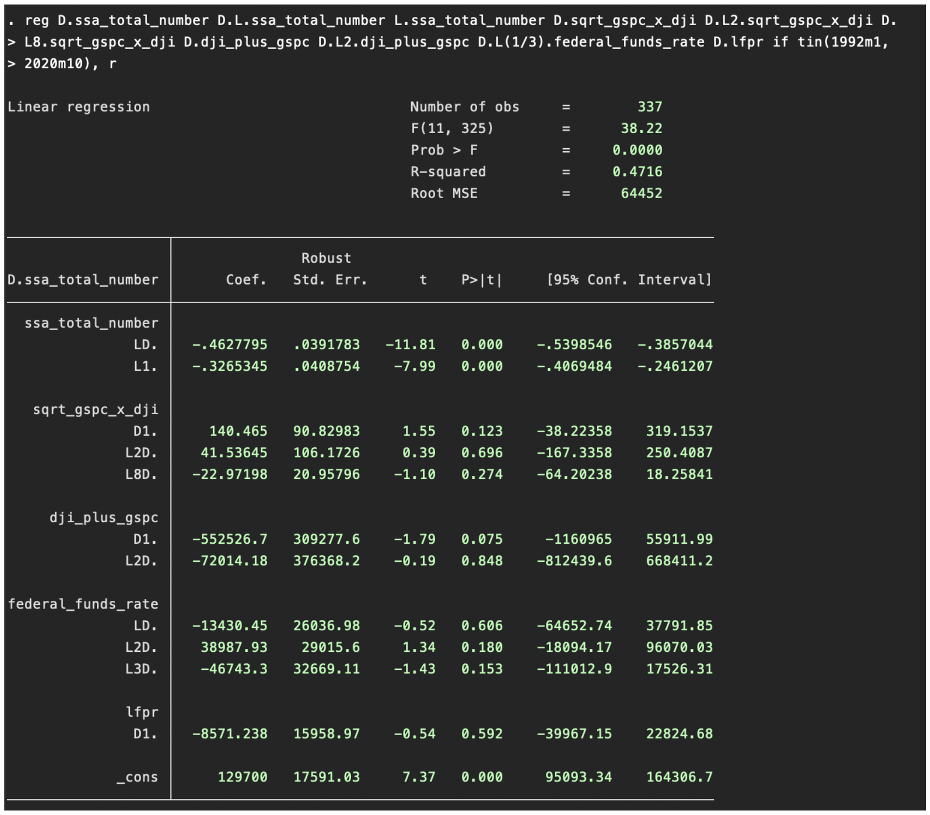
This model provided statistical significance for all the regressors and an R-squared of 64.86%, indicating that about 64% of the variation in the total number of social security filers is explained by the model (Figure 4).

After looking into a simple linear regression model, I decided to use an autoregressive distributed lag model. By plotting the total number of social security filers, I noticed a stochastic trend with a drift (as shown in figures 2 and 3). So I tested for autocorrelation in the dependent variable and confirmed its presence.

In order to assess stationarity, I used perron, dickey-fuller GLS (dfgls), and ADF tests. All of these tests are similar, with just slight variations. The dfgls test is like an augmented dickey fuller test that uses generalized least squares in addition to it. In STATA, this test operates on 16 legs and the lags that are below the critical threshhold we're able to reject the null hypothesis that the variable is a random walk, possibly with a drift. I ran this test against all of the variables to identify stationary variables to use in my model.

After this, I used the Bayes Information Criteria and Akike Information Criteria to determine appropriate lags. Since the addition of each lag resulted in both the BIC and AIC values decreasing, I plotted the values and identified kinks in the plot as appropriate lags.

My final specification is in Figure 1. This is an ADL(1,3) model with an R-squared of 47%.



```
. reg D.ssa_total_number D.L.ssa_total_number L.ssa_total_number D.sqrt_gspc_x_dji D.L2.sqrt_gspc_x_dji D.
> L8.sqrt_gspc_x_dji D.dji_plus_gspc D.L2.dji_plus_gspc D.L(1/3).federal_funds_rate D.lfpr if tin(1992m1,
> 2020m10), r

Linear regression                               Number of obs     =        337
                                                F(11, 325)        =      38.22
                                                Prob > F          =     0.0000
                                                R-squared         =     0.4716
                                                Root MSE          =      64452

------------------------------------------------------------------------------
                 |               Robust
D.ssa_total_number |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-----------------+------------------------------------------------------------
  ssa_total_number |
             LD. |  -.4627795   .0391783   -11.81   0.000    -.5398546   -.3857044
             L1. |  -.3265345   .0408754    -7.99   0.000    -.4069484   -.2461207
                 |
   sqrt_gspc_x_dji |
             D1. |    140.465   90.82983     1.55   0.123    -38.22358    319.1537
            L2D. |   41.53645   106.1726     0.39   0.696    -167.3358    250.4087
            L8D. |  -22.97198   20.95796    -1.10   0.274    -64.20238    18.25841
                 |
     dji_plus_gspc |
             D1. |  -552526.7   309277.6    -1.79   0.075     -1160965    55911.99
            L2D. |  -72014.18   376368.2    -0.19   0.848    -812439.6    668411.2
                 |
federal_funds_rate |
             LD. |  -13430.45   26036.98    -0.52   0.606    -64652.74    37791.85
            L2D. |   38987.93    29015.6     1.34   0.180    -18094.17    96070.03
            L3D. |   -46743.3   32669.11    -1.43   0.153    -111012.9    17526.31
                 |
            lfpr |
             D1. |  -8571.238   15958.97    -0.54   0.592    -39967.15    22824.68
                 |
           _cons |     129700   17591.03     7.37   0.000     95093.34    164306.7
------------------------------------------------------------------------------
```

Figure 1: Total Number of Social Security Filers

6 Analysis

Using python and the numpy and pandas libraries, I retrieved data via APIs and by directly downloading and cleaned the data to ensure that date formats were consistent. After cleaning each of the datasets, I created synthetic variables such as real instead of nominal values, monthly changes in total and rates, and binary variables indicating the direction of the change.

I merged all the data into one dataset, which resulted in 347 observations, and exported the tabular data as an excel file. Then I conducted linear regressions and statistical tests in Stata to validate that the relationships between the data points and the total number of social security were statistically significant and the model appropriately represented the relationships. When I included both the DJI and S&P 500 as independent regressors, the model had multicollinearity issues. This is likely due to how these variables affect one

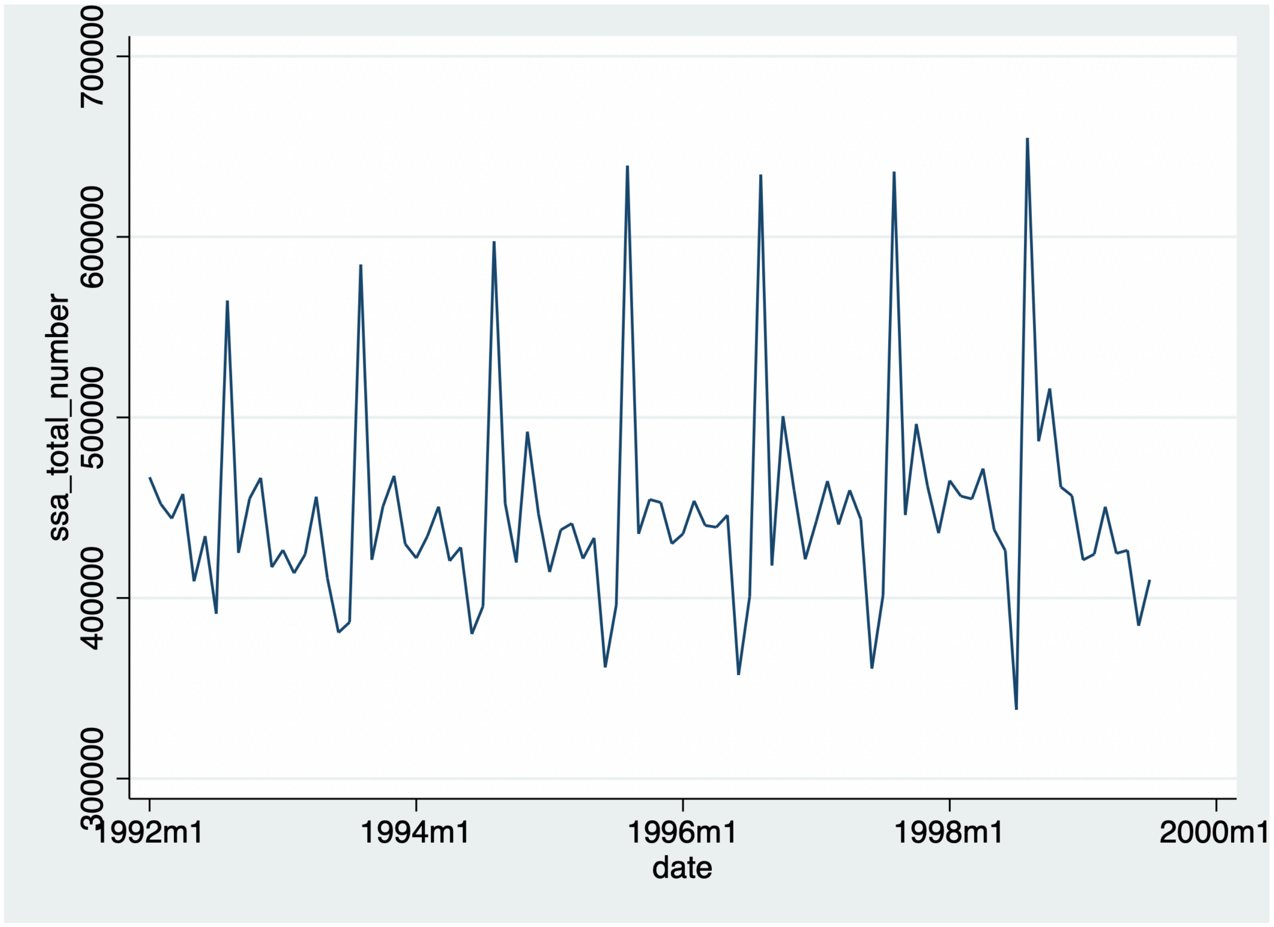


Figure 2: Total Number of Social Security Filers

another. In order to reduce the multicollinearity issue, I transformed the datapoints into new datapoints and used the synthetic variables that capture changes in both the S&P 500 and the DJI in place of the original datapoints.

After including the average amount paid out for social security, the sign on the coefficient for the S&P 500, DJI, and the synthetic variable all became negative. So I normalized the DJI and SP datapoints and summed them together. This resulted in the expected direction while maintaining statistical significance in the OLS model.

The model implies that for a one dollar increase in the real value of the average amount paid out to social security filers, the total number of filers will increase by 685. This value is statistically significant at the 1% level (Figure 4). All of the independent variables in the model are statistically significant at at least the 5% level, indicating that it's unlikely

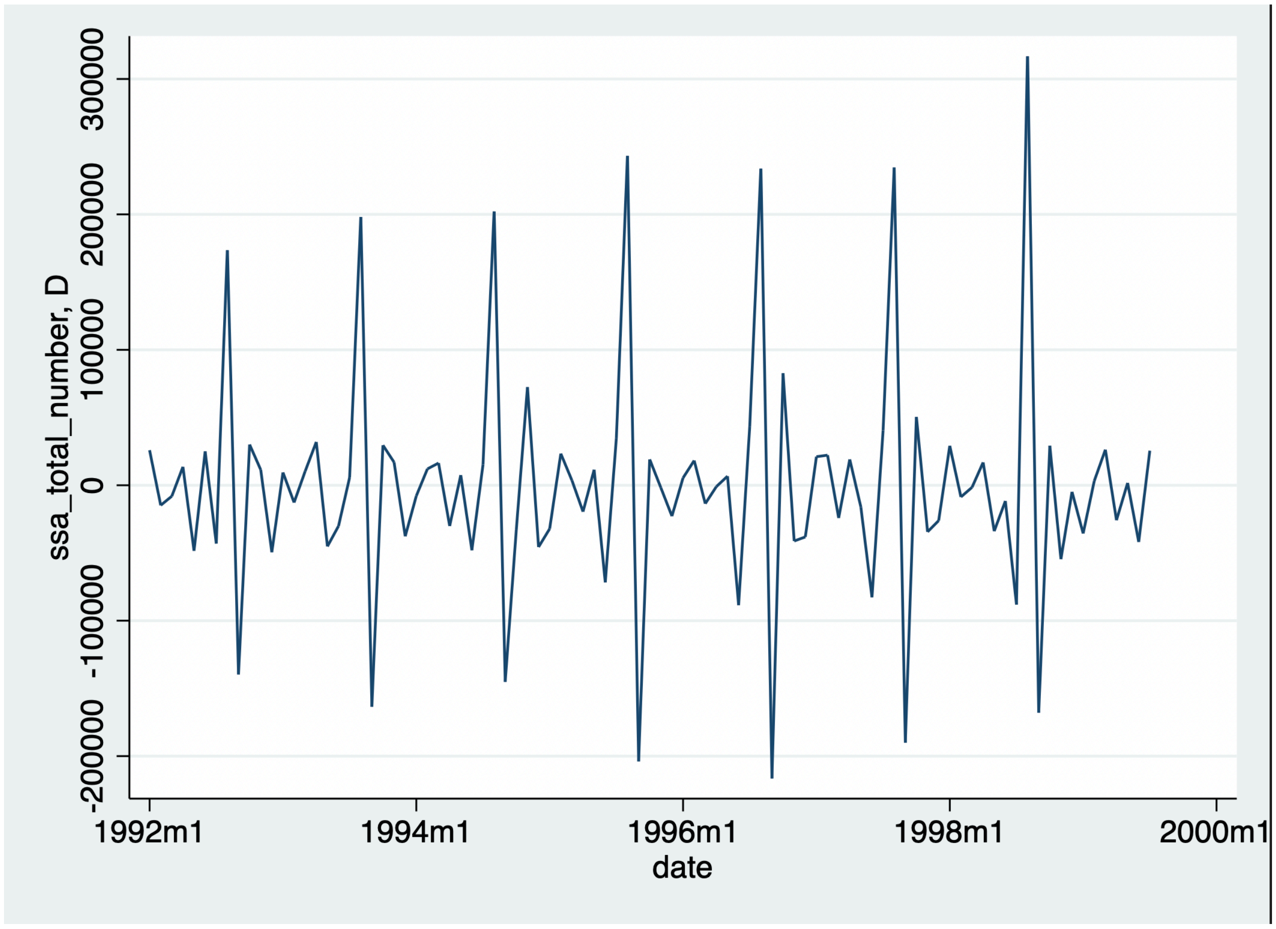
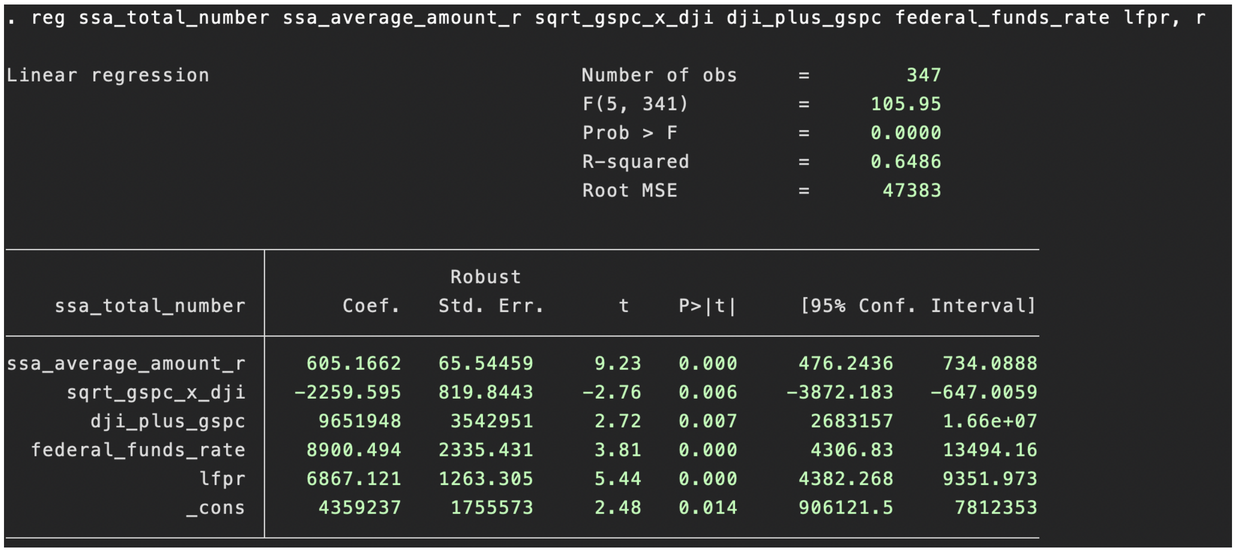


Figure 3: Change In Number of Social Security Filers

that the relationship between the independent variables and the dependent variable (total number of filers) is simply a correlation.

In the ADL model, the R-squared wasn't quite as good as it was in the simple OLS. But I was surprised by how small the forecast error was. Figure 6 shows a plot of the fitted difference values against the actual differences. The adjusted R-squared for the ADL model is 47.16%. As evident by Figure 6, the fit of the prediction to the model is not very good. My forecast error in this model is 18.6%.

Earlier on I was getting what appeared to be much better results. The model specified in Figure 5 had an adjusted R-squared of nearly 64% and a forecast error of around 3%. Unfortunately, this model suffered from stationarity issues in a couple of the independent variables and its results were more a matter of the non-stationary data inflating the R-squared and luck in the data I was forecasting than anything else (Figure 7).

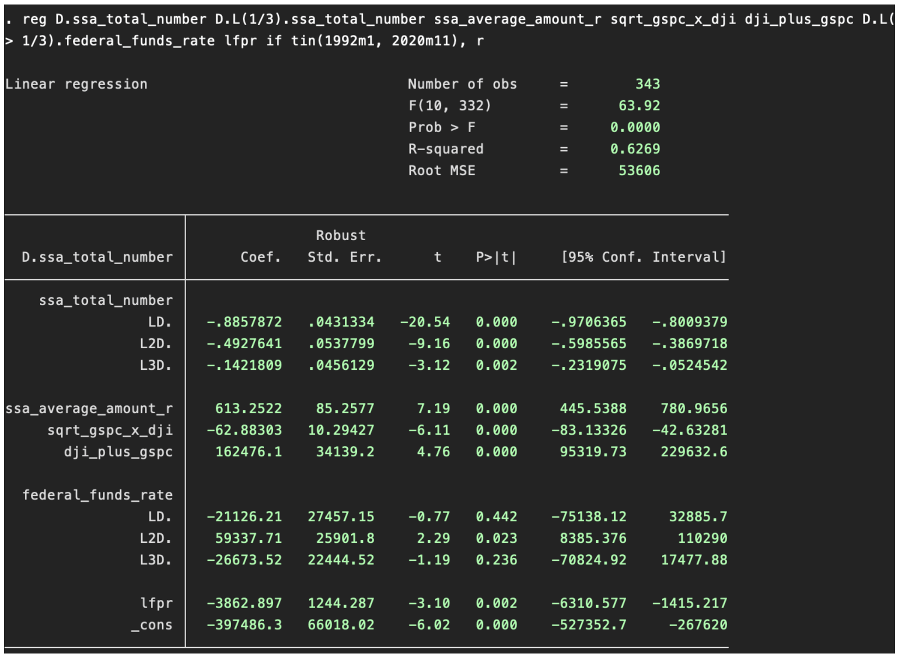


```
. reg ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji dji_plus_gspc federal_funds_rate lfpr, r

Linear regression                               Number of obs     =        347
                                                F(5, 341)         =     105.95
                                                Prob > F          =     0.0000
                                                R-squared         =     0.6486
                                                Root MSE          =      47383

------------------------------------------------------------------------------------
                    |               Robust
   ssa_total_number |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
--------------------+---------------------------------------------------------------
ssa_average_amount_r |   605.1662   65.54459     9.23   0.000     476.2436    734.0888
     sqrt_gspc_x_dji |  -2259.595   819.8443    -2.76   0.006    -3872.183   -647.0059
       dji_plus_gspc |    9651948    3542951     2.72   0.007      2683157    1.66e+07
  federal_funds_rate |   8900.494   2335.431     3.81   0.000      4306.83    13494.16
                lfpr |   6867.121   1263.305     5.44   0.000     4382.268    9351.973
               _cons |    4359237    1755573     2.48   0.014     906121.5     7812353
------------------------------------------------------------------------------------
```

Figure 4: OLS



```
. reg D.ssa_total_number D.L(1/3).ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji dji_plus_gspc D.L(
> 1/3).federal_funds_rate lfpr if tin(1992m1, 2020m11), r

Linear regression                               Number of obs     =        343
                                                F(10, 332)        =      63.92
                                                Prob > F          =     0.0000
                                                R-squared         =     0.6269
                                                Root MSE          =      53606

------------------------------------------------------------------------------------
                    |               Robust
 D.ssa_total_number |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
--------------------+---------------------------------------------------------------
   ssa_total_number |
                LD. |  -.8857872   .0431334   -20.54   0.000    -.9706365   -.8009379
               L2D. |  -.4927641   .0537799    -9.16   0.000    -.5985565   -.3869718
               L3D. |  -.1421809   .0456129    -3.12   0.002    -.2319075   -.0524542
                    |
ssa_average_amount_r |   613.2522    85.2577     7.19   0.000     445.5388    780.9656
    sqrt_gspc_x_dji |  -62.88303   10.29427    -6.11   0.000    -83.13326   -42.63281
      dji_plus_gspc |   162476.1    34139.2     4.76   0.000     95319.73    229632.6
                    |
 federal_funds_rate |
                LD. |  -21126.21   27457.15    -0.77   0.442    -75138.12     32885.7
               L2D. |   59337.71    25901.8     2.29   0.023     8385.376      110290
               L3D. |  -26673.52   22444.52    -1.19   0.236    -70824.92    17477.88
                    |
               lfpr |  -3862.897   1244.287    -3.10   0.002    -6310.577   -1415.217
              _cons |  -397486.3   66018.02    -6.02   0.000    -527352.7     -267620
------------------------------------------------------------------------------------
```

Figure 5: Time Series

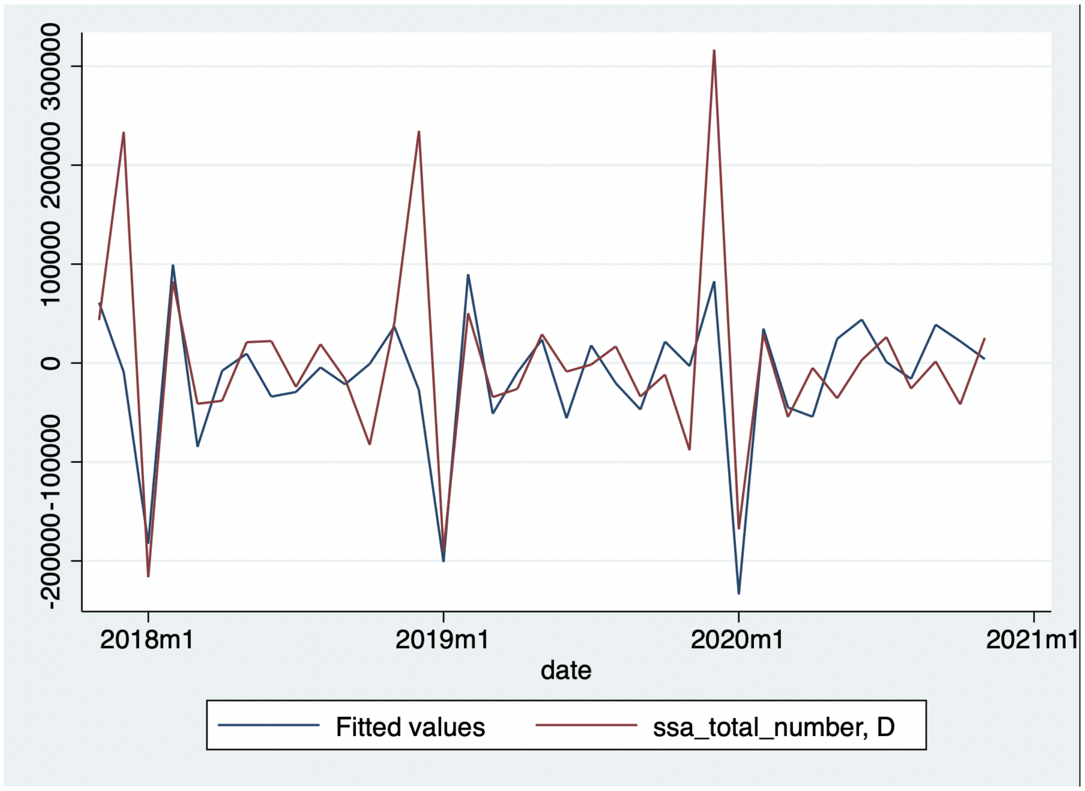


Figure 6: Forecast

7 Conclusion

Policymakers could use linear regressions to evaluate the changes in the number of people who file for social security. Each of the models demonstrated that the SP 500 or Dow Jones Index was statistically significant in estimating the effect of the financial market on the number of social security filers.

This research demonstrates that we can predict the effect of changes in financial markets on social security—though I remain skeptical as to the magnitudes of these effects, I feel confident that with these datapoints, policymakers can estimate the change in the total number of social security filers.

Further analyses can be done using probit and logit models to help validate that these findings aren't specific to the model selections.

[1] [3] [2]

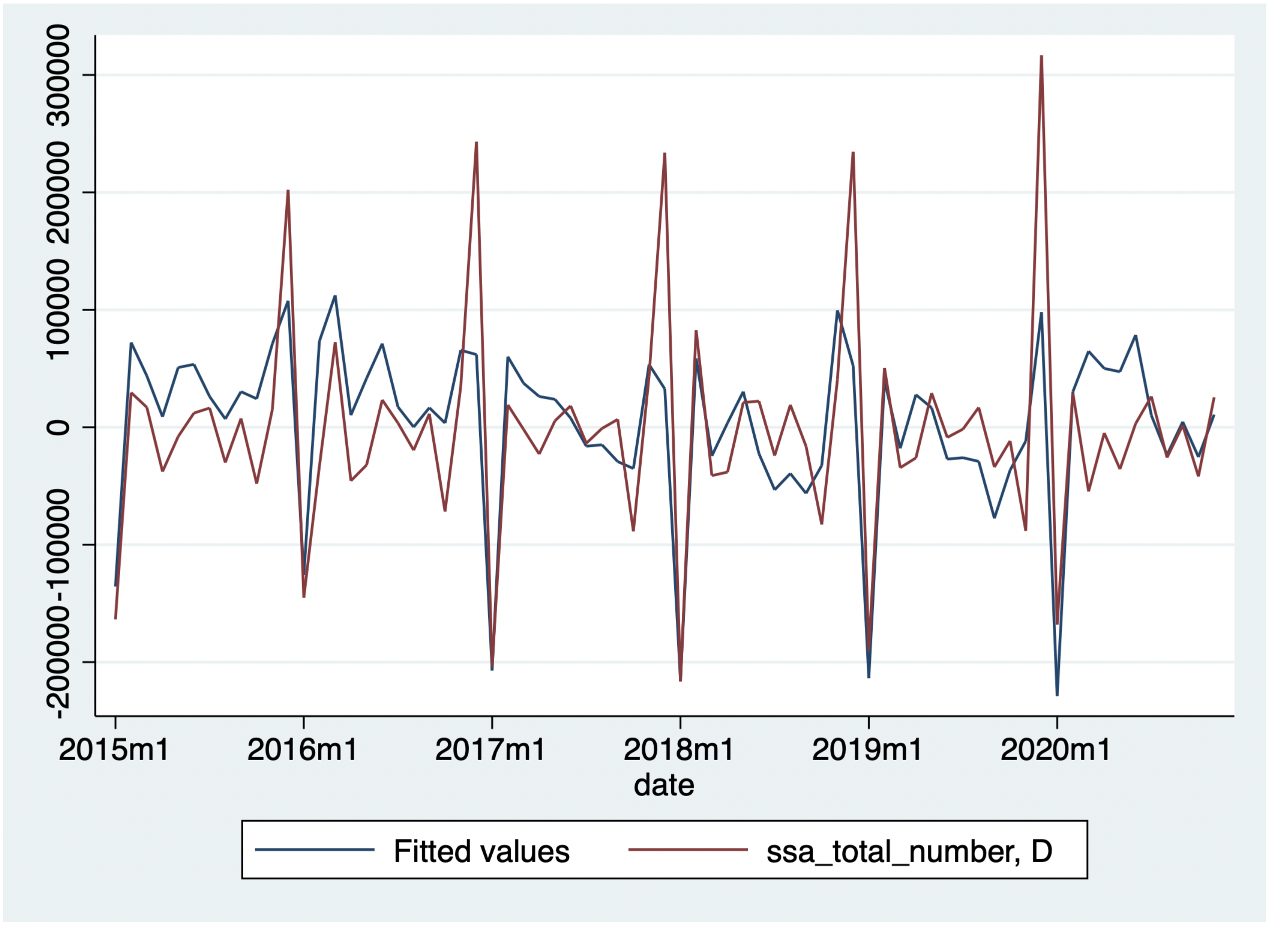


Figure 7: Forecast

References

[1] Thomas L. Steinmeier Alan L. Gustman and Nahid Tabatabai. What the stock market decline means for the financial security and retirement choices of the near-retirement population. *The Journal of Economic Perspectives*, 24(1):161–182, 2010.

[2] John B. Shoven Gopi Shah Goda and Sita Nataraj Slavov. Does stock market performance influence retirement intentions? *The Journal of Human Resources*, 47(4):1055–1081, 2012.

[3] Courtney C. Coile Phillip B. Levine. Bulls, bears, and retirement behavior. *NBER Working Paper Series 10779*, (10779):1–52, 2004.

8 Python Code

Listing 1: Pulling Data

```
import pandas as pd
import yfinance as yf
from fredapi import Fred
from api_key import fred_api_key
from datetime import datetime

# API key for FRED
fred = Fred(api_key=fred_api_key)

# retrieve fed funds rate data
fs = fred.get_series('FEDFUNDS')
fedfunds_df = fs.reset_index()
fedfunds_df.columns = ['month', 'federal_funds_rate']

# retrieve cpi data
cs = fred.get_series('CPIAUCSL')
cpi_df = cs.reset_index()
cpi_df.columns = ['month', 'cpi']

# set the current cpi value for cpi-adjustment
cpi_current = cpi_df['cpi'].iloc[-1]

# retrieve dji data
dji_df = yf.download('^DJI', start='1990-01-01', end='2021-03-01', interval="
dji_df.reset_index(inplace=True)
dji_df.columns = ['month', 'dji_open', 'dji_high', 'dji_low', 'dji_close', 'd

# combine dji and cpi data for inflation adjustment
dji_cpi_df = pd.merge(dji_df, cpi_df, on=['month'], how='inner')
cols = ['dji_open', 'dji_high', 'dji_low', 'dji_close', 'dji_adj_close']

# calculate cpi adjusted monetary values
for col in cols:
    r = col + '_r'
    dji_cpi_df[r] = (dji_cpi_df[col] * cpi_current) / dji_cpi_df['cpi']
# calculate differences in real and nominal values from month to month
cols = ['dji_open', 'dji_high', 'dji_low', 'dji_close', 'dji_adj_close']
for col in cols:
    r = col + '_r'
    real_diff = col + '_diff_r'
    nominal_diff = col + '_diff_n'
    dji_cpi_df[real_diff] = dji_cpi_df[r].diff(1)
    dji_cpi_df[nominal_diff] = dji_cpi_df[col].diff(1)

# rearrange columns for reading purposes
dji_cpi_df = dji_cpi_df.reindex(sorted(dji_cpi_df.columns), axis=1)

# retrieve sp500 data
gspc_df = yf.download('^GSPC', start='1990-01-01', end='2021-03-01', interval
gspc_df.reset_index(inplace=True)
gspc_df.columns = ['month', 'gspc_open', 'gspc_high', 'gspc_low', 'gspc_close
```

```
# combine sp500 and cpi data for inflation adjustment
gspc_cpi_df = pd.merge(gspc_df, cpi_df, on=['month'], how='inner')
cols = ['gspc_open', 'gspc_high', 'gspc_low', 'gspc_close', 'gspc_adj_close']

# calculate cpi adjusted monetary values
for col in cols:
    r = col + '_r'
    gspc_cpi_df[r] = (gspc_cpi_df[col] * cpi_current) / gspc_cpi_df['cpi']
# calculate differences in real and nominal values from month to month
for col in cols:
    r = col + '_r'
    real_diff = col + '_diff_r'
    nominal_diff = col + '_diff_n'
    gspc_cpi_df[real_diff] = gspc_cpi_df[r].diff(1)
    gspc_cpi_df[nominal_diff] = gspc_cpi_df[col].diff(1)

# rearrange columns for reading purposes
gspc_cpi_df = gspc_cpi_df.reindex(sorted(gspc_cpi_df.columns), axis=1)

# merge dji and sp datasets
dji_sp_df = pd.merge(dji_cpi_df, gspc_cpi_df, on=['month', 'cpi'], how='inner

# merge federal funds rate data in to financial data
fin_df = pd.merge(fedfunds_df, dji_sp_df, on=['month'], how='inner')

# pull in social security data
```

```
ssa_benefits_path = 'data/SSA.xlsx'
ssa_df = pd.read_excel(ssa_benefits_path)

# clean social security data
ssa_df['Month'] = pd.to_datetime(ssa_df['Month'], format='%b_%Y', errors='ign
ssa_df['TotalNumber'] = ssa_df['TotalNumber'].replace('[\,]', '', regex=True)
ssa_df['AverageAmount'] = ssa_df['AverageAmount'].replace('[\$,]', '', regex=
ssa_df = ssa_df.drop(labels='Unnamed:_3', axis=1)
ssa_df.columns = ['month', 'ssa_total_number', 'ssa_average_amount']

# calculate change in the total number of social security recipients by month
ssa_df['ssa_total_number_diff'] = ssa_df['ssa_total_number'].diff(1)

# combine social security data and cpi data for inflation adjustment
ssa_cpi_df = pd.merge(ssa_df, cpi_df, on=['month'], how='inner')

# calculate cpi adjusted values
ssa_cpi_df['ssa_average_amount_r'] = (ssa_cpi_df['ssa_average_amount'] * cpi_
# calculate differences in real and nominal values from month to month
ssa_cpi_df['ssa_average_amount_diff_n'] = ssa_cpi_df['ssa_average_amount'].di
ssa_cpi_df['ssa_average_amount_diff_r'] = ssa_cpi_df['ssa_average_amount_r'].

# rearrange columns for reading purposes
ssa_cpi_df = ssa_cpi_df.reindex(sorted(ssa_cpi_df.columns), axis=1)

# combine ssa dataset with financial dataset
ssa_fin_df = pd.merge(ssa_cpi_df, fin_df, on=['month', 'cpi'], how='inner')
```

```
# copy dataframe to a new master
master_df = ssa_fin_df.copy()

# export dataset to excel
master_df.to_excel(r'data/master_data.xlsx', index=False, header=True)

# create a dataframe of the percent changes by month
percent_change_df = master_df.copy()
cols = ['ssa_average_amount', 'ssa_average_amount_diff_n', 'cpi',
        'ssa_average_amount_diff_r', 'ssa_average_amount_r', 'ssa_total_numbe
        'ssa_total_number_diff', 'dji_adj_close', 'federal_funds_rate',
        'dji_adj_close_diff_n', 'dji_adj_close_diff_r', 'dji_adj_close_r',
        'dji_close', 'dji_close_diff_n', 'dji_close_diff_r', 'dji_close_r',
        'dji_high', 'dji_high_diff_n', 'dji_high_diff_r', 'dji_high_r',
        'dji_low', 'dji_low_diff_n', 'dji_low_diff_r', 'dji_low_r', 'dji_open
        'dji_open_diff_n', 'dji_open_diff_r', 'dji_open_r', 'dji_volume',
        'gspc_adj_close', 'gspc_adj_close_diff_n', 'gspc_adj_close_diff_r',
        'gspc_adj_close_r', 'gspc_close', 'gspc_close_diff_n',
        'gspc_close_diff_r', 'gspc_close_r', 'gspc_high', 'gspc_high_diff_n',
        'gspc_high_diff_r', 'gspc_high_r', 'gspc_low', 'gspc_low_diff_n',
        'gspc_low_diff_r', 'gspc_low_r', 'gspc_open', 'gspc_open_diff_n',
        'gspc_open_diff_r', 'gspc_open_r', 'gspc_volume']
for col in cols:
    percent_change_df[col] = percent_change_df[[col]].pct_change()[col]
# export dataframe to excel
percent_change_df.to_excel(r'data/percent_change_data.xlsx', index=False, hea
# create dataframe of binary values to indicate whether a value increased
is_increased_df = percent_change_df.copy()
cols = ['ssa_average_amount', 'ssa_average_amount_diff_n', 'cpi',
        'ssa_average_amount_diff_r', 'ssa_average_amount_r', 'ssa_total_numbe
        'ssa_total_number_diff', 'dji_adj_close', 'federal_funds_rate',
        'dji_adj_close_diff_n', 'dji_adj_close_diff_r', 'dji_adj_close_r',
        'dji_close', 'dji_close_diff_n', 'dji_close_diff_r', 'dji_close_r',
        'dji_high', 'dji_high_diff_n', 'dji_high_diff_r', 'dji_high_r',
        'dji_low', 'dji_low_diff_n', 'dji_low_diff_r', 'dji_low_r', 'dji_open
        'dji_open_diff_n', 'dji_open_diff_r', 'dji_open_r', 'dji_volume',
        'gspc_adj_close', 'gspc_adj_close_diff_n', 'gspc_adj_close_diff_r',
        'gspc_adj_close_r', 'gspc_close', 'gspc_close_diff_n',
        'gspc_close_diff_r', 'gspc_close_r', 'gspc_high', 'gspc_high_diff_n',
        'gspc_high_diff_r', 'gspc_high_r', 'gspc_low', 'gspc_low_diff_n',
        'gspc_low_diff_r', 'gspc_low_r', 'gspc_open', 'gspc_open_diff_n',
        'gspc_open_diff_r', 'gspc_open_r', 'gspc_volume']
for col in cols:
    is_increased_df[col] = is_increased_df[col].apply(lambda x: 1 if x > 0 el
# export dataframe to excel
is_increased_df.to_excel(r'data/is_increased_data.xlsx', index=False, header=

# update column names for combining with master
percent_change_df.columns = percent_change_df.columns.map(lambda x: x + '_dpe
is_increased_df.columns = is_increased_df.columns.map(lambda x: x + '_increas
print(is_increased_df.columns)
# combine master with additional datapoints
```

```
master_df.reset_index(inplace=True, drop=True)
percent_change_df.reset_index(inplace=True, drop=True)

# create a copy of master to merge dataframes into
master2_df = master_df.copy()
master2_df = master2_df.merge(percent_change_df, on=['month'], how='inner').m
# export new master to excel
master2_df.to_excel(r'data/master2_data.xlsx', index=False, header=True)

# check balance of dataset
master_df.isnull().values.any()
cols = list(master_df.columns)
cols_with_nulls = []
print('columns␣with␣nulls:␣')
for col in cols:
    if master_df[col].isnull().sum() > 0:
        cols_with_nulls.append(col + ':␣' + str(master_df[col].isnull().sum()
# names of columns containing nulls and count of nulls
print(cols_with_nulls)
print(master_df.head())
print(is_increased_df.head())
print(percent_change_df.head())

# reindex master dataframe on date time values
# master_df.index = master_df['month']

# labor force participation rate seasonally adjusted data
```

```
lfpr_path = 'data/lfpr.xlsx'
lfpr_df = pd.read_excel(lfpr_path)
# reformat month column
lfpr_df['month'] = lfpr_df['month'].apply(lambda x: datetime.strptime(x, '%Y␣

# copy master2 to master3
master3_df = master2_df.copy()
# merge to master3 and export to excel
master3_df = master3_df.merge(lfpr_df, on=['month'], how='inner')
master3_df.to_excel(r'data/master3_data.xlsx', index=False, header=True)
```

9 Stata Code

Listing 2: OLS

```
use /Users/henryjohnson/PycharmProjects/pythonProject1/data/master3_data.dta

xtset month
reg ssa_total_number ssa_average_amount gspc_adj_close_diff_r

summarize month dji_open dji_high dji_low dji_close dji_adj_close dji_volume
gspc_open gspc_high gspc_low gspc_close gspc_adj_close gspc_volume ///
federal_funds_rate cpi ssa_total_number ssa_average_amount

summarize ssa_total_number

//
/*
```

```
ssa_total_number ssa_total_number_diff ssa_average_amount ssa_average_amount
*/
reg ssa_total_number ssa_average_amount ssa_average_amount_diff_n ssa_average


drop if _n == 1


reg ssa_total_number ssa_average_amount dji_adj_close_r dji_close_r dji_high_


generate ssa_total_number_increase = 0
replace ssa_total_number_increase = 1 if ssa_total_number_diff > 0


logistic ssa_total_number_increase ssa_average_amount dji_adj_close_r dji_clo


logit ssa_total_number_increase ssa_average_amount dji_adj_close_r dji_close_


codebook ssa_total_number_increase


logit ssa_total_number_increase ssa_average_amount dji_adj_close_r



dfuller gspc_volume, lags(3) trend regress



logit ssa_total_number_increase dji_close_r_increased ssa_average_amount_r_in


logistic ssa_total_number_increase dji_open_r_increased dji_close_r_increased
logistic ssa_total_number dji_open_r_increased dji_close_r_increased ssa_aver




tsset month
histogram lfpr, frequency normal
// lfpr is for 55+ group

histogram ssa_total_number, frequency normal

histogram gspc_open_r_dpercent, frequency normal



reg ssa_total_number ssa_average_amount_r federal_funds_rate cpi dji_close_r

reg ssa_total_number ssa_average_amount_r federal_funds_rate cpi lfpr, robust



reg ssa_total_number dji_open_r dji_open_r_increased gspc_open_r gspc_open_r_
gen ln_lfpr = log(lfpr)
reg ssa_total_number dji_volume gspc_close_r gspc_volume ssa_average_amount_r
vif
```

```
probit ssa_total_number_increase dji_open_r_increased ssa_average_amount_r gs
vif uncentered




reg ssa_total_number ssa_average_amount_r federal_funds_rate cpi dji_close_r


gen gspc_x_dji = gspc_close_r * dji_close_r
gen sqrt_gspc_x_dji = sqrt(gspc_x_dji)
gen sqrt_gspc_x_dji_n = sqrt(gspc_close * dji_close)


reg ssa_total_number ssa_average_amount_r gspc_close_r lfpr , r
reg ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji federal_funds_rate
vif


probit ssa_total_number_increased ssa_average_amount_r gspc_close_r lfpr , r


reg ssa_total_number ssa_average_amount sqrt_gspc_x_dji_n federal_funds_rate
vif




// this model looks pretty good
xtset month
xtsum ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji federal_funds_rat
gen sqrt_gspc_x_dji = sqrt(gspc_close * dji_close)
gen sqrt_gspc_r = sqrt(gspc_close_r)
gen sqrt_dji_r = sqrt(dji_close_r)
gen market = dji_close_r + gspc_close_r
gen normalized_gspc = (gspc_close_r -763.7345) / (3779.061-763.7345)
gen normalized_dji = (dji_close_r -5991.788) / (30793.82-5991.788)
gen dji_plus_gspc = normalized_dji + normalized_gspc
reg ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji dji_plus_gspc feder


probit ssa_total_number_increase ssa_average_amount_r sqrt_gspc_x_dji dji_plu


logit ssa_total_number_increase ssa_average_amount ssa_average_amount_diff_n
predict ssa_total_number_increased


logit ssa_total_number_increase ssa_average_amount_r dji_close_r_increased fe



logit ssa_total_number_increase ssa_average_amount dji_adj_close dji_adj_clos
gspc_adj_close_r gspc_close gspc_close_diff_n gspc_close_diff_r gspc_close_r


estat classification , cutoff(.57)


logit ssa_total_number_increase dji_open_r_increased ssa_average_amount_r gsp
estat classification , cutoff(.57)
```

```
vif
reg ssa_total_number ssa_average_amount_r gspc_close_r federal_funds_rate lfp
reg ssa_total_number dji_close_r federal_funds_rate lfpr , r
eststo: reg ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji federal_fun
estout
vif
graph twoway (lfit ssa_total_number sqrt_gspc_x_dji) (scatter ssa_total_numbe
graph twoway (lfit ssa_total_number federal_funds_rate) (scatter ssa_total_nu

collin ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji federal_funds_ra

twoway (scatter ssa_total_number ssa_average_amount_r) (lfit ssa_total_number
acprplot ssa_average_amount_r , lowess
kdensity ssa_average_amount_r , normal
// may be good to transform ssa_average_amount_r to log to help correct with
gen ln_ssa_average_amount_r = log(ssa_average_amount_r)
// compare the new model to the old model
reg ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji federal_funds_rate
reg ssa_total_number ln_ssa_average_amount_r sqrt_gspc_x_dji federal_funds_ra
// check skewness again
kdensity ln_ssa_average_amount_r , normal
acprplot ln_ssa_average_amount_r , lowess



twoway (scatter ssa_total_number sqrt_gspc_x_dji) (lfit ssa_total_number sqrt
acprplot sqrt_gspc_x_dji , lowess
kdensity sqrt_gspc_x_dji , normal

twoway (scatter ssa_total_number federal_funds_rate) (lfit ssa_total_number f
acprplot federal_funds_rate , lowess
kdensity federal_funds_rate , normal
// try to fix skewness
gen ln_federal_funds_rate = log(federal_funds_rate)
kdensity ln_federal_funds_rate , normal // doesn't seem to have helped

twoway (scatter ssa_total_number lfpr) (lfit ssa_total_number lfpr) (lowess s
acprplot lfpr , lowess
kdensity lfpr , normal
predict r, resid
kdensity r, normal
pnorm r



graph matrix ssa_average_amount_r lfpr , half



linktest // _hatsq is statistically significant when it shouldn't be so we re
ovtest // the model is mispecified by the ovtest
// Note that the RESET test only tests whether the right functional form has
```

```
tsset month
gen date = tm(1992m1) + _n-1
format %tm date
tsset date
sum

gen sqrt_gspc_x_dji = sqrt(gspc_close * dji_close)
gen sqrt_gspc_r = sqrt(gspc_close_r)
gen sqrt_dji_r = sqrt(dji_close_r)
gen market = dji_close_r + gspc_close_r
gen normalized_gspc = (gspc_close_r-763.7345) / (3779.061-763.7345)
gen normalized_dji = (dji_close_r-5991.788) / (30793.82-5991.788)
gen dji_plus_gspc = normalized_dji + normalized_gspc
gen ln_ssa_total_number = ln(ssa_total_number)
gen ln_sqrt_gspc_x_dji = ln(sqrt_gspc_x_dji)
gen ln_dji_plus_gspc = ln(dji_plus_gspc)

sum
// inspect for autocorrelation
tsline ssa_total_number if tin(1992m1, 2020m12)

twoway (tsline ssa_total_number, lwidth(medium)) (tsline S12.ssa_total_number
twoway (tsline ssa_total_number, lwidth(medium)) (tsline S4.ssa_total_number,
twoway (tsline ssa_total_number, lwidth(medium)) (tsline S4.ssa_total_number,

gen ln_ssa_total_number2 =  ln(ssa_total_number)
```

```
twoway (tsline ln_ssa_total_number2) (tsline ssa_total_number, yaxis(2))
tsline D.ln_ssa_total_number2


forvalues p = 2/12 {
        tsline S(2/`p').ln_ssa_total_number2 if tin(1992m1, 1995m1)
}


tsline S12.ln_ssa_total_number2
// tsline S(0/`p'.ln_ssa_total_number2


forvalues p = 1/40 {
        qui reg S12.L(0/`p').ssa_total_number if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}


foreach var of varlist {
  dfuller `var', reg lags(0) drift
}


gen varlist =



dfuller ssa_total_number, reg lags(3) trend


tsline ssa_total_number
dfuller ssa_total_number, reg lags(3) drift
```

```
gen num_ssa_d1 = ssa_total_number - L1.ssa_total_number
gen num_ssa_d1_12 = num_ssa_d1 - L12.num_ssa_d1
gen num_ssa_d1_12_9 = num_ssa_d1_12 - L9.num_ssa_d1_12
gen num_ssa_d1_12_9_4 = num_ssa_d1_12_9 - L4.num_ssa_d1_12_9
gen num_ssa_d_perc = num_ssa_d1_12_9_4 / ssa_total_number
tsline num_ssa_d_perc if tin(1994m1, 1999m1)

dfuller num_ssa_d_perc, reg lags(0) trend
dfuller num_ssa_d_perc, reg lags(0) drift

tsline num_ssa_d_perc2

gen num_ssa_d1_12_9_3 = num_ssa_d1_12_9 - L3.num_ssa_d1_12_9
gen num_ssa_d_perc2 = num_ssa_d1_12_9_3 / ssa_total_number
dfuller num_ssa_d_perc2, reg lags(0) trend



tssmooth ma ssa_perc_ma111 = num_ssa_d_perc, window(1 1 1)
tsline ssa_perc_ma111

tssmooth ma ssa_perc_ma212 = num_ssa_d_perc, window(2 1 2)
tsline ssa_perc_ma212
dfuller D.ssa_perc_ma212, reg lags(0) trend

tssmooth ma ssa_perc_ma312 = num_ssa_d_perc, window(3 1 2)
```

```
dfuller D.ssa_perc_ma312, reg lags(0) trend


tssmooth ma ssa_perc_ma512 = num_ssa_d_perc, window(5 1 2)
dfuller ssa_perc_ma512, reg lags(0) trend



tssmooth ma ssa_perc_ma912 = num_ssa_d_perc, window(9 1 2)
dfuller ssa_perc_ma912_w, reg lags(0) trend


tssmooth ma ssa_perc_ma812 = num_ssa_d_perc, window(8 1 2)
dfuller D.ssa_perc_ma812, reg lags(0) trend
//, weights(1 <2> 1)
dfuller ssa_perc_ma912_w, reg lags(0) trend


tssmooth ma ssa_perc_ma1112 = num_ssa_d_perc, window(11 1 2)
tssmooth ma ssa_perc_ma1112_111_w = ssa_perc_ma1112, weights(1 <2> 1)
dfuller D1.ssa_perc_ma1112, reg lags(0) trend
dfuller ssa_perc_ma1112, reg lags(0) trend
dfuller ssa_perc_ma1112_111_w, reg lags(0) trend
tsline ssa_perc_ma1112_111_w


// this is the best one so far!
tssmooth ma ssa_perc_ma1113 = num_ssa_d_perc, window(11 1 3)
dfuller ssa_perc_ma1113, reg lags(0) trend


tssmooth ma ssa_perc_ma1103 = num_ssa_d_perc, window(11 0 3)
dfuller D.ssa_perc_ma1103, reg lags(0) trend
```

```
tssmooth ma ssa_perc_ma1114 = num_ssa_d_perc, window(11 1 4)
dfuller ssa_perc_ma1114, reg lags(0) trend

tssmooth ma ssa_perc_ma1115 = num_ssa_d_perc, window(11 1 5)
dfuller ssa_perc_ma1115, reg lags(0) trend

// STATIONARY!!!!
tssmooth ma ssa_perc_ma1203 = num_ssa_d_perc, window(12 0 3)
dfuller ssa_perc_ma1203, reg lags(0) trend

forvalues p = 1/8 {
        qui reg D.L(0/`p').ssa_perc_ma1203 if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}
// 1 lag

// still stationary!
tssmooth ma ssa_perc_ma1200 = num_ssa_d_perc, window(12 0 0)
dfuller ssa_perc_ma1200, reg lags(0) trend
dfuller ssa_perc_ma1200, reg lags(0) drift

forvalues p = 1/13 {
        qui reg D.L(0/`p').ssa_perc_ma1200 if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}

tsline ssa_perc_ma1200 if tin(1994m6, 2020m11)

// not stationary
tssmooth ma ssa_perc_ma1100 = num_ssa_d_perc, window(11 0 0)
dfuller ssa_perc_ma1100, reg lags(0) trend

// try to run it now
reg D.ssa_perc_ma1200 D.L.ssa_perc_ma1200 ssa_average_amount_r L(0/1).sqrt_gs
predict forecasts7
list forecasts7 if tin(1994m6, 2020m11)
// -.0070285
list D.ssa_perc_ma1200 if tin(1994m6, 2020m11)
// -.028061



reg ssa_perc_ma1203 D.L(1/3).ssa_perc_ma1203

// dfuller num_ssa_d_perc

ac D.ssa_total_number, lags(8)
// gen(ac8_ssa_total_amount)
list ac_ssa_total_amount in 1/4

tsline D8.ssa_total_number if tin(1992m1, 2020m12)
```

```
forvalues p = 1/8 {
        qui reg D.L(0/`p').ssa_total_number if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}
// 3 lags looks the best according to BIC

// de-trend it
// dictate seasons seasonality

reg D.ssa_total_number D.L(1/3).ssa_total_number if tin(1992m1, 2020m11), r
dfuller D.ssa_total_number, reg lags(3) drift
dfuller D.ssa_total_number, reg lags(3) trend
dfuller D.ssa_total_number_perc_change, reg lags(3) drift
dfuller D.ssa_total_number_perc_change, reg lags(3) trend

dfuller S4.ssa_total_number_perc_change, reg lags(3) trend
dfuller S12.ssa_total_number_perc_change, reg lags(3) trend
dfuller S4.ssa_total_number, reg lags(3) trend
dfuller S12.ssa_total_number, reg lags(3) trend

// shows that there's a trend
reg ssa_total_number month if tin(1992m1, 2020m11), r

// shows that there's no longer a time trend
reg D.ssa_total_number month if tin(1992m1, 2020m11), r
```

```
// detrending attempt
gen ssa_total_number_diff2 = D.ssa_total_number
reg ssa_total_number_diff2 month if tin(1992m1, 2020m11), r
reg ssa_total_number_perc_change month if tin(1992m1, 2020m11), r

predict ssa_total_number_perc_change_p, residual
predict ssa_total_number_diff2_predict, residual
tsline ssa_total_number_perc_change_p
dfuller ssa_total_number_perc_change_p, reg lags(3) trend
dfuller D.ssa_total_number_perc_change_p, reg lags(3) trend
// according to the test the p-value is .868, still WAY too high

list month ssa_total_number L1.ssa_total_number ssa_total_number_diff2 in 1/1

reg ssa_total_number_perc_change_p ssa_average_amount_r D.L(1/3).sqrt_gspc_x_


gen ssa_total_number_perc_change = ssa_total_number_diff / ssa_total_number

reg D.ssa_total_number_perc_change D.L(1/3).ssa_total_number if tin(1992m1, 2

reg D.ssa_total_number D.L(1/3).ssa_total_number if tin(1992m1, 2020m11), r
predict forecasts

reg D.ssa_total_number D.L(1/3).ssa_total_number D.L(1/3).ssa_average_amount_
predict forecasts2
```

```
reg D.ssa_total_number D.L(1/3).ssa_total_number ssa_average_amount_r sqrt_gs
predict forecasts3

reg D.ssa_total_number D.L(1/3).ssa_total_number ssa_average_amount_r D.L(1/3
predict forecasts4

reg ssa_total_number D.L(1/3).ssa_total_number ssa_average_amount_r sqrt_gspc
reg ssa_total_number lfpr if tin(1992m1, 2020m11), r
dfuller ssa_total_number, reg lags(3) trend
dfuller D.ssa_total_number, reg lags(3) trend

// reg ssa_total_number ssa_average_amount_r sqrt_gspc_x_dji dji_plus_gspc fe

dfgls ssa_total_number
// pass for lags 1-7 at 1%
dfgls ssa_total_number, notrend
// fail for everything
dfgls D.ssa_total_number, notrend
// pass for lags 1-7, 10 at 1%

pperron ssa_total_number
pperron ssa_total_number, trend
pperron ssa_total_number, trend lags(7)
// passed

dfuller ssa_total_number
// passed
```

```
dfuller ssa_total_number, reg lags(0) trend
// passed
dfuller ssa_total_number, reg lags(1) trend
// passed
dfuller ssa_total_number, reg lags(3) trend
// passed
dfuller ssa_total_number, reg lags(7) trend
// passed for 1-4 at 5%

// find ideal lags
forvalues p = 1/7 {
        qui reg L(0/`p').ssa_total_number if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}
// for some reason, it keeps getting lower. Though there appears to be a kink
// only 1 lag may be best

dfgls ssa_average_amount_r
// pass for 1-3 at 1% critical value
dfgls ssa_average_amount_r, notrend
// failed for no trend
dfgls D.ssa_average_amount_r, notrend
// passed for first difference with no trend up to the 11th lag for the 1% cr

// find ideal lags
forvalues p = 0/3 {
        qui reg L(0/`p').ssa_average_amount_r if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}
// maybe just 1 lag?



dfgls sqrt_gspc_x_dji
// failed
dfgls D.sqrt_gspc_x_dji
// passed for 1-14 at 1%
dfgls D.sqrt_gspc_x_dji, notrend
// passed for 1-14 at 1%
forvalues p = 0/14 {
        qui reg D.L(0/`p').sqrt_gspc_x_dji if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}
// looks like there's a kink in the second lag and the 7th


dfgls dji_plus_gspc
// failed
dfgls D.dji_plus_gspc
// passed for 1-14 at 1%
dfgls D.dji_plus_gspc, notrend
// passed for 1-14 at 1%
forvalues p = 0/14 {
        qui reg D.L(0/`p').sqrt_gspc_x_dji if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}

dfgls federal_funds_rate
// passed for 8-9 at 1%, 5-12 at 5%, 4-14 at 10%
dfgls D.federal_funds_rate
// passed for 1-6 and 8-16 at 1%
dfgls D.federal_funds_rate, notrend
// passed at 1%

forvalues p = 0/14 {
        qui reg D.L(0/`p').federal_funds_rate if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}

dfgls lfpr
// failed
dfgls D.lfpr
// passed for 1-10 at 1%
dfgls D.lfpr, notrend
// passed for 1-5 at 1%
forvalues p = 0/5 {
        qui reg D.L(0/`p').lfpr if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}



reg D.ssa_total_number D.L.ssa_total_number L.ssa_total_number D.sqrt_gspc_x_
predict prediction3
predict predictions3 if date>tm(2019m11)
tsline predictions3 D.ssa_total_number if date>tm(2019m11)
dis (prediction3 + ssa_total_number)
list prediction3 L.ssa_total_number D.ssa_total_number if tin(2020m11, 2020m1
list prediction2 D.ssa_total_number if tin(2020m11, 2020m12)
tsline prediction3 D.ssa_total_number if tin(2017m11, 2020m12)

mean prediction ssa_total_number_diff
predict predictions if date>tm(2020m11)

forvalues p = 1/7 {
        qui reg D.L(0/`p').ssa_total_number if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}

dfuller ssa_average_amount_r, reg lags(3) trend
dfgls ssa_total_number, trend
```

```
reg D.ln_ssa_total_number D.L(1/3).ln_ssa_total_number ssa_average_amount_r l
predict forecasts6
// but the R—squared falls a lot

forvalues p = 1/8 {
        qui reg D.L(0/`p').ln_ssa_total_number if tin(1992m1, 2020m11), r
        display "p = `p'"
        estat ic
}
// with ln(ssq_total_number), one lag appears to be the best
reg D.ln_ssa_total_number D.L.ln_ssa_total_number ssa_average_amount_r ln_sqr
// but the R—squared falls a lot

list forecasts forecasts2 forecasts3 forecasts4 forecasts5 forecasts6 month i
// tsline forecasts D.ssa_total_number if tin(1992m1, 2020m11)
// tsline forecasts2 D.ssa_total_number if tin(1992m1, 2020m11)
// tsline forecasts3 D.ssa_total_number if tin(1992m1, 2020m11)
tsline forecasts5 D.ssa_total_number if tin(2015m1, 2020m11)

list forecasts5 if tin(1992m1, 2020m11)

// −43536.2 for 1st forecasts but 10739.66 for 5th——this is the predicted cha
list ssa_total_number
// 384598 to 410127=25529 ——actual change in filers from 2020m11—2020m12
// 14790 off or 3.6% off
//
```