```
. cd "C:\Users\rxg230005\Downloads"
C:\Users\rxg230005\Downloads

. import delimited "AETsA_Project_dataset.csv", clear
(encoding automatically selected: ISO-8859-1)
(8 vars, 132 obs)
```

Define the regression model and **diagnostic tests**.

Basic OLS Regression

```
. regress houses_sold interest_rate household_income unemployment_rate housing_supply inflation_rate consumer_confidence

      Source |       SS           df       MS      Number of obs   =       132
-------------+----------------------------------   F(6, 125)       =     64.75
       Model |  1565341.83         6  260890.305   Prob > F        =    0.0000
    Residual |  503683.888       125  4029.47111   R-squared       =    0.7566
-------------+----------------------------------   Adj R-squared   =    0.7449
       Total |  2069025.72       131  15794.0895   Root MSE        =    63.478

------------------------------------------------------------------------------------
       houses_sold | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------------+----------------------------------------------------------------
     interest_rate |   3.701492   11.07831     0.33   0.739    -18.22386    25.62685
  household_income |   .0069901   .0031841     2.20   0.030     .0006883    .0132918
 unemployment_rate |   28.77898   5.030282     5.72   0.000     18.82343    38.73453
    housing_supply |   .4328557   .0767416     5.64   0.000     .2809747    .5847368
    inflation_rate |  -63.10927   23.03191    -2.74   0.007    -108.6923   -17.52625
consumer_confidence |  4.099199   .7411266     5.53   0.000     2.632418    5.565981
             _cons |  -801.3235   176.9307    -4.53   0.000    -1151.491   -451.1557
------------------------------------------------------------------------------------
```

Multicollinearity Check

Use the Variance Inflation Factor (VIF) to check for multicollinearity.

```
. vif

    Variable |       VIF       1/VIF
-------------+----------------------
household_~e |     27.04    0.036982
interest_r~e |     13.77    0.072607
housing_su~y |      9.48    0.105462
consumer_c~e |      3.36    0.298055
unemployme~e |      2.40    0.415952
inflation_~e |      2.28    0.438385
-------------+----------------------
    Mean VIF |      9.72
```

Autocorrelation Test

The Breusch-Godfrey test can identify autocorrelation in the residuals for time series data

```
. gen date_var = date(date,"MDY")

. tsset date_var

Time variable: date_var, 19724 to 23711, but with gaps
        Delta: 1 unit

. estat bgodfrey

Number of gaps in sample = 131

Breusch–Godfrey LM test for autocorrelation
```

lags(*p*)	chi2	df	Prob > chi2
1	0.000	1	1.0000

```
                        H0: no serial correlation
```

The Durbin-Watson test can identify autocorrelation in the residuals.

```
. estat dwatson

Number of gaps in sample = 131

Durbin–Watson d-statistic(  7,   132) =          0
```

Heteroskedasticity Test

The Breusch-Pagan test is useful for identifying heteroskedasticity.

```
. estat hettest

Breusch–Pagan/Cook–Weisberg test for heteroskedasticity
Assumption: Normal error terms
Variable: Fitted values of houses_sold

H0: Constant variance

    chi2(1) =  33.07
Prob > chi2 = 0.0000
```

Adding Interaction Term

```
. gen interest_income_interaction = interest_rate * household_income

. regress houses_sold interest_rate household_income unemployment_rate housing_supply inflation_rate consumer_confidence interest_income
> _interaction

      Source |       SS           df       MS      Number of obs   =       132
-------------+----------------------------------   F(7, 124)       =     55.06
       Model |  1565368.04         7  223624.005   Prob > F        =    0.0000
    Residual |  503657.685       124  4061.75552   R-squared       =    0.7566
-------------+----------------------------------   Adj R-squared   =    0.7428
       Total |  2069025.72       131  15794.0895   Root MSE        =    63.732

---------------------------------------------------------------------------------------------
                houses_sold | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
----------------------------+----------------------------------------------------------------
              interest_rate |   8.570014   61.62616     0.14   0.890    -113.4054    130.5454
           household_income |    .006954   .0032282     2.15   0.033     .0005645    .0133435
          unemployment_rate |   28.91618   5.331432     5.42   0.000     18.36378    39.46857
             housing_supply |   .4332339   .0771921     5.61   0.000     .2804492    .5860186
             inflation_rate |  -62.98594   23.17492    -2.72   0.008    -108.8556   -17.11628
        consumer_confidence |   4.060783   .8845496     4.59   0.000     2.310012    5.811555
interest_income_interaction |  -.0000584   .0007268    -0.08   0.936    -.0014968    .0013801
                      _cons |  -797.7742    183.052    -4.36   0.000    -1160.085   -435.4631
---------------------------------------------------------------------------------------------
```

Time Series Analysis

For time series-specific analysis, start by setting up the dataset as a time series and then perform any necessary transformations or tests for stationarity.

```
. gen numeric_date = date(date, "MDY")

. format numeric_date %td

. gen year_month = ym(year(numeric_date), month(numeric_date))

. format year_month %tm

. tsset year_month, monthly

Time variable: year_month, 2014m1 to 2024m12
        Delta: 1 month
```

```
. dfuller houses_sold, regress lags(1)

Augmented Dickey-Fuller test for unit root

Variable: houses_sold                     Number of obs  = 130
                                          Number of lags =   1

H0: Random walk without drift, d = 0

                                       Dickey-Fuller
                   Test     ──────── critical value ─────────
              statistic           1%           5%          10%
──────────────────────────────────────────────────────────────
 Z(t)            -2.435       -3.500       -2.888       -2.578
──────────────────────────────────────────────────────────────
MacKinnon approximate p-value for Z(t) = 0.1321.

Regression table
──────────────────────────────────────────────────────────────────────────────
D.           │
 houses_sold │ Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
─────────────┼────────────────────────────────────────────────────────────────
 houses_sold │
         L1. │  -.0907553   .0372722    -2.43   0.016    -.1645103   -.0170003
         LD. │  -.1190492   .0872424    -1.36   0.175    -.2916861    .0535877
             │
       _cons │   60.73966   24.22354     2.51   0.013     12.80566    108.6737
──────────────────────────────────────────────────────────────────────────────


. dfuller interest_rate, regress lags(1)

Augmented Dickey-Fuller test for unit root

Variable: interest_rate                   Number of obs  = 130
                                          Number of lags =   1

H0: Random walk without drift, d = 0

                                       Dickey-Fuller
                   Test     ──────── critical value ─────────
              statistic           1%           5%          10%
──────────────────────────────────────────────────────────────
 Z(t)            -0.690       -3.500       -2.888       -2.578
──────────────────────────────────────────────────────────────
MacKinnon approximate p-value for Z(t) = 0.8492.

Regression table
──────────────────────────────────────────────────────────────────────────────
D.            │
interest_rate │ Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
──────────────┼───────────────────────────────────────────────────────────────
interest_rate │
          L1. │  -.0044768   .0064834    -0.69   0.491    -.0173063    .0083527
          LD. │    .647095   .0687599     9.41   0.000     .5110316    .7831584
              │
        _cons │   .0201748   .0157534     1.28   0.203    -.0109983    .0513479
──────────────────────────────────────────────────────────────────────────────
```

```
. gen D_houses_sold = D.houses_sold
(1 missing value generated)

. gen D_interest_rate = D.interest_rate
(1 missing value generated)

. gen D_unemployment_rate = D.unemployment_rate
(1 missing value generated)
```

```
. regress D_houses_sold D_interest_rate D_unemployment_rate D.household_income D.housing_supply D.inflation_rate D.consumer_confidence

      Source |       SS           df       MS      Number of obs   =       131
-------------+----------------------------------   F(6, 124)       =      1.72
       Model |  28276.9001         6  4712.81668   Prob > F        =    0.1226
    Residual |  340571.787       124  2746.54667   R-squared       =    0.0767
-------------+----------------------------------   Adj R-squared   =    0.0320
       Total |  368848.687       130  2837.29759   Root MSE        =    52.408

------------------------------------------------------------------------------------
     D_houses_sold | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------------+----------------------------------------------------------------
   D_interest_rate |   8.704054   32.54878     0.27   0.790     -55.7191    73.1272
D_unemployment_rate|  -7.430871   5.741127    -1.29   0.198    -18.79417   3.932428
                   |
  household_income |
               D1. |  -.0312337   .0331022    -0.94   0.347    -.0967523   .0342849
                   |
    housing_supply |
               D1. |   -.021642   .0663343    -0.33   0.745    -.1529362   .1096523
                   |
    inflation_rate |
               D1. |   35.55543   41.20835     0.86   0.390    -46.00745   117.1183
                   |
consumer_confidence|
               D1. |   1.965289   1.235161     1.59   0.114    -.4794412    4.41002
                   |
             _cons |   8.938533   8.263704     1.08   0.282    -7.417652   25.29472
------------------------------------------------------------------------------------
```

Forecasting

Depending on your preference, you can use regression-based forecasting or apply ARIMA if only the dependent variable needs to be forecasted.

*** Predict future values of D_houses_sold**

*** To accumulate the forecast for `houses_sold`, use cumulative summing if differenced**

```
. predict D_houses_sold_forecast, xb
(1 missing value generated)

. gen forecast_houses_sold = sum(D_houses_sold_forecast)
```

ARIMA Forecasting

If you want to forecast houses_sold based on past values, ARIMA modeling is a good option

```
. arima houses_sold, arima(1,1,1)

(setting optimization to BHHH)
Iteration 0:  Log likelihood = -705.35846
Iteration 1:  Log likelihood = -704.39013
Iteration 2:  Log likelihood = -704.30142
Iteration 3:  Log likelihood = -704.04786
Iteration 4:  Log likelihood = -702.99359
(switching optimization to BFGS)
Iteration 5:  Log likelihood = -702.71412
Iteration 6:  Log likelihood = -702.50752
Iteration 7:  Log likelihood = -702.50432
Iteration 8:  Log likelihood = -702.44322
Iteration 9:  Log likelihood = -702.41194
Iteration 10: Log likelihood = -702.40847
Iteration 11: Log likelihood = -702.40822
Iteration 12: Log likelihood = -702.40822

ARIMA regression

Sample: 2014m2 thru 2024m12                     Number of obs     =        131
                                                Wald chi2(2)      =     561.88
Log likelihood = -702.4082                      Prob > chi2       =     0.0000
```

D. houses_sold	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
houses_sold						
_cons	2.117021	1.247306	1.70	0.090	-.3276534	4.561696
ARMA						
ar L1.	.8604672	.0553164	15.56	0.000	.752049	.9688854
ma L1.	-1.000002	342.5392	-0.00	0.998	-672.3646	670.3646
/sigma	51.10379	8752.886	0.01	0.498	0	17206.44

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.

.
```

Forecast 12 periods ahead

```
. predict houses_sold_forecast
(option xb assumed; predicted values)

. list houses_sold_forecast

     +-----------+
     | houses_~t |
     |-----------|
  1. |  2.117021 |
  2. |  2.117021 |
  3. |  3.869345 |
  4. |  5.027139 |
  5. |  5.171371 |
     |-----------|
  6. |  1.240043 |
  7. |  4.272402 |
  8. |  5.751759 |
  9. |  .8852222 |
 10. | -.1580938 |
     |-----------|
 11. | -.4241638 |
 12. |  3.176394 |
```

Forecast Accuracy

Calculate Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for accuracy

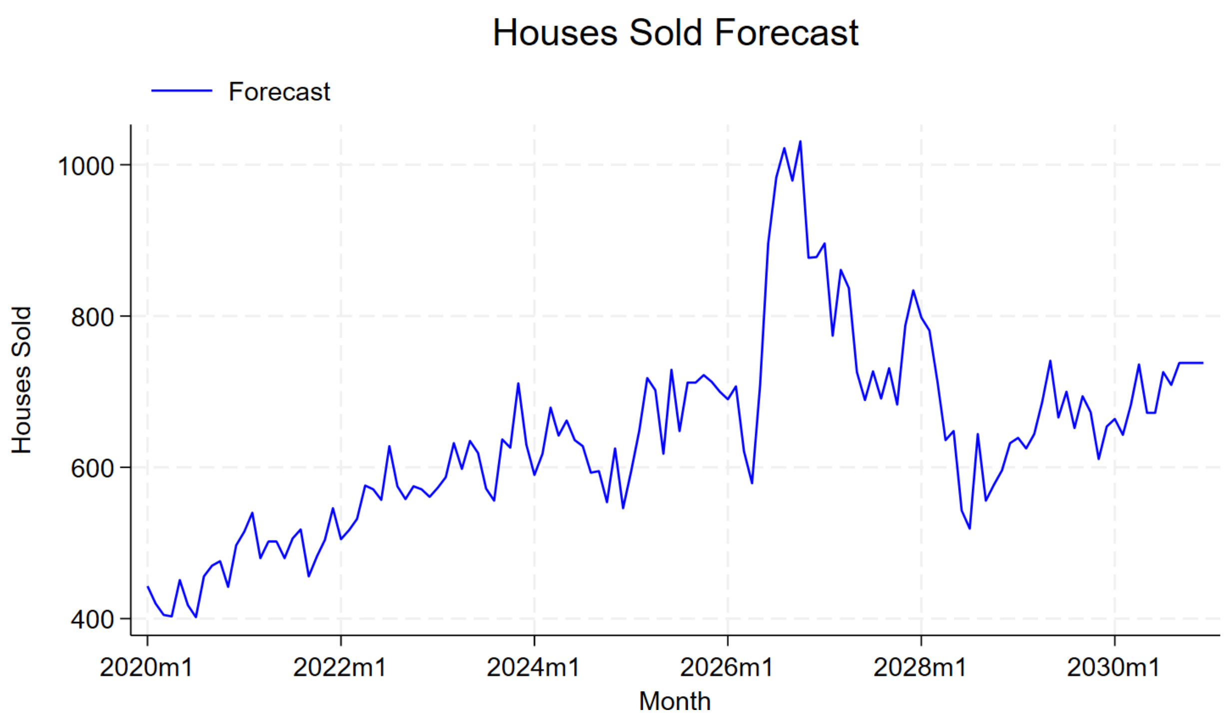
```
. gen forecast_error = houses_sold - houses_sold_forecast

. summarize forecast_error, meanonly

. display "MAE: " r(mean)
MAE: 638.64979

. gen squared_error = forecast_error^2

. summarize squared_error, meanonly

. display "RMSE: " sqrt(r(mean))
RMSE: 652.0177
```

Graphs

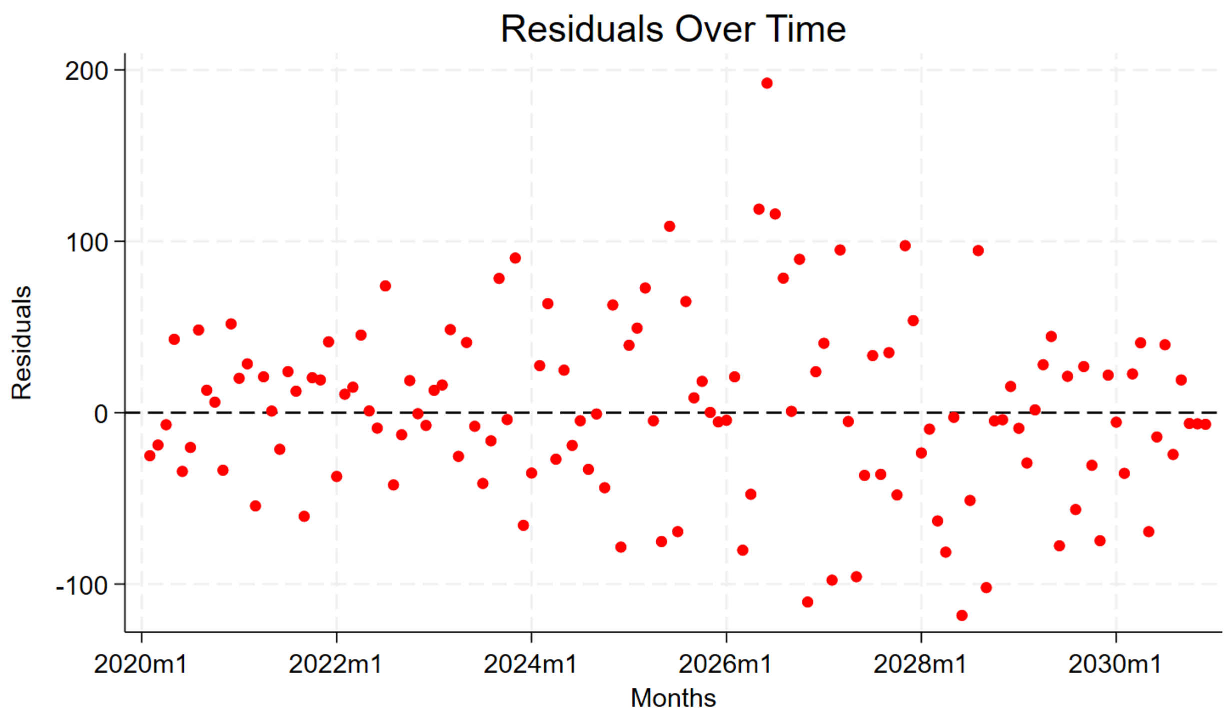
```
. gen time = tm(2020m1) + _n - 1
```

```
. format time %tm
```

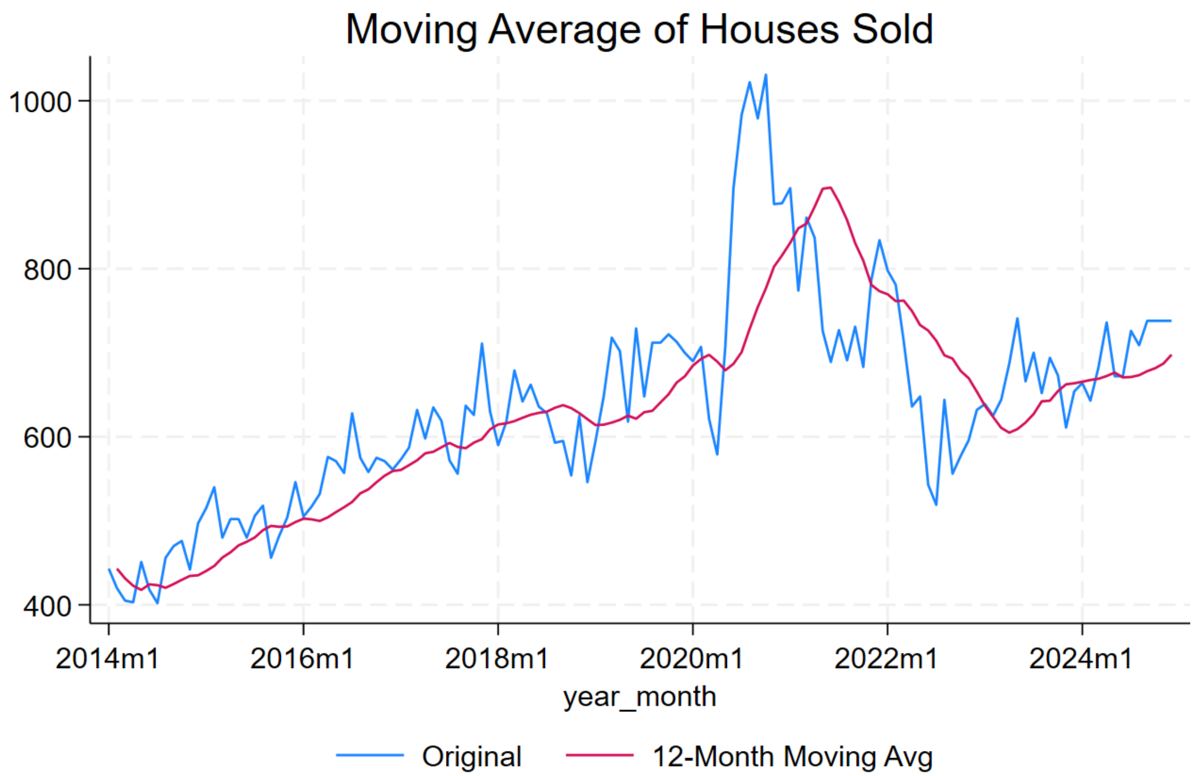
```
. twoway (line houses_sold time, lcolor(blue) lpattern(solid)) (line houses_sold_forecast time if _n > _N, lcolor(red) lpattern(dash)), title("Houses Sold Forecast") xtitle("Month") ytitle("Houses Sold") legend(order(1 "Forecast") position(11)) xlabel(, format(%tm)) ylabel(, grid)
```



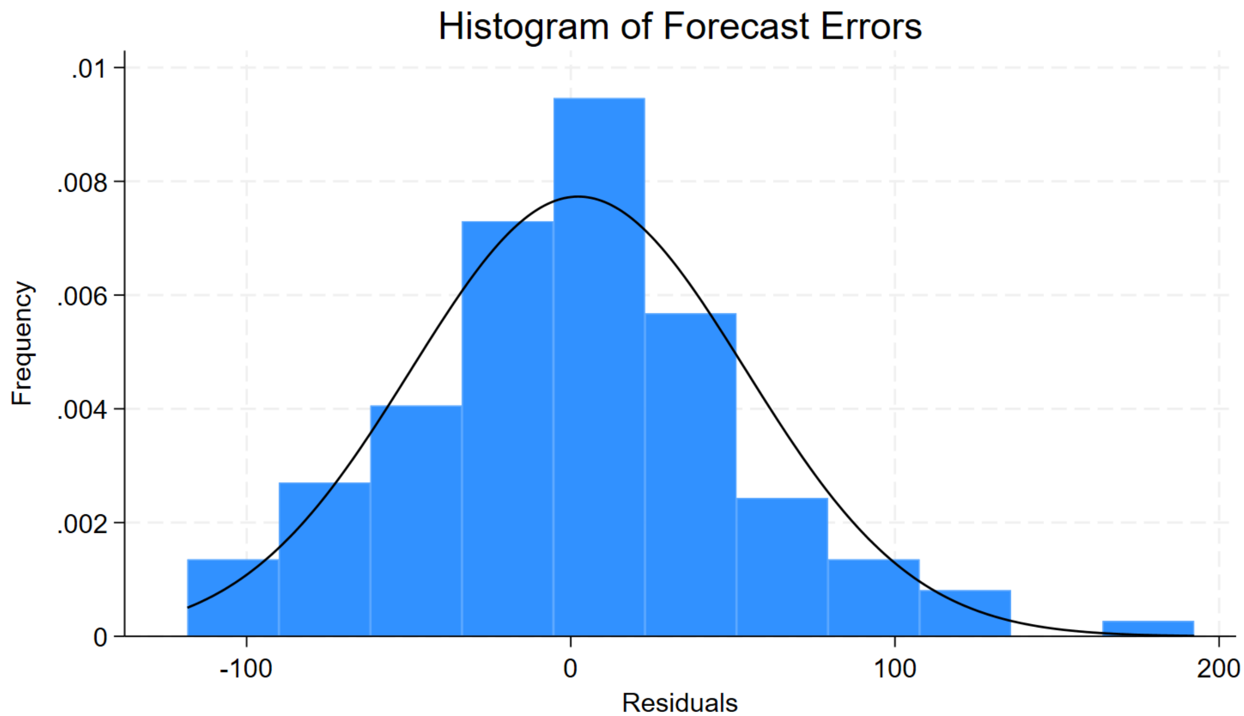
```
predict residuals, resid
(1 missing value generated)

. scatter residuals time, mcolor(red) title("Residuals Over Time") yline(0, lcolor(black)) xtitle("Months") ytitle("Residuals")
```



```
. tssmooth ma moving_avg = houses_sold, window(12)

. tsline houses_sold moving_avg, title("Moving Average of Houses Sold") legend(order(1 "Original" 2 "12-Month Moving Avg"))
```



```
histogram residuals, normal title("Histogram of Forecast Errors") xtitle("Residuals") ytitle("Frequency")
```



```
twoway (line houses_sold time if time <= tm(2023m12), lcolor(blue) lpattern(solid) yaxis(1)) (line interest_rate time if time <= tm(2023m12), lcolor(red) lpattern(dash) yaxis(2)), title("Houses Sold and Interest Rate Over Time") xtitle("Month") ytitle("Houses Sold", axis(1)) ytitle("Interest Rate (%)", axis(2)) legend(order(1 "Houses Sold" 2 "Interest Rate")) xlabel(, format(%tm))
```