Forecast of average weekly earnings and total employment for North Port-Sarasota-Bradenton MSA

Joshua Cantera

Abstract

The employment and average weekly earnings are important indicators of a Metropolitan Statistical Area's economic condition. Because of this, being able to produce a trustworthy forecast is important for both state and local officials to determine where resources need to be located. This report explains the process used to estimate forecasts of employment and average weekly earnings for the North Port-Sarasota-Bradenton MSA in Florida for the month of March in 2020.

Introduction

Predicting employment and average weekly earnings will allow city and state officials to properly allocate resources according to the economic strength of different regions. This report will focus on forecasting the average weekly earnings (in dollars per week) and employment (in thousands of persons) for the Metropolitan Statistical Area (MSA) of North Port-Sarasota-Bradenton for the month of March in the year 2020.

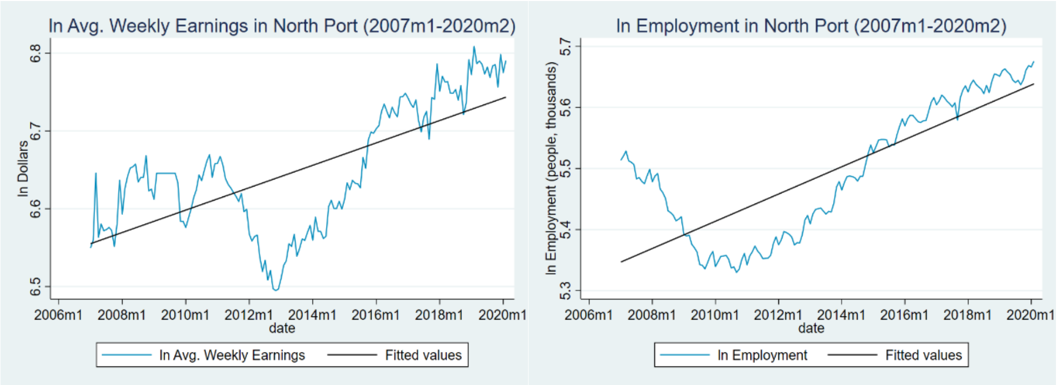
The models used for this forecast will use data collected by the Federal Reserve and other governmental entities. Full lists of variables used can be found later in this report under the title *Data*. Most of the variables used have only started being collect in 2007 so there is only a total of 13 years of data available for this forecast, which will be enough in this case.

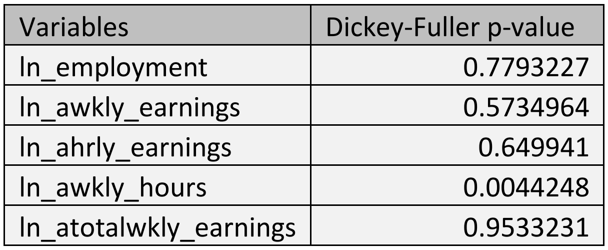
Data

Variables:

employment:	Employment - thousands of people
awkly_earnings:	Average weekly earnings - dollars per week
ahrly_earnings:	Average hourly earnings - dollars per hour
awkly_hours:	Average weekly hours - hours per week
atotalwkly_earnings:	Average total weekly earnings - dollars per week
rt_merch:	Employees in retail trade: general merchandise - thousands of people
rt_food_bev:	Employees in retail trade: food and beverages - thousands of people
goods_producing:	Employees in goods producing - thousands of people
unemployed:	Unemployed people in MSA - individual people
wholesale_trade:	Employees in wholesale trade - thousands of people
retail_trade:	Employees in retail trade - thousands of people
service_providing:	Employees in service providing - thousands of people

Variables to be predicted:



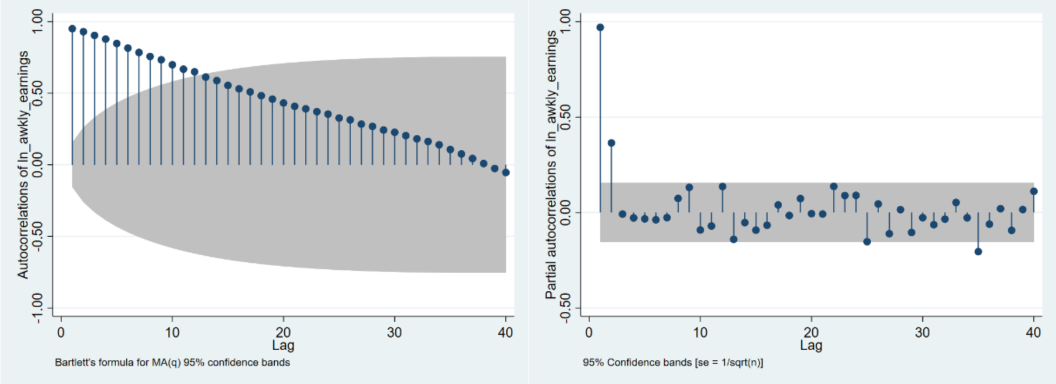


Variables	Dickey-Fuller p-value
ln_employment	0.7793227
ln_awkly_earnings	0.5734964
ln_ahrly_earnings	0.649941
ln_awkly_hours	0.0044248
ln_atotalwkly_earnings	0.9533231

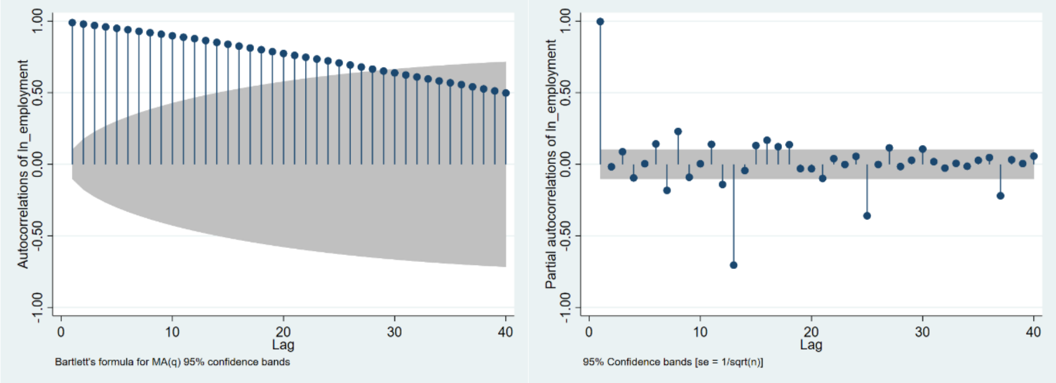
The results of the Dickey-Fuller tests show that all variables but *ln_awkly_hours* are an I(1) process. Looking at the autocorrelograms and partial-autocorrelograms will be the determining factor if the variables should be differenced, however.

Autocorrelograms and Partial-Autocorrelograms:

ln_awkly_earnings



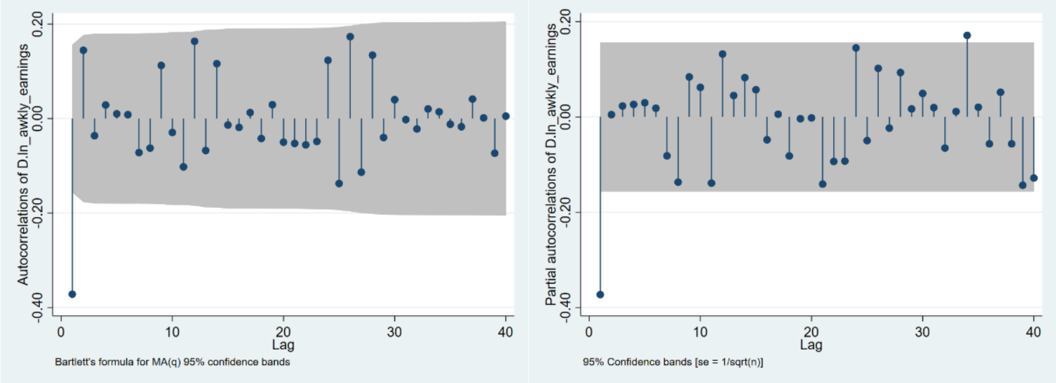
ln_employment



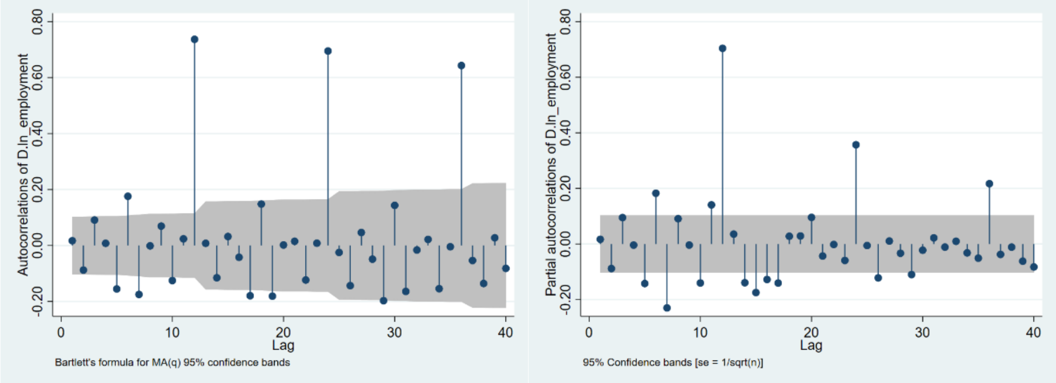
It is clear from the graphs above that *ln_awkly_earnings* and *ln_employment* should be differenced because of the strong first order auto-regressive relationship.

The differenced autocorrelograms and partial-autocorrelograms are below.

ln_awkly_earnings



ln_employment



Model Estimation & Selection

Average Weekly Earnings

To find the best model for average weekly employment an initial model where the only predictor variables are lags one through twelve of *ln_awkly_earnings* and eleven monthly indicator variables. This initial model, with a window of 60 months, is shown below.

```
Regress d.ln_awkly_earnings l(1/12)d.ln_awkly_earnings ///
m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12
```

Table 1	
RWRMSE:	0.02273833
Window:	60

In order to find candidates for the best possible model for forecasting one period ahead there was extensive use of the GSREG command. This command runs through all combinations of the given variables and ranks the resulting model based on the AIC, BIC and out-of-sample RMSE values.

Example GSREG command:

```
gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l2d_ln_employment l12d_ln_employment l24d_ln_employment ///
l2d_ln_awkly_hours l12d_ln_awkly_hours ///
l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l2d_ln_rt_merch l12d_ln_rt_merch ///
l2d_ln_rt_food_bev l12d_ln_rt_food_bev, ///
ncomb(1,8) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_1) replace
```

Multiple GSREG commands were run trying different combinations of the variables explained earlier on in this report. The results from this command were ranked according to a weighted scale with 40% based on out-of-sample RMSE, 30% on AIC value and 30% on the BIC value. Intuition was then used to look through the top choices and select the most promising models. The chosen models and their corresponding RMSE, AIC and BIC can be found below.

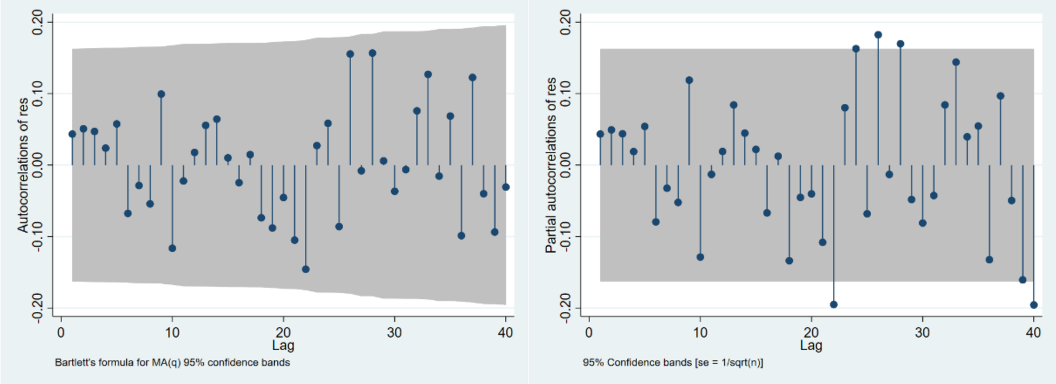
Table 2 – Average Weekly Earnings Models

Model #	Model Predictors	Out-of-Sample RMSE	AIC	BIC
1	l1d.ln_awkly_earnings l12d.ln_ahrly_earnings l12d.ln_rt_merch	0.0035645	-633.4619	-591.525
2	l1d.ln_awkly_earnings l12d.ln_ahrly_earnings	0.0038157	-633.6818	-594.5408
3	l12d._ln_ahrly_earnings l1d.ln_atotalwkly_earnings l12d.ln_rt_merch	0.0036476	-632.8224	-590.8856
4	l1d.ln_awkly_earnings l12d.ln_ahrly_earnings l12d.ln_rt_merch l1d.ln_goods_producing	0.0036373	-632.2894	-587.5567
5	l1d.ln_awkly_earnings l2d.ln_awkly_earnings l12d.ln_rt_merch	0.0039816	-633.1087	-591.1719

In order to find the best of the 5 models listed in Table 2 each model was run through a rolling window program. The results of the rolling window program are shown below.

Table 3 – Earnings Rolling Window Results		
Model #	RWRMSE	Window Length
1	0.02147014	60
2	0.01971482	88
3	0.01997173	60
4	0.02009816	116
5	0.02144576	76

The model with the best rolling window RMSE is Model 2. The autocorrelogram and partial-autocorrelogram of the residuals of Model 2 are located below.



Along with having the lowest RWRMSE (rolling window RMSE), Model 2 also appears to be dynamically complete. Overall, Model 2 seems to be the best suited for forecasting average weekly earnings for March of 2020.

Employment

Like finding the best model for average weekly earnings, a similar process will be used for employment. It will begin my estimating an initial model using the first twelve lags of *ln_employment* and followed up by estimating multiple GSREG commands to search through many possible combinations of variables.

```
Regress d.ln_employment l(1/12)d.ln_employment /// m2 m3 m4
m5 m6 m7 m8 m9 m10 m11 m12
```

Table 4	
RWRMSE:	0.00619164
Window:	60

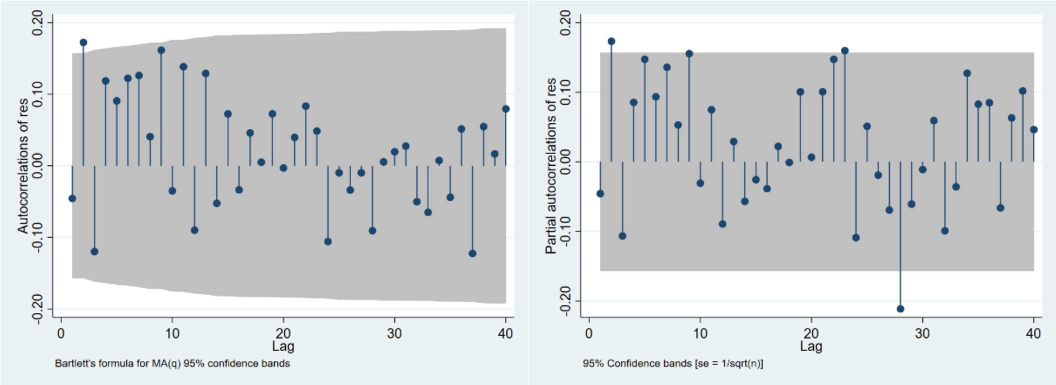
The following table (Table 5) contains the top models collected from the GSREG search. Intuition was used to pick the best five models from the search, just like how the models for average weekly earnings was found.

Table 5 – Employment Models				
Model #	Model Predictors	Out-of-Sample RMSE	AIC	BIC
6	l12d.ln_employment l1d.ln_awkly_hours l12d.ln_rt_merch	0.0014127	-932.224	-890.2872
7	l12d.ln_employment l1d.ln_awkly_hours l12d.ln_rt_merch l1d.ln_wholesale_trade	0.0013664	-932.0991	-887.3665
8	l12d.ln_employment l1d.ln_ahrly_earnings l1d.ln_atotalwkly_earnings l1d.ln_wholesale_trade	0.0012865	-929.1284	-884.3958
9	l12d.ln_employment l1d.ln_awkly_hours	0.0013158	-926.801	-887.8278
10	l1d.ln_employment l3d.ln_employment l12d.ln_employment l1d.ln_awkly_hours l12d.rt_merch	0.001656	-936.3562	-888.8278

The five models in Table 5 were all put through a rolling window program to find their rolling window RMSE for different window lengths. The table of the results is below.

Table 6 – Employment Rolling Window Results		
Model #	RWRMSE	Window Length
6	0.00637292	100
7	0.0064171	64
8	0.00638539	64
9	0.00626973	64
10	0.00612647	68

Model 10 has the lowest RWRMSE and is the best candidate for forecasting employment. The autocorrelogram and partial-autocorrelogram of the residuals of the model can be found below.



The selected model (Model 10) looks to be nearly dynamically complete. More important, though, is the low RWRMSE. Model 10 will be the model used to forecast the employment for March of 2020.

Forecast Models

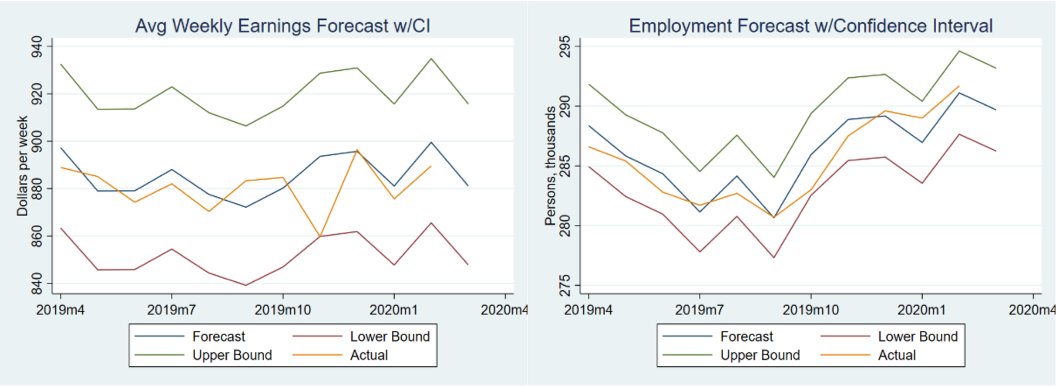
The final models being used to forecast average weekly earnings and employment for the North Port-Sarasota-Bradenton MSA are:

Average weekly earnings:

```
reg d.ln_employment l1d_ln_employment l3d_ln_employment ///
l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_merch ///
m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m6,2020m2)
```

Employment:

```
reg d_ln_awkly_earnings l1d_ln_awkly_earnings ///
l12d_ln_ahrly_earnings ///
m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m6,2020m2)
```

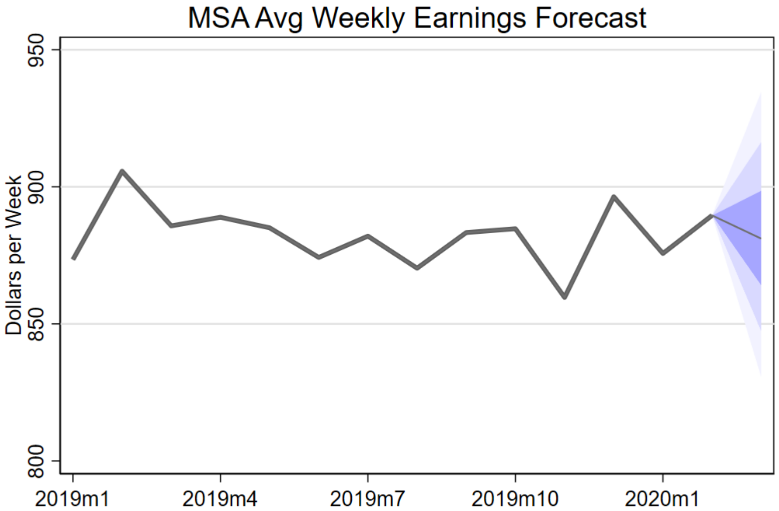


It can be seen that each model fits the data quite well with the only time the actual value going outside of the 95% confidence interval in November of 2019 for average weekly earnings.

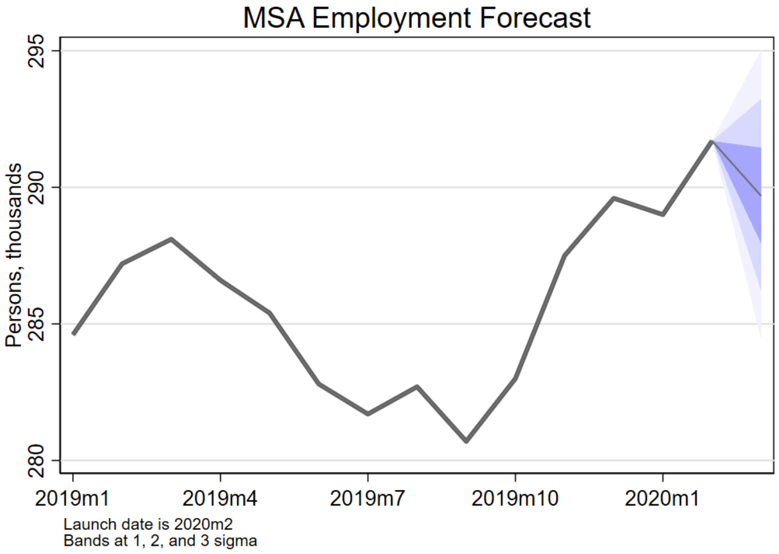
Final Results

Each model was run over the appropriate window and calculated a point forecast for March of 2020. Additionally, a 95% confidence interval was created for each forecast. The point forecasts and upper/lower bounds of the confidence interval are in the following Table 7.

Table 7 - Point Forecasts and Upper/Lower Bounds for March of 2020				
	Lower Bound	Point Forecast	Upper Bound	RWRMSE
Average Weekly Earnings	847.7749	881.092	915.7184	0.01966677
Employment	286.234	289.6809	293.1693	0.00610728



Launch date is 2020m2
Bands at 1, 2, and 3 sigma



Conclusion

According to the forecasts above there is an expected decrease in employment and average weekly earnings for the North Port-Sarasota-Bradenton MSA. This model does not consider any extraneous events (such as a worldwide pandemic) and has a limited amount of data available for prediction purposes (due to some data only being collected since 2007). Nonetheless, these models have shown to have good fit within the sample and the methods used (such as rolling window program) give confidence that they have good out of sample fit as well.

It is also important to note that normality was assumed in order to calculate the confidence intervals for these forecasts. This was done because of the small sample of data, meaning that the tails of the distributions only have a small number of data points causing an empirical approach to be severely affected by outliers.

Index:

Appendix A: Do-File

```
//Joshua Cantera
//Final Project Spring 2020
//Time Series and Forecasting
clear
cd "C:\Users\Josh\Desktop\School\3rd\TimeSeries\Project"
*log using
"C:\Users\Josh\Desktop\School\3rd\TimeSeries\Project\ProjectLog.smcl", replace
import delimited "Data\Project2020_3_Monthly.txt"


*Thousands of persons (Variable to predict)
rename smu12358400500000001_20200327 employment
*Hours per week
rename smu12358400500000002_20200327 awkly_hours
*Dollars per hour
rename smu12358400500000003_20200327 ahrly_earnings
*Dollars per week (Variable to predict)
rename smu12358400500000011_20200327 awkly_earnings
*Dollars times employed people per week
gen atotalwkly_earnings = awkly_earnings * employment

rename laumt123584000000004* unemployed
rename laumt123584000000005* employed
rename sara212lfn* civilian_lf
rename smu12358400600000001* goods_producing
rename smu12358400700000001* service_providing
rename smu12358400800000001* private_service_providing
rename smu12358404100000001* wholesale_trade
rename smu12358404200000001* retail_trade
rename smu12358404244500001* rt_food_bev
rename smu12358404245200001* rt_merch
rename smu12358404300000001* trans_util_warehouse
rename smu12358406562100001* hc_ambulance
rename smu12358406562200001* hc_hospital
rename smu12358409091000001* fed_gov
rename smu12358409092000001* state_gov
rename smu12358409093000001* local_gov

gen ln_ahrly_earnings = ln(ahrly_earnings)
gen ln_awkly_earnings = ln(awkly_earnings)
gen ln_awkly_hours = ln(awkly_hours)
gen ln_employment = ln(employment)
gen ln_atotalwkly_earnings = ln(atotalwkly_earnings)

rename observation_date datestring
gen datec=date(datestring, "YMD")
gen date=mofd(datec)
format date %tm
tsset date

tsappend, add(1)

gen month = month(datec)

gen m2 = 0
replace m2 = 1 if month == 2
gen m3 = 0
```

```
replace m3 = 1 if month == 3
gen m4 = 0
replace m4 = 1 if month == 4
gen m5 = 0
replace m5 = 1 if month == 5
gen m6 = 0
replace m6 = 1 if month == 6
gen m7 = 0
replace m7 = 1 if month == 7
gen m8 = 0
replace m8 = 1 if month == 8
gen m9 = 0
replace m9 = 1 if month == 9
gen m10 = 0
replace m10 = 1 if month == 10
gen m11 = 0
replace m11 = 1 if month == 11
gen m12 = 0
replace m12 = 1 if month == 12


cd Graphs

summ ln_employment, detail
twoway (tsline ln_employment, lcolor(ebblue)) (lfit ln_employment date,
lcolor(black)) if tin(2007m1,2020m2), ///
title("ln Employment in North Port (2007m1-2020m2)") ytitle("ln Employment
(people, thousands)") ///
legend(label(1 "ln Employment"))
graph export twoway_emp.png, replace
ac ln_employment
graph export ac_emp.png, replace
pac ln_employment
graph export pac_emp.png, replace
ac d.ln_employment
graph export ac_d_emp.png, replace
pac d.ln_employment
graph export pac_d_emp.png, replace
dfuller ln_employment
scalar df_1 = r(p)

summ ln_awkly_earnings, detail
twoway (tsline ln_awkly_earnings, lcolor(ebblue)) (lfit ln_awkly_earnings date,
lcolor(black)) if tin(2007m1,2020m2), ///
title("ln Avg. Weekly Earnings in North Port (2007m1-2020m2)") ytitle("ln
Dollars") ///
legend(label(1 "ln Avg. Weekly Earnings"))
graph export twoway_earnings.png, replace
ac ln_awkly_earnings
graph export ac_awkly_earnings.png, replace
pac ln_awkly_earnings
graph export pac_awkly_earnings.png, replace
ac d.ln_awkly_earnings
graph export ac_d_awkly_earnings.png, replace
pac d.ln_awkly_earnings
graph export pac_d_awkly_earnings.png, replace
dfuller ln_awkly_earnings
scalar df_2 = r(p)
```

```
summ ln_ahrly_earnings, detail
twoway (tsline ln_ahrly_earnings) if tin(2007m1,2020m1)
ac ln_ahrly_earnings
graph export ac_ahrly_earnings.png, replace
pac ln_ahrly_earnings
graph export pac_ahrly_earnings.png, replace
dfuller ln_ahrly_earnings
scalar df_3 = r(p)

summ ln_awkly_hours, detail
twoway (tsline ln_awkly_hours) if tin(2007m1,2020m1)
ac ln_awkly_hours
graph export ac_awkly_hours.png, replace
pac ln_awkly_hours
graph export pac_awkly_hours.png, replace
dfuller ln_awkly_hours
scalar df_4 = r(p)

summ ln_atotalwkly_earnings, detail
twoway (tsline ln_atotalwkly_earnings) if tin(2007m1,2020m1)
ac ln_atotalwkly_earnings
graph export ac_atotalwkly_earnings.png, replace
pac ln_atotalwkly_earnings
graph export pac_atotalwkly_earnings.png, replace
dfuller ln_atotalwkly_earnings
scalar df_5 = r(p)

matrix m = (df_1\df_2\df_3\df_4\df_5)
matrix rownames m = ln_employment ln_awkly_earnings ln_ahrly_earnings
ln_awkly_hours ln_atotalwkly_earnings
matrix colnames m = "p-value"
matlist m, rowtitle(Variables) twidth(22)

*Creating variables for GSREG
gen d_ln_awkly_earnings = d.ln_awkly_earnings
gen l1d_ln_awkly_earnings = l1d.ln_awkly_earnings
gen l2d_ln_awkly_earnings = l2d.ln_awkly_earnings
gen l3d_ln_awkly_earnings = l3d.ln_awkly_earnings
gen l6d_ln_awkly_earnings = l6d.ln_awkly_earnings
gen l12d_ln_awkly_earnings = l12d.ln_awkly_earnings
gen l24d_ln_awkly_earnings = l24d.ln_awkly_earnings
gen l36d_ln_awkly_earnings = l36d.ln_awkly_earnings

gen d_ln_employment = d.ln_employment
gen l1d_ln_employment = l1d.ln_employment
gen l2d_ln_employment = l2d.ln_employment
gen l3d_ln_employment = l3d.ln_employment
gen l6d_ln_employment = l6d.ln_employment
gen l12d_ln_employment = l12d.ln_employment
gen l24d_ln_employment = l24d.ln_employment
gen l36d_ln_employment = l36d.ln_employment

gen l1d_ln_ahrly_earnings = l1d.ln_ahrly_earnings
gen l2d_ln_ahrly_earnings = l2d.ln_ahrly_earnings
gen l3d_ln_ahrly_earnings = l3d.ln_ahrly_earnings
gen l6d_ln_ahrly_earnings = l6d.ln_ahrly_earnings
gen l12d_ln_ahrly_earnings = l12d.ln_ahrly_earnings
gen l24d_ln_ahrly_earnings = l24d.ln_ahrly_earnings
gen l36d_ln_ahrly_earnings = l36d.ln_ahrly_earnings

gen l1d_ln_awkly_hours = l1d.ln_awkly_hours
gen l2d_ln_awkly_hours = l2d.ln_awkly_hours
gen l3d_ln_awkly_hours = l3d.ln_awkly_hours
gen l6d_ln_awkly_hours = l6d.ln_awkly_hours
gen l12d_ln_awkly_hours = l12d.ln_awkly_hours
gen l24d_ln_awkly_hours = l24d.ln_awkly_hours
gen l36d_ln_awkly_hours = l36d.ln_awkly_hours

gen l1d_ln_atotalwkly_earnings = l1d.ln_atotalwkly_earnings
gen l2d_ln_atotalwkly_earnings = l2d.ln_atotalwkly_earnings
gen l3d_ln_atotalwkly_earnings = l3d.ln_atotalwkly_earnings
gen l6d_ln_atotalwkly_earnings = l6d.ln_atotalwkly_earnings
gen l12d_ln_atotalwkly_earnings = l12d.ln_atotalwkly_earnings
gen l24d_ln_atotalwkly_earnings = l24d.ln_atotalwkly_earnings
gen l36d_ln_atotalwkly_earnings = l36d.ln_atotalwkly_earnings

gen ln_rt_merch = ln(rt_merch)
gen l1d_ln_rt_merch = l1d.ln_rt_merch
gen l2d_ln_rt_merch = l2d.ln_rt_merch
gen l3d_ln_rt_merch = l3d.ln_rt_merch
gen l6d_ln_rt_merch = l6d.ln_rt_merch
gen l12d_ln_rt_merch = l12d.ln_rt_merch
gen l24d_ln_rt_merch = l24d.ln_rt_merch
gen l36d_ln_rt_merch = l36d.ln_rt_merch

gen ln_rt_food_bev = ln(rt_food_bev)
gen l1d_ln_rt_food_bev = l1d.ln_rt_food_bev
gen l2d_ln_rt_food_bev = l2d.ln_rt_food_bev
gen l3d_ln_rt_food_bev = l3d.ln_rt_food_bev
gen l6d_ln_rt_food_bev = l6d.ln_rt_food_bev
gen l12d_ln_rt_food_bev = l12d.ln_rt_food_bev
gen l24d_ln_rt_food_bev = l24d.ln_rt_food_bev
gen l36d_ln_rt_food_bev = l36d.ln_rt_food_bev

gen ln_goods_producing = ln(goods_producing)
gen l1d_ln_goods_producing = l1d.ln_goods_producing
gen l2d_ln_goods_producing = l2d.ln_goods_producing
gen l3d_ln_goods_producing = l3d.ln_goods_producing
gen l6d_ln_goods_producing = l6d.ln_goods_producing
gen l12d_ln_goods_producing = l12d.ln_goods_producing
gen l24d_ln_goods_producing = l24d.ln_goods_producing
gen l36d_ln_goods_producing = l36d.ln_goods_producing

gen ln_unemployed = ln(unemployed)
gen l1d_ln_unemployed = l1d.ln_unemployed
gen l2d_ln_unemployed = l2d.ln_unemployed
gen l3d_ln_unemployed = l3d.ln_unemployed
gen l6d_ln_unemployed = l6d.ln_unemployed
gen l12d_ln_unemployed = l12d.ln_unemployed
gen l24d_ln_unemployed = l24d.ln_unemployed
gen l36d_ln_unemployed = l36d.ln_unemployed

gen ln_wholesale_trade = ln(wholesale_trade)
gen l1d_ln_wholesale_trade = l1d.ln_wholesale_trade
gen l2d_ln_wholesale_trade = l2d.ln_wholesale_trade
gen l3d_ln_wholesale_trade = l3d.ln_wholesale_trade
gen l6d_ln_wholesale_trade = l6d.ln_wholesale_trade
gen l12d_ln_wholesale_trade = l12d.ln_wholesale_trade
gen l24d_ln_wholesale_trade = l24d.ln_wholesale_trade
gen l36d_ln_wholesale_trade = l36d.ln_wholesale_trade

gen ln_retail_trade = ln(retail_trade)
gen l1d_ln_retail_trade = l1d.ln_retail_trade
gen l2d_ln_retail_trade = l2d.ln_retail_trade
gen l3d_ln_retail_trade = l3d.ln_retail_trade
gen l6d_ln_retail_trade = l6d.ln_retail_trade
gen l12d_ln_retail_trade = l12d.ln_retail_trade
gen l24d_ln_retail_trade = l24d.ln_retail_trade
gen l36d_ln_retail_trade = l36d.ln_retail_trade

gen ln_service_providing = ln(service_providing)
gen l1d_ln_service_providing = l1d.ln_service_providing
gen l2d_ln_service_providing = l2d.ln_service_providing
gen l3d_ln_service_providing = l3d.ln_service_providing
gen l6d_ln_service_providing = l6d.ln_service_providing
gen l12d_ln_service_providing = l12d.ln_service_providing
gen l24d_ln_service_providing = l24d.ln_service_providing
gen l36d_ln_service_providing = l36d.ln_service_providing

ac ln_rt_merch
graph export ac_rt_merch.png, replace
pac ln_rt_merch
graph export pac_rt_merch.png, replace

ac ln_rt_food_bev
graph export ac_rt_food_bev.png, replace
pac ln_rt_food_bev
graph export pac_rt_food_bev.png, replace

ac ln_goods_producing
graph export ac_goods_producing.png, replace
pac ln_goods_producing
graph export pac_goods_producing.png, replace

ac ln_retail_trade
graph export ac_retail_trade.png, replace
pac ln_retail_trade
graph export pac_retail_trade.png, replace

ac ln_wholesale_trade
graph export ac_wholesale_trade.png, replace
pac ln_wholesale_trade
graph export pac_wholesale_trade.png, replace

ac ln_service_providing
graph export ac_service_providing.png, replace
pac ln_service_providing
graph export pac_service_providing.png, replace

ac ln_unemployed
graph export ac_unemployed.png, replace
pac ln_unemployed
graph export pac_wunemployed.png, replace
```

```
*GSREG
/*
gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l2d_ln_employment l12d_ln_employment l24d_ln_employment ///
l2d_ln_awkly_hours l12d_ln_awkly_hours ///
l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l2d_ln_rt_merch l12d_ln_rt_merch ///
l2d_ln_rt_food_bev l12d_ln_rt_food_bev, ///
ncomb(1,8) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_1) replace
*/

/*
gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l12d_ln_awkly_earnings l24d_ln_awkly_earnings l36d_ln_awkly_earnings ///
l2d_ln_employment l12d_ln_employment l24d_ln_employment ///
l2d_ln_awkly_hours l12d_ln_awkly_hours ///
l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l2d_ln_rt_merch l12d_ln_rt_merch ///
l2d_ln_unemployed l6d_ln_unemployed l12d_ln_unemployed, ///
ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_3) replace
*/

/*
gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l12d_ln_awkly_earnings l24d_ln_awkly_earnings l36d_ln_awkly_earnings ///
l2d_ln_employment l12d_ln_employment l24d_ln_employment ///
l2d_ln_awkly_hours l12d_ln_awkly_hours ///
l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l2d_ln_rt_merch l12d_ln_rt_merch ///
l2d_ln_wholesale_trade l12d_ln_wholesale_trade ///
l2d_ln_unemployed l6d_ln_unemployed l12d_ln_unemployed, ///
ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_4) replace
*/
/*
gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l6d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings
l36d_ln_awkly_earnings ///
l1d_ln_employment l2d_ln_employment l6d_ln_employment l12d_ln_employment
l24d_ln_employment ///
l2d_ln_awkly_hours l6d_ln_awkly_hours l12d_ln_awkly_hours l24d_ln_awkly_hours
///
l2d_ln_ahrly_earnings l6d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
```

```
l2d_ln_rt_merch l12d_ln_rt_merch, ///
ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_5) replace
*/
/*
gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l6d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l1d_ln_employment l2d_ln_employment l3d_ln_employment l12d_ln_employment ///
l2d_ln_awkly_hours l12d_ln_awkly_hours ///
l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l2d_ln_rt_merch l2d_ln_rt_merch l12d_ln_rt_merch ///
l2d_ln_goods_producing l12d_ln_goods_producing, ///
ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_6) replace
*/
/*
gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l1d_ln_employment l2d_ln_employment l12d_ln_employment ///
l1d_ln_awkly_hours l12d_ln_awkly_hours ///
l1d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l1d_ln_rt_merch l2d_ln_rt_merch l12d_ln_rt_merch ///
l1d_ln_goods_producing l12d_ln_goods_producing, ///
ncomb(1,6) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_2) replace

gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l1d_ln_employment l2d_ln_employment l12d_ln_employment ///
l1d_ln_awkly_hours l12d_ln_awkly_hours ///
l1d_ln_ahrly_earnings l12d_ln_ahrly_earnings l24d_ln_ahrly_earnings ///
l1d_ln_rt_merch l2d_ln_rt_merch l12d_ln_rt_merch ///
l1d_ln_wholesale_trade l12d_ln_wholesale_trade ///
l1d_ln_service_providing l12d_ln_service_providing ///
l1d_ln_retail_trade l12d_ln_retail_trade ///
l1d_ln_goods_producing l12d_ln_goods_producing, ///
ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_8) replace

gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l3d_ln_awkly_earnings ///
l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l1d_ln_employment l2d_ln_employment l12d_ln_employment ///
l1d_ln_awkly_hours l2d_ln_awkly_hours l12d_ln_awkly_hours ///
l1d_ln_ahrly_earnings l2d_ln_ahrly_earnings l6d_ln_ahrly_earnings
l12d_ln_ahrly_earnings l24d_ln_ahrly_earnings ///
l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l1d_ln_rt_merch l6d_ln_rt_merch l12d_ln_rt_merch ///
l1d_ln_goods_producing l12d_ln_goods_producing, ///
```

```
ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_7) replace
*/
 /*
gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings ///
l3d_ln_awkly_earnings l6d_ln_awkly_earnings l12d_ln_awkly_earnings ///
l24d_ln_awkly_earnings l1d_ln_ahrly_earnings l2d_ln_ahrly_earnings ///
l6d_ln_ahrly_earnings l12d_ln_ahrly_earnings l24d_ln_ahrly_earnings, ///
ncomb(1,4) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) ///
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_final) replace
*/

summ date if tin(2007m1,2020m2)
*564-721
/*
*Rolling window program Initial
scalar drop _all
quietly forval w=12(4)180 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/721 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d.ln_awkly_earnings l(1/12)d.ln_awkly_earnings ///
              m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
              if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
```

```
*End of rolling window program

*Rolling window program gsreg_2-1
scalar drop _all
quietly forval w=12(4)120 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/721 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings
l12d_ln_rt_merch ///
             m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_2-2
scalar drop _all
quietly forval w=12(4)120 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
```

```
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings ///
              m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
              if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_2-4
scalar drop _all
quietly forval w=12(4)120 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/721 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_awkly_earnings l12d_ln_ahrly_earnings l1d_ln_atotalwkly_earnings
l12d_ln_rt_merch ///
              m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
```

```
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_2-20
scalar drop _all
quietly forval w=12(4)180 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings
l12d_ln_rt_merch l1d_ln_goods_producing ///
             m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
```

```
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_2-21
scalar drop _all
quietly forval w=12(4)120 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings
l12d_ln_ahrly_earnings ///
             m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_7-4
scalar drop _all
quietly forval w=12(4)180 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point
```

```
       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_awkly_hours
l12d_ln_atotalwkly_earnings l12d_ln_rt_merch ///
              m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
              if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*/

**********************End of awkly_earnings model selection
**Start model selection for total employment

/*
gsreg d_ln_employment l1d_ln_employment l2d_ln_employment l3d_ln_employment
l12d_ln_employment l24d_ln_employment ///
l1d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l1d_ln_awkly_hours l12d_ln_awkly_hours ///
l1d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l1d_ln_rt_merch l12d_ln_rt_merch ///
l1d_ln_unemployed l12d_ln_unemployed ///
l1d_ln_retail_trade l12d_ln_retail_trade ///
l1d_ln_wholesale_trade l12d_ln_wholesale_trade, ///
ncomb(1,6) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_21) replace

gsreg d_ln_employment l1d_ln_employment l2d_ln_employment l3d_ln_employment
l12d_ln_employment l24d_ln_employment ///
l2d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l2d_ln_awkly_hours l12d_ln_awkly_hours ///
l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
```

```
l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l2d_ln_rt_merch l12d_ln_rt_merch ///
l2d_ln_rt_food_bev l12d_ln_rt_food_bev, ///
ncomb(1,7) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_22) replace

gsreg d_ln_employment l1d_ln_employment l2d_ln_employment l12d_ln_employment
l24d_ln_employment ///
l1d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
l1d_ln_awkly_hours l12d_ln_awkly_hours ///
l1d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l1d_ln_rt_merch l12d_ln_rt_merch ///
l1d_ln_goods_producing l12d_ln_goods_producing ///
l1d_ln_retail_trade l12d_ln_retail_trade ///
l1d_ln_wholesale_trade l12d_ln_wholesale_trade, ///
ncomb(1,6) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_23) replace

gsreg d_ln_employment l1d_ln_employment l2d_ln_employment l12d_ln_employment
l24d_ln_employment ///
l1d_ln_awkly_earnings l6d_ln_awkly_earnings l12d_ln_awkly_earnings
l24d_ln_awkly_earnings ///
l1d_ln_awkly_hours l6d_ln_awkly_hours l12d_ln_awkly_hours ///
l1d_ln_ahrly_earnings l6d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
l1d_ln_rt_merch l6d_ln_rt_merch l12d_ln_rt_merch ///
l1d_ln_goods_producing l12d_ln_goods_producing, ///
ncomb(1,6) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
samesample ///
nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_24) replace
*/
/*
*Rolling window program Initial
scalar drop _all
quietly forval w=12(4)180 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d.ln_employment l(1/12)d.ln_employment ///
             m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_employment)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_21-1
scalar drop _all
quietly forval w=12(4)180 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_employment l12d_ln_employment l1d_ln_awkly_hours
l12d_ln_rt_merch ///
             m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_employment)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_21-3
scalar drop _all
quietly forval w=12(4)220 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_employment l12d_ln_employment l1d_ln_awkly_hours
l12d_ln_rt_merch l1d_ln_wholesale_trade ///
             m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_employment)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_21-22
scalar drop _all
quietly forval w=12(4)180 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
```

```
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_employment l12d_ln_employment l1d_ln_ahrly_earnings
l1d_ln_atotalwkly_earnings l1d_ln_wholesale_trade ///
              m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
              if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_employment)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_21-25
scalar drop _all
quietly forval w=12(4)180 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_employment l12d_ln_employment l1d_ln_awkly_hours ///
             m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_employment)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*Rolling window program gsreg_21-45
scalar drop _all
quietly forval w=12(4)180 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point

       forval t=580/720 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_employment l1d_ln_employment l3d_ln_employment
l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_merch ///
             m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq=(pred-d_ln_employment)^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
```

```
scalar RWminobs`w'=r(min) // in obs used in the window width
scalar RWmaxobs`w'=r(max) // max obs used in the window width
drop errsq pred nobs // clearing for the next loop
}
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

*/



regress d.ln_awkly_earnings l1d.ln_awkly_earnings l12d.ln_ahrly_earnings ///
m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12
predict res, residuals
pac res
graph export "earnings_pac.png", replace
ac res
graph export "earnings_ac.png", replace

drop res

reg d_ln_employment l1d_ln_employment l3d_ln_employment l12d_ln_employment ///
l1d_ln_awkly_hours l12d_ln_rt_merch m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12
predict res, residuals
pac res
graph export "employment_pac.png", replace
ac res
graph export "employment_ac.png", replace



*Rolling window program - just for w=68, employment
scalar drop _all
quietly forval w=68(4)68 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred2=. // out of sample prediction
gen nobs2=. // number of observations in the window for each forecast point

       forval t=580/722 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_employment l1d_ln_employment l3d_ln_employment ///
       l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_merch ///
       m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
```

```
              if date>=wstart & date<=wend // restricts the model to the window
       replace nobs=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred2=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq2=(pred2-d.ln_employment)^2 // generating squared errors
}
*End of rolling window program

summ nobs2 // checking all had a full window
*get error info for normal interval
summ errsq2
scalar rwrmse2=r(mean)^0.5
scalar list rwrmse2
*Forecast for employment
reg d.ln_employment l1d_ln_employment l3d_ln_employment ///
l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_merch ///
m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m6,2020m2)
predict temp if tin(2020m3,2020m3)
replace pred2=temp if tin(2020m3,2020m3)
drop temp
gen np_employment = exp(l.ln_employment+pred2+(rwrmse2^2)/2)
gen ubn2=exp(l.ln_employment+pred2+1.96*rwrmse2+(rwrmse2^2)/2)
gen lbn2=exp(l.ln_employment+pred2-1.96*rwrmse2+(rwrmse2^2)/2)
list date np_employment lbn2 ubn2 if tin(2020m3,2020m3)
tsline np_employment lbn2 ubn2 employment if tin(2019m4,2020m3)

*Fan chart - Employment (rwrmse2)
reg d.ln_employment l1d_ln_employment l3d_ln_employment ///
l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_merch ///
m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m6,2020m2)
predict fpd_employment
gen fp_employment=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment) if
date==tm(2020m3)

gen ub1=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment+rwrmse2) if
date==tm(2020m3)
gen lb1=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment-rwrmse2) if
date==tm(2020m3)

gen ub2=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment+2*rwrmse2) if
date==tm(2020m3)
gen lb2=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment-2*rwrmse2) if
date==tm(2020m3)

gen ub3=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment+3*rwrmse2) if
date==tm(2020m3)
gen lb3=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment-3*rwrmse2) if
date==tm(2020m3)

drop fpd_employment

gen y=employment if tin(2019m1,2020m2)
replace y=fp_employment if date>tm(2020m2)

gen yub1=employment if date==tm(2020m2)
replace yub1=ub1 if date>tm(2020m2)
```

```
gen ylb1=employment if date==tm(2020m2)
replace ylb1=lb1 if date>tm(2020m2)

gen yub2=employment if date==tm(2020m2)
replace yub2=ub2 if date>tm(2020m2)
gen ylb2=employment if date==tm(2020m2)
replace ylb2=lb2 if date>tm(2020m2)

gen yub3=employment if date==tm(2020m2)
replace yub3=ub3 if date>tm(2020m2)
gen ylb3=employment if date==tm(2020m2)
replace ylb3=lb3 if date>tm(2020m2)

cd ..

twoway (tsrline yub3 yub2 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
       (tsrline yub2 yub1 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
       (tsrline yub1 y if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
       (tsrline y ylb1 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
       (tsrline ylb1 ylb2 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
       (tsrline ylb2 ylb3 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
       (tsline employment y if tin(2019m1,2020m3) , ///
       lcolor(gs6) lwidth(thick) ), scheme(s1mono) legend(off) ///
       title("MSA Employment Forecast") legend(off) ///
       xtitle("") ytitle("Persons, thousands") ylabel(,grid) ///
       note("Launch date is 2020m2" "Bands at 1, 2, and 3 sigma")
graph export "Emp_FanChart.png", replace

stop
*Rolling window program - just for w=88, awkly_earnings
scalar drop _all
quietly forval w=88(4)88 {
/* w=small(inc)large
small is the smallest window
inc is the window size increment
large is the largest window.
(large-small)/inc must be an interger */
gen pred10=. // out of sample prediction
gen nobs10=. // number of observations in the window for each forecast point

       forval t=580/722 {
       /* t=first/last
       first is the first date for which you want to make a forecast.
       first-1 is the end date of the earliest window used to fit the model.
       first-w, where w is the window width, is the date of the first
       observation used to fit the model in the earliest window.
       You must choose first so it is preceded by a full set of
    lags for the model with the longest lag length to be estimated.
       last is  the last observation to be forecast. */
       gen wstart=`t'-`w' // fit window start date
       gen wend=`t'-1 // fit window end date
       /* Enter the regression command immediately below.
       Leave the if statement intact to control the window  */
       reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings ///
       m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
             if date>=wstart & date<=wend // restricts the model to the window
       replace nobs10=e(N) if date==`t' // number of observations used
       predict ptemp // temporary predicted values
       replace pred10=ptemp if date==`t' // saving the single forecast value
       drop ptemp wstart wend // clear these to prepare for the next loop
       }
gen errsq10=(pred10-d.ln_awkly_earnings)^2 // generating squared errors
}
*End of rolling window program

summ nobs10 // checking all had a full window
*get error info for normal interval
summ errsq10
scalar rwrmse10=r(mean)^0.5
scalar list rwrmse10
*Forecast for awkly_earnings
reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings ///
m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2012m10,2020m2)
predict temp if tin(2020m3,2020m3)
replace pred10=temp if tin(2020m3,2020m3)
drop temp
gen np_awkly_earnings = exp(l.ln_awkly_earnings+pred10+(rwrmse10^2)/2)
gen ubn10=exp(l.ln_awkly_earnings+pred10+1.96*rwrmse10+(rwrmse10^2)/2)
gen lbn10=exp(l.ln_awkly_earnings+pred10-1.96*rwrmse10+(rwrmse10^2)/2)
list date np_awkly_earnings lbn10 ubn10 if tin(2020m3,2020m3)
tsline np_awkly_earnings lbn10 ubn10 awkly_earnings if tin(2019m4,2020m3)

drop ub1 lb1 ub2 lb2 ub3 lb3
*Fan chart - Avg Weekly Earnings (rwrmse10)
reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings ///
m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2012m10,2020m2)
predict fpd_awkly_earnings
gen
fp_awkly_earnings=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earning
s) if date==tm(2020m3)

gen
ub1=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings+rwrmse10) if
date==tm(2020m3)
gen lb1=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings-
rwrmse10) if date==tm(2020m3)

gen
ub2=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings+2*rwrmse10)
if date==tm(2020m3)
gen lb2=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings-
2*rwrmse10) if date==tm(2020m3)

gen
ub3=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings+3*rwrmse10)
if date==tm(2020m3)
gen lb3=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings-
3*rwrmse10) if date==tm(2020m3)

drop fpd_awkly_earnings
drop y yub1 yub2 ylb1 ylb2 yub3 ylb3
gen y=awkly_earnings if tin(2019m1,2020m2)
replace y=fp_awkly_earnings if date>tm(2020m2)

gen yub1=awkly_earnings if date==tm(2020m2)
replace yub1=ub1 if date>tm(2020m2)
gen ylb1=awkly_earnings if date==tm(2020m2)
replace ylb1=lb1 if date>tm(2020m2)

gen yub2=awkly_earnings if date==tm(2020m2)
replace yub2=ub2 if date>tm(2020m2)
gen ylb2=awkly_earnings if date==tm(2020m2)
replace ylb2=lb2 if date>tm(2020m2)

gen yub3=awkly_earnings if date==tm(2020m2)
replace yub3=ub3 if date>tm(2020m2)
gen ylb3=awkly_earnings if date==tm(2020m2)
replace ylb3=lb3 if date>tm(2020m2)

twoway (tsrline yub3 yub2 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
       (tsrline yub2 yub1 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
       (tsrline yub1 y if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
       (tsrline y ylb1 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
       (tsrline ylb1 ylb2 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
       (tsrline ylb2 ylb3 if tin(2020m2,2020m3), ///
       recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
       (tsline awkly_earnings y if tin(2019m1,2020m3) , ///
       lcolor(gs6) lwidth(thick) ), scheme(s1mono) legend(off) ///
       title("MSA Avg Weekly Earnings Forecast") legend(off) ///
       xtitle("") ytitle("Dollars per Week") ylabel(,grid) ///
       note("Launch date is 2020m2" "Bands at 1, 2, and 3 sigma")
graph export "awkly_earnings_FanChart.png", replace


*log close
```

Appendix B: Log-File

```
--------------------------------------------------------------------------------
      name:  <unnamed>
       log:  C:\Users\Josh\Desktop\School\3rd\TimeSeries\Project\ProjectLog.smc
> l
  log type:  smcl
 opened on:  16 Apr 2020, 22:50:15

. import delimited "Data\Project2020_3_Monthly.txt"
(21 vars, 362 obs)

. 
. 
. *Thousands of persons (Variable to predict)
. rename smu12358400500000001_20200327 employment

. *Hours per week
. rename smu12358400500000002_20200327 awkly_hours

. *Dollars per hour
. rename smu12358400500000003_20200327 ahrly_earnings

. *Dollars per week (Variable to predict)
. rename smu12358400500000011_20200327 awkly_earnings

. *Dollars times employed people per week
. gen atotalwkly_earnings = awkly_earnings * employment
(204 missing values generated)

. 
. rename laumt123584000000004* unemployed

. rename laumt123584000000005* employed

. rename sara212lfn* civilian_lf

. rename smu12358400600000001* goods_producing

. rename smu12358400700000001* service_providing

. rename smu12358400800000001* private_service_providing

. rename smu12358404100000001* wholesale_trade

. rename smu12358404200000001* retail_trade

. rename smu12358404244500001* rt_food_bev

. rename smu12358404245200001* rt_merch

. rename smu12358404300000001* trans_util_warehouse

. rename smu12358406562100001* hc_ambulance

. rename smu12358406562200001* hc_hospital

. rename smu12358409091000001* fed_gov

. rename smu12358409092000001* state_gov

. rename smu12358409093000001* local_gov

. 
. gen ln_ahrly_earnings = ln(ahrly_earnings)
(204 missing values generated)
```

```
. gen ln_awkly_earnings = ln(awkly_earnings)
(204 missing values generated)

. gen ln_awkly_hours = ln(awkly_hours)
(204 missing values generated)

. gen ln_employment = ln(employment)

. gen ln_atotalwkly_earnings = ln(atotalwkly_earnings)
(204 missing values generated)

. 
. rename observation_date datestring

. gen datec=date(datestring, "YMD")

. gen date=mofd(datec)

. format date %tm

. tsset date
        time variable:  date, 1990m1 to 2020m2
                delta:  1 month

. 
. tsappend, add(1)

. 
. gen month = month(datec)
(1 missing value generated)

. 
. gen m2 = 0

. replace m2 = 1 if month == 2
(31 real changes made)

. gen m3 = 0

. replace m3 = 1 if month == 3
(30 real changes made)

. gen m4 = 0

. replace m4 = 1 if month == 4
(30 real changes made)

. gen m5 = 0

. replace m5 = 1 if month == 5
(30 real changes made)

. gen m6 = 0

. replace m6 = 1 if month == 6
(30 real changes made)

. gen m7 = 0

. replace m7 = 1 if month == 7
(30 real changes made)

. gen m8 = 0

. replace m8 = 1 if month == 8
```

```
(30 real changes made)

. gen m9 = 0

. replace m9 = 1 if month == 9
(30 real changes made)

. gen m10 = 0

. replace m10 = 1 if month == 10
(30 real changes made)

. gen m11 = 0

. replace m11 = 1 if month == 11
(30 real changes made)

. gen m12 = 0

. replace m12 = 1 if month == 12
(30 real changes made)

.
.
. cd Graphs
C:\Users\Josh\Desktop\School\3rd\TimeSeries\Project\Graphs

.
. summ ln_employment, detail

                        ln_employment
-------------------------------------------------------------
      Percentiles      Smallest
 1%     4.962145       4.957938
 5%      4.99315       4.957938
10%     5.022564       4.960043       Obs                 362
25%     5.259576       4.962145       Sum of Wgt.         362

50%     5.378283                      Mean           5.356701
                        Largest       Std. Dev.      .1941859
75%     5.506144       5.663308
90%      5.60617       5.666427       Variance       .0377082
95%     5.637999       5.668501       Skewness       -.513883
99%     5.663308       5.675726       Kurtosis        2.38252

. twoway (tsline ln_employment, lcolor(ebblue)) (lfit ln_employment date, lcolo
> r(black)) if tin(2007m1,2020m2), ///
> title("ln Employment in North Port (2007m1-2020m2)") ytitle("ln Employment (p
> eople, thousands)") ///
> legend(label(1 "ln Employment"))

. graph export twoway_emp.png, replace
(file twoway_emp.png written in PNG format)

. ac ln_employment

. graph export ac_emp.png, replace
(file ac_emp.png written in PNG format)

. pac ln_employment

. graph export pac_emp.png, replace
(file pac_emp.png written in PNG format)

. ac d.ln_employment
. graph export ac_d_emp.png, replace
(file ac_d_emp.png written in PNG format)

. pac d.ln_employment

. graph export pac_d_emp.png, replace
(file pac_d_emp.png written in PNG format)

. dfuller ln_employment

Dickey-Fuller test for unit root                   Number of obs   =       361

                               ---------- Interpolated Dickey-Fuller ---------
                  Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value             Value
------------------------------------------------------------------------------
 Z(t)             -0.926            -3.451            -2.876            -2.570
------------------------------------------------------------------------------
MacKinnon approximate p-value for Z(t) = 0.7793

. scalar df_1 = r(p)

. 
. summ ln_awkly_earnings, detail

                      ln_awkly_earnings
-------------------------------------------------------------
      Percentiles      Smallest
 1%      6.49676       6.494767
 5%      6.52711        6.49676
10%     6.551623       6.497288       Obs                 158
25%     6.580389       6.508277       Sum of Wgt.         158

50%      6.64032                      Mean           6.649282
                        Largest       Std. Dev.      .0804675
75%     6.725106        6.79075
90%     6.768861       6.791671       Variance        .006475
95%     6.785655       6.798365       Skewness        .189171
99%     6.798365       6.808708       Kurtosis       2.012982

. twoway (tsline ln_awkly_earnings, lcolor(ebblue)) (lfit ln_awkly_earnings dat
> e, lcolor(black)) if tin(2007m1,2020m2), ///
> title("ln Avg. Weekly Earnings in North Port (2007m1-2020m2)") ytitle("ln Dol
> lars") ///
> legend(label(1 "ln Avg. Weekly Earnings"))

. graph export twoway_earnings.png, replace
(file twoway_earnings.png written in PNG format)

. ac ln_awkly_earnings

. graph export ac_awkly_earnings.png, replace
(file ac_awkly_earnings.png written in PNG format)

. pac ln_awkly_earnings

. graph export pac_awkly_earnings.png, replace
(file pac_awkly_earnings.png written in PNG format)

. ac d.ln_awkly_earnings

. graph export ac_d_awkly_earnings.png, replace
(file ac_d_awkly_earnings.png written in PNG format)

. pac d.ln_awkly_earnings
. graph export pac_d_awkly_earnings.png, replace
(file pac_d_awkly_earnings.png written in PNG format)

. dfuller ln_awkly_earnings

Dickey-Fuller test for unit root                   Number of obs   =       157

                               ---------- Interpolated Dickey-Fuller ---------
                  Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value             Value
------------------------------------------------------------------------------
 Z(t)             -1.418            -3.491            -2.886            -2.576
------------------------------------------------------------------------------
MacKinnon approximate p-value for Z(t) = 0.5735

. scalar df_2 = r(p)

. 
. summ ln_ahrly_earnings, detail

                      ln_ahrly_earnings
-------------------------------------------------------------
      Percentiles      Smallest
 1%     2.950735       2.933325
 5%     2.971952       2.950735
10%     2.996732       2.953868       Obs                 158
25%     3.033028       2.955431       Sum of Wgt.         158

50%     3.066424                      Mean           3.093731
                        Largest       Std. Dev.      .0853631
75%     3.181382       3.244933
90%     3.220874       3.247658       Variance       .0072869
95%     3.237501       3.252697       Skewness        .390056
99%     3.252697       3.266141       Kurtosis       1.902749

. twoway (tsline ln_ahrly_earnings) if tin(2007m1,2020m1)

. ac ln_ahrly_earnings

. graph export ac_ahrly_earnings.png, replace
(file ac_ahrly_earnings.png written in PNG format)

. pac ln_ahrly_earnings

. graph export pac_ahrly_earnings.png, replace
(file pac_ahrly_earnings.png written in PNG format)

. dfuller ln_ahrly_earnings

Dickey-Fuller test for unit root                   Number of obs   =       157

                               ---------- Interpolated Dickey-Fuller ---------
                  Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value             Value
------------------------------------------------------------------------------
 Z(t)             -1.254            -3.491            -2.886            -2.576
------------------------------------------------------------------------------
MacKinnon approximate p-value for Z(t) = 0.6499

. scalar df_3 = r(p)

. 
. summ ln_awkly_hours, detail

                       ln_awkly_hours
-------------------------------------------------------------
```

```
      Percentiles      Smallest
 1%      3.50255       3.462606
 5%     3.520461        3.50255
10%     3.523415       3.505557       Obs                 158
25%     3.538057       3.508556       Sum of Wgt.         158

50%      3.54674                      Mean           3.555551
                        Largest       Std. Dev.      .0287259
75%     3.577948       3.616309
90%     3.597312       3.616309       Variance       .0008252
95%     3.608212       3.624341       Skewness       .3801169
99%     3.624341        3.62966       Kurtosis       2.949013

. twoway (tsline ln_awkly_hours) if tin(2007m1,2020m1)

. ac ln_awkly_hours

. graph export ac_awkly_hours.png, replace
(file ac_awkly_hours.png written in PNG format)

. pac ln_awkly_hours

. graph export pac_awkly_hours.png, replace
(file pac_awkly_hours.png written in PNG format)

. dfuller ln_awkly_hours

Dickey-Fuller test for unit root                   Number of obs   =       157

                               ---------- Interpolated Dickey-Fuller ---------
                  Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value             Value
------------------------------------------------------------------------------
 Z(t)             -3.679            -3.491            -2.886            -2.576
------------------------------------------------------------------------------
MacKinnon approximate p-value for Z(t) = 0.0044

. scalar df_4 = r(p)

.
. summ ln_atotalwkly_earnings, detail

                   ln_atotalwkly_earnings
-------------------------------------------------------------
      Percentiles      Smallest
 1%      11.8891       11.88679
 5%     11.91958        11.8891
10%      11.9525       11.89876       Obs                 158
25%     11.99015       11.90689       Sum of Wgt.         158

50%     12.07709                      Mean           12.14196
                        Largest       Std. Dev.      .1755604
75%     12.31874       12.44975
90%     12.41325       12.46648       Variance       .0308215
95%     12.43955       12.46687       Skewness        .455638
99%     12.46687       12.46889       Kurtosis       1.744934

. twoway (tsline ln_atotalwkly_earnings) if tin(2007m1,2020m1)

. ac ln_atotalwkly_earnings

. graph export ac_atotalwkly_earnings.png, replace
(file ac_atotalwkly_earnings.png written in PNG format)

. pac ln_atotalwkly_earnings
. graph export pac_atotalwkly_earnings.png, replace
(file pac_atotalwkly_earnings.png written in PNG format)

. dfuller ln_atotalwkly_earnings

Dickey-Fuller test for unit root                   Number of obs   =       157

                               ---------- Interpolated Dickey-Fuller ---------
                  Test         1% Critical       5% Critical      10% Critical
               Statistic           Value             Value             Value
------------------------------------------------------------------------------
 Z(t)             -0.060            -3.491            -2.886            -2.576
------------------------------------------------------------------------------
MacKinnon approximate p-value for Z(t) = 0.9533

. scalar df_5 = r(p)

.
. matrix m = (df_1\df_2\df_3\df_4\df_5)

. matrix rownames m = ln_employment ln_awkly_earnings ln_ahrly_earnings ln_awkl
> y_hours ln_atotalwkly_earnings

. matrix colnames m = "p-value"

. matlist m, rowtitle(Variables) twidth(22)

             Variables |   p-value
-----------------------+-----------
         ln_employment |  .7793227
     ln_awkly_earnings |  .5734964
     ln_ahrly_earnings |   .649941
        ln_awkly_hours |  .0044248
ln_atotalwkly_earnings |  .9533231

.
. *Creating variables for GSREG
. gen d_ln_awkly_earnings = d.ln_awkly_earnings
(206 missing values generated)

. gen l1d_ln_awkly_earnings = l1d.ln_awkly_earnings
(206 missing values generated)

. gen l2d_ln_awkly_earnings = l2d.ln_awkly_earnings
(207 missing values generated)

. gen l3d_ln_awkly_earnings = l3d.ln_awkly_earnings
(208 missing values generated)

. gen l6d_ln_awkly_earnings = l6d.ln_awkly_earnings
(211 missing values generated)

. gen l12d_ln_awkly_earnings = l12d.ln_awkly_earnings
(217 missing values generated)

. gen l24d_ln_awkly_earnings = l24d.ln_awkly_earnings
(229 missing values generated)

. gen l36d_ln_awkly_earnings = l36d.ln_awkly_earnings
(241 missing values generated)

.
. gen d_ln_employment = d.ln_employment
(2 missing values generated)

. gen l1d_ln_employment = l1d.ln_employment
```

```
(2 missing values generated)

. gen l2d_ln_employment = l2d.ln_employment
(3 missing values generated)

. gen l3d_ln_employment = l3d.ln_employment
(4 missing values generated)

. gen l6d_ln_employment = l6d.ln_employment
(7 missing values generated)

. gen l12d_ln_employment = l12d.ln_employment
(13 missing values generated)

. gen l24d_ln_employment = l24d.ln_employment
(25 missing values generated)

. gen l36d_ln_employment = l36d.ln_employment
(37 missing values generated)

.
. gen l1d_ln_ahrly_earnings = l1d.ln_ahrly_earnings
(206 missing values generated)

. gen l2d_ln_ahrly_earnings = l2d.ln_ahrly_earnings
(207 missing values generated)

. gen l3d_ln_ahrly_earnings = l3d.ln_ahrly_earnings
(208 missing values generated)

. gen l6d_ln_ahrly_earnings = l6d.ln_ahrly_earnings
(211 missing values generated)

. gen l12d_ln_ahrly_earnings = l12d.ln_ahrly_earnings
(217 missing values generated)

. gen l24d_ln_ahrly_earnings = l24d.ln_ahrly_earnings
(229 missing values generated)

. gen l36d_ln_ahrly_earnings = l36d.ln_ahrly_earnings
(241 missing values generated)

.
. gen l1d_ln_awkly_hours = l1d.ln_awkly_hours
(206 missing values generated)

. gen l2d_ln_awkly_hours = l2d.ln_awkly_hours
(207 missing values generated)

. gen l3d_ln_awkly_hours = l3d.ln_awkly_hours
(208 missing values generated)

. gen l6d_ln_awkly_hours = l6d.ln_awkly_hours
(211 missing values generated)

. gen l12d_ln_awkly_hours = l12d.ln_awkly_hours
(217 missing values generated)

. gen l24d_ln_awkly_hours = l24d.ln_awkly_hours
(229 missing values generated)

. gen l36d_ln_awkly_hours = l36d.ln_awkly_hours
(241 missing values generated)

.
. gen l1d_ln_atotalwkly_earnings = l1d.ln_atotalwkly_earnings
```

```
(206 missing values generated)

. gen l2d_ln_atotalwkly_earnings = l2d.ln_atotalwkly_earnings
(207 missing values generated)

. gen l3d_ln_atotalwkly_earnings = l3d.ln_atotalwkly_earnings
(208 missing values generated)

. gen l6d_ln_atotalwkly_earnings = l6d.ln_atotalwkly_earnings
(211 missing values generated)

. gen l12d_ln_atotalwkly_earnings = l12d.ln_atotalwkly_earnings
(217 missing values generated)

. gen l24d_ln_atotalwkly_earnings = l24d.ln_atotalwkly_earnings
(229 missing values generated)

. gen l36d_ln_atotalwkly_earnings = l36d.ln_atotalwkly_earnings
(241 missing values generated)

.
. gen ln_rt_merch = ln(rt_merch)
(1 missing value generated)

. gen l1d_ln_rt_merch = l1d.ln_rt_merch
(2 missing values generated)

. gen l2d_ln_rt_merch = l2d.ln_rt_merch
(3 missing values generated)

. gen l3d_ln_rt_merch = l3d.ln_rt_merch
(4 missing values generated)

. gen l6d_ln_rt_merch = l6d.ln_rt_merch
(7 missing values generated)

. gen l12d_ln_rt_merch = l12d.ln_rt_merch
(13 missing values generated)

. gen l24d_ln_rt_merch = l24d.ln_rt_merch
(25 missing values generated)

. gen l36d_ln_rt_merch = l36d.ln_rt_merch
(37 missing values generated)

.
. gen ln_rt_food_bev = ln(rt_food_bev)
(1 missing value generated)

. gen l1d_ln_rt_food_bev = l1d.ln_rt_food_bev
(2 missing values generated)

. gen l2d_ln_rt_food_bev = l2d.ln_rt_food_bev
(3 missing values generated)

. gen l3d_ln_rt_food_bev = l3d.ln_rt_food_bev
(4 missing values generated)

. gen l6d_ln_rt_food_bev = l6d.ln_rt_food_bev
(7 missing values generated)

. gen l12d_ln_rt_food_bev = l12d.ln_rt_food_bev
(13 missing values generated)

. gen l24d_ln_rt_food_bev = l24d.ln_rt_food_bev
(25 missing values generated)
```

```
. gen l36d_ln_rt_food_bev = l36d.ln_rt_food_bev
(37 missing values generated)

.
. gen ln_goods_producing = ln(goods_producing)
(1 missing value generated)

. gen l1d_ln_goods_producing = l1d.ln_goods_producing
(2 missing values generated)

. gen l2d_ln_goods_producing = l2d.ln_goods_producing
(3 missing values generated)

. gen l3d_ln_goods_producing = l3d.ln_goods_producing
(4 missing values generated)

. gen l6d_ln_goods_producing = l6d.ln_goods_producing
(7 missing values generated)

. gen l12d_ln_goods_producing = l12d.ln_goods_producing
(13 missing values generated)

. gen l24d_ln_goods_producing = l24d.ln_goods_producing
(25 missing values generated)

. gen l36d_ln_goods_producing = l36d.ln_goods_producing
(37 missing values generated)

.
. gen ln_unemployed = ln(unemployed)
(1 missing value generated)

. gen l1d_ln_unemployed = l1d.ln_unemployed
(2 missing values generated)

. gen l2d_ln_unemployed = l2d.ln_unemployed
(3 missing values generated)

. gen l3d_ln_unemployed = l3d.ln_unemployed
(4 missing values generated)

. gen l6d_ln_unemployed = l6d.ln_unemployed
(7 missing values generated)

. gen l12d_ln_unemployed = l12d.ln_unemployed
(13 missing values generated)

. gen l24d_ln_unemployed = l24d.ln_unemployed
(25 missing values generated)

. gen l36d_ln_unemployed = l36d.ln_unemployed
(37 missing values generated)

.
. gen ln_wholesale_trade = ln(wholesale_trade)
(1 missing value generated)

. gen l1d_ln_wholesale_trade = l1d.ln_wholesale_trade
(2 missing values generated)

. gen l2d_ln_wholesale_trade = l2d.ln_wholesale_trade
(3 missing values generated)

. gen l3d_ln_wholesale_trade = l3d.ln_wholesale_trade
(4 missing values generated)
```

```
. gen l6d_ln_wholesale_trade = l6d.ln_wholesale_trade
(7 missing values generated)

. gen l12d_ln_wholesale_trade = l12d.ln_wholesale_trade
(13 missing values generated)

. gen l24d_ln_wholesale_trade = l24d.ln_wholesale_trade
(25 missing values generated)

. gen l36d_ln_wholesale_trade = l36d.ln_wholesale_trade
(37 missing values generated)

. 
. gen ln_retail_trade = ln(retail_trade)
(1 missing value generated)

. gen l1d_ln_retail_trade = l1d.ln_retail_trade
(2 missing values generated)

. gen l2d_ln_retail_trade = l2d.ln_retail_trade
(3 missing values generated)

. gen l3d_ln_retail_trade = l3d.ln_retail_trade
(4 missing values generated)

. gen l6d_ln_retail_trade = l6d.ln_retail_trade
(7 missing values generated)

. gen l12d_ln_retail_trade = l12d.ln_retail_trade
(13 missing values generated)

. gen l24d_ln_retail_trade = l24d.ln_retail_trade
(25 missing values generated)

. gen l36d_ln_retail_trade = l36d.ln_retail_trade
(37 missing values generated)

. 
. gen ln_service_providing = ln(service_providing)
(1 missing value generated)

. gen l1d_ln_service_providing = l1d.ln_service_providing
(2 missing values generated)

. gen l2d_ln_service_providing = l2d.ln_service_providing
(3 missing values generated)

. gen l3d_ln_service_providing = l3d.ln_service_providing
(4 missing values generated)

. gen l6d_ln_service_providing = l6d.ln_service_providing
(7 missing values generated)

. gen l12d_ln_service_providing = l12d.ln_service_providing
(13 missing values generated)

. gen l24d_ln_service_providing = l24d.ln_service_providing
(25 missing values generated)

. gen l36d_ln_service_providing = l36d.ln_service_providing
(37 missing values generated)

. 
. ac ln_rt_merch
. graph export ac_rt_merch.png, replace
(file ac_rt_merch.png written in PNG format)

. pac ln_rt_merch

. graph export pac_rt_merch.png, replace
(file pac_rt_merch.png written in PNG format)

.
. ac ln_rt_food_bev

. graph export ac_rt_food_bev.png, replace
(file ac_rt_food_bev.png written in PNG format)

. pac ln_rt_food_bev

. graph export pac_rt_food_bev.png, replace
(file pac_rt_food_bev.png written in PNG format)

.
. ac ln_goods_producing

. graph export ac_goods_producing.png, replace
(file ac_goods_producing.png written in PNG format)

. pac ln_goods_producing

. graph export pac_goods_producing.png, replace
(file pac_goods_producing.png written in PNG format)

.
. ac ln_retail_trade

. graph export ac_retail_trade.png, replace
(file ac_retail_trade.png written in PNG format)

. pac ln_retail_trade

. graph export pac_retail_trade.png, replace
(file pac_retail_trade.png written in PNG format)

.
. ac ln_wholesale_trade

. graph export ac_wholesale_trade.png, replace
(file ac_wholesale_trade.png written in PNG format)

. pac ln_wholesale_trade

. graph export pac_wholesale_trade.png, replace
(file pac_wholesale_trade.png written in PNG format)

.
. ac ln_service_providing

. graph export ac_service_providing.png, replace
(file ac_service_providing.png written in PNG format)

. pac ln_service_providing

. graph export pac_service_providing.png, replace
(file pac_service_providing.png written in PNG format)

.
. ac ln_unemployed
. graph export ac_unemployed.png, replace
(file ac_unemployed.png written in PNG format)

. pac ln_unemployed

. graph export pac_wunemployed.png, replace
(file pac_wunemployed.png written in PNG format)

.
. *GSREG
. /*
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l3d_ln_
> awkly_earnings ///
> l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
> l2d_ln_employment l12d_ln_employment l24d_ln_employment ///
> l2d_ln_awkly_hours l12d_ln_awkly_hours ///
> l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l2d_ln_rt_merch l12d_ln_rt_merch ///
> l2d_ln_rt_food_bev l12d_ln_rt_food_bev, ///
> ncomb(1,8) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_1) replace
> */
.
. /*
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l3d_ln_
> awkly_earnings ///
> l12d_ln_awkly_earnings l24d_ln_awkly_earnings l36d_ln_awkly_earnings ///
> l2d_ln_employment l12d_ln_employment l24d_ln_employment ///
> l2d_ln_awkly_hours l12d_ln_awkly_hours ///
> l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l2d_ln_rt_merch l12d_ln_rt_merch ///
> l2d_ln_unemployed l6d_ln_unemployed l12d_ln_unemployed, ///
> ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_3) replace
> */
.
. /*
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l3d_ln_
> awkly_earnings ///
> l12d_ln_awkly_earnings l24d_ln_awkly_earnings l36d_ln_awkly_earnings ///
> l2d_ln_employment l12d_ln_employment l24d_ln_employment ///
> l2d_ln_awkly_hours l12d_ln_awkly_hours ///
> l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l2d_ln_rt_merch l12d_ln_rt_merch ///
> l2d_ln_wholesale_trade l12d_ln_wholesale_trade ///
> l2d_ln_unemployed l6d_ln_unemployed l12d_ln_unemployed, ///
> ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_4) replace
> */
. /*
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l3d_ln_
> awkly_earnings ///
> l6d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings l36d_ln_a
> wkly_earnings ///
> l1d_ln_employment l2d_ln_employment l6d_ln_employment l12d_ln_employment l24d
> _ln_employment ///
> l2d_ln_awkly_hours l6d_ln_awkly_hours l12d_ln_awkly_hours l24d_ln_awkly_hours
>  ///
> l2d_ln_ahrly_earnings l6d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l2d_ln_rt_merch l12d_ln_rt_merch, ///
> ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_5) replace
> */
. /*
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l3d_ln_
> awkly_earnings ///
> l6d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
> l1d_ln_employment l2d_ln_employment l3d_ln_employment l12d_ln_employment ///
> l2d_ln_awkly_hours l12d_ln_awkly_hours ///
> l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l2d_ln_rt_merch l2d_ln_rt_merch l12d_ln_rt_merch ///
> l2d_ln_goods_producing l12d_ln_goods_producing, ///
> ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_6) replace
> */
. /*
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l3d_ln_
> awkly_earnings ///
> l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
> l1d_ln_employment l2d_ln_employment l12d_ln_employment ///
> l1d_ln_awkly_hours l12d_ln_awkly_hours ///
> l1d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l1d_ln_rt_merch l2d_ln_rt_merch l12d_ln_rt_merch ///
> l1d_ln_goods_producing l12d_ln_goods_producing, ///
> ncomb(1,6) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_2) replace
>
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l3d_ln_
> awkly_earnings ///
> l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
> l1d_ln_employment l2d_ln_employment l12d_ln_employment ///
> l1d_ln_awkly_hours l12d_ln_awkly_hours ///
> l1d_ln_ahrly_earnings l12d_ln_ahrly_earnings l24d_ln_ahrly_earnings ///
> l1d_ln_rt_merch l2d_ln_rt_merch l12d_ln_rt_merch ///
> l1d_ln_wholesale_trade l12d_ln_wholesale_trade ///
> l1d_ln_service_providing l12d_ln_service_providing ///
> l1d_ln_retail_trade l12d_ln_retail_trade ///
> l1d_ln_goods_producing l12d_ln_goods_producing, ///
> ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_8) replace
>
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l3d_ln_
> awkly_earnings ///
> l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
> l1d_ln_employment l2d_ln_employment l12d_ln_employment ///
> l1d_ln_awkly_hours l2d_ln_awkly_hours l12d_ln_awkly_hours ///
> l1d_ln_ahrly_earnings l2d_ln_ahrly_earnings l6d_ln_ahrly_earnings l12d_ln_ahr
> ly_earnings l24d_ln_ahrly_earnings ///
> l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l1d_ln_rt_merch l6d_ln_rt_merch l12d_ln_rt_merch ///
> l1d_ln_goods_producing l12d_ln_goods_producing, ///
> ncomb(1,5) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_7) replace
> */
.  /*
> gsreg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings ///
> l3d_ln_awkly_earnings l6d_ln_awkly_earnings l12d_ln_awkly_earnings ///
> l24d_ln_awkly_earnings l1d_ln_ahrly_earnings l2d_ln_ahrly_earnings ///
> l6d_ln_ahrly_earnings l12d_ln_ahrly_earnings l24d_ln_ahrly_earnings, ///
> ncomb(1,4) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) ///
> samesample ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_final) replace
> */
.
. summ date if tin(2007m1,2020m2)

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
        date |        158       642.5    45.75478        564        721

. *564-721
. /*
> *Rolling window program Initial
> scalar drop _all
> quietly forval w=12(4)180 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/721 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d.ln_awkly_earnings l(1/12)d.ln_awkly_earnings ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_2-1
> scalar drop _all
> quietly forval w=12(4)120 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/721 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings
> l12d_ln_rt_merch ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_2-2
> scalar drop _all
> quietly forval w=12(4)120 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings
> ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_2-4
> scalar drop _all
> quietly forval w=12(4)120 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/721 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_awkly_earnings l12d_ln_ahrly_earnings l1d_ln_atotalwkly_earn
> ings l12d_ln_rt_merch ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_2-20
> scalar drop _all
> quietly forval w=12(4)180 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings
> l12d_ln_rt_merch l1d_ln_goods_producing ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_2-21
> scalar drop _all
> quietly forval w=12(4)120 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_awkly_earnings l1d_ln_awkly_earnings l2d_ln_awkly_earnings l
> 12d_ln_ahrly_earnings ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_7-4
> scalar drop _all
> quietly forval w=12(4)180 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_awkly_hours l12
> d_ln_atotalwkly_earnings l12d_ln_rt_merch ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_awkly_earnings)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> */
.
```

```
. **********************End of awkly_earnings model selection
. **Start model selection for total employment
.
. /*
> gsreg d_ln_employment l1d_ln_employment l2d_ln_employment l3d_ln_employment l
> 12d_ln_employment l24d_ln_employment ///
> l1d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
> l1d_ln_awkly_hours l12d_ln_awkly_hours ///
> l1d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l1d_ln_rt_merch l12d_ln_rt_merch ///
> l1d_ln_unemployed l12d_ln_unemployed ///
> l1d_ln_retail_trade l12d_ln_retail_trade ///
> l1d_ln_wholesale_trade l12d_ln_wholesale_trade, ///
> ncomb(1,6) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_21) replace
>
> gsreg d_ln_employment l1d_ln_employment l2d_ln_employment l3d_ln_employment l
> 12d_ln_employment l24d_ln_employment ///
> l2d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
> l2d_ln_awkly_hours l12d_ln_awkly_hours ///
> l2d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l2d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l2d_ln_rt_merch l12d_ln_rt_merch ///
> l2d_ln_rt_food_bev l12d_ln_rt_food_bev, ///
> ncomb(1,7) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_22) replace
>
> gsreg d_ln_employment l1d_ln_employment l2d_ln_employment l12d_ln_employment
> l24d_ln_employment ///
> l1d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_awkly_earnings ///
> l1d_ln_awkly_hours l12d_ln_awkly_hours ///
> l1d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l1d_ln_rt_merch l12d_ln_rt_merch ///
> l1d_ln_goods_producing l12d_ln_goods_producing ///
> l1d_ln_retail_trade l12d_ln_retail_trade ///
> l1d_ln_wholesale_trade l12d_ln_wholesale_trade, ///
> ncomb(1,6) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_23) replace
>
> gsreg d_ln_employment l1d_ln_employment l2d_ln_employment l12d_ln_employment
> l24d_ln_employment ///
> l1d_ln_awkly_earnings l6d_ln_awkly_earnings l12d_ln_awkly_earnings l24d_ln_aw
> kly_earnings ///
> l1d_ln_awkly_hours l6d_ln_awkly_hours l12d_ln_awkly_hours ///
> l1d_ln_ahrly_earnings l6d_ln_ahrly_earnings l12d_ln_ahrly_earnings ///
> l1d_ln_atotalwkly_earnings l12d_ln_atotalwkly_earnings ///
> l1d_ln_rt_merch l6d_ln_rt_merch l12d_ln_rt_merch ///
> l1d_ln_goods_producing l12d_ln_goods_producing, ///
> ncomb(1,6) aic outsample(12) fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) samesam
> ple ///
> nindex( -0.3 aic -0.3 bic -0.4 rmse_out) results(gsreg_24) replace
> */
. /*
> *Rolling window program Initial
> scalar drop _all
> quietly forval w=12(4)180 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d.ln_employment l(1/12)d.ln_employment ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_employment)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_21-1
> scalar drop _all
> quietly forval w=12(4)180 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_employment l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_
> merch ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_employment)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_21-3
> scalar drop _all
> quietly forval w=12(4)220 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_employment l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_
> merch l1d_ln_wholesale_trade ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_employment)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_21-22
> scalar drop _all
> quietly forval w=12(4)180 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
> 
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_employment l12d_ln_employment l1d_ln_ahrly_earnings l1d_ln_a
> totalwkly_earnings l1d_ln_wholesale_trade ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_employment)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
> 
> *Rolling window program gsreg_21-25
> scalar drop _all
> quietly forval w=12(4)180 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
> 
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_employment l12d_ln_employment l1d_ln_awkly_hours ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_employment)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> *Rolling window program gsreg_21-45
> scalar drop _all
> quietly forval w=12(4)180 {
> /* w=small(inc)large
> small is the smallest window
> inc is the window size increment
> large is the largest window.
> (large-small)/inc must be an interger */
> gen pred=. // out of sample prediction
> gen nobs=. // number of observations in the window for each forecast point
>
>         forval t=580/720 {
>         /* t=first/last
>         first is the first date for which you want to make a forecast.
>         first-1 is the end date of the earliest window used to fit the model.
>         first-w, where w is the window width, is the date of the first
>         observation used to fit the model in the earliest window.
>         You must choose first so it is preceded by a full set of
>     lags for the model with the longest lag length to be estimated.
>         last is  the last observation to be forecast. */
>         gen wstart=`t'-`w' // fit window start date
>         gen wend=`t'-1 // fit window end date
>         /* Enter the regression command immediately below.
>         Leave the if statement intact to control the window  */
>         reg d_ln_employment l1d_ln_employment l3d_ln_employment l12d_ln_emplo
> yment l1d_ln_awkly_hours l12d_ln_rt_merch ///
>                 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12  ///
>                 if date>=wstart & date<=wend // restricts the model to the wi
> ndow
>         replace nobs=e(N) if date==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if date==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d_ln_employment)^2 // generating squared errors
> summ errsq // getting the mean of the squared errors
> scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
> summ nobs // getting min and max obs used
> scalar RWminobs`w'=r(min) // in obs used in the window width
> scalar RWmaxobs`w'=r(max) // max obs used in the window width
> drop errsq pred nobs // clearing for the next loop
> }
> scalar list // list the RMSE and min and max obs for each window width
> *End of rolling window program
>
> */
.
.
.
```

```
. regress d.ln_awkly_earnings l1d.ln_awkly_earnings l12d.ln_ahrly_earnings ///
> m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12

      Source |       SS           df       MS      Number of obs   =       145
-------------+----------------------------------   F(13, 131)      =      3.79
       Model |  .013391589        13  .001030122   Prob > F        =    0.0000
    Residual |  .035565626       131  .000271493   R-squared       =    0.2735
-------------+----------------------------------   Adj R-squared   =    0.2014
       Total |  .048957215       144  .000339981   Root MSE        =    .01648

------------------------------------------------------------------------------
> ----
D.                |
ln_awkly_earnings |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Inter
> val]
------------------+-----------------------------------------------------------
> ----
ln_awkly_earnings |
              LD. |  -.1855246   .0836475    -2.22   0.028    -.3509993   -.020
> 0498
                  |
ln_ahrly_earnings |
            L12D. |   .1953839   .0907017     2.15   0.033     .0159543    .374
> 8134
                  |
               m2 |   .0187647   .0070418     2.66   0.009     .0048344     .03
> 2695
               m3 |    .007141   .0068234     1.05   0.297    -.0063573    .020
> 6393
               m4 |   .0107188   .0068995     1.55   0.123      -.00293    .024
> 3676
               m5 |   .0001538    .006766     0.02   0.982     -.013231    .013
> 5385
               m6 |   .0034961   .0069778     0.50   0.617    -.0103077       .
> 0173
               m7 |   .0075111   .0069082     1.09   0.279     -.006155    .021
> 1771
               m8 |   .0070988    .006878     1.03   0.304    -.0065075    .020
> 7051
               m9 |   .0051293   .0068974     0.74   0.458    -.0085154    .018
> 7739
              m10 |   .0133952   .0068529     1.95   0.053    -.0001615    .026
> 9519
              m11 |  -.0023271   .0067506    -0.34   0.731    -.0156814    .011
> 0272
              m12 |   .0155757   .0072318     2.15   0.033     .0012695    .029
> 8818
            _cons |  -.0061856   .0049256    -1.26   0.211    -.0159295    .003
> 5583
------------------------------------------------------------------------------
> ----

. predict res, residuals
(218 missing values generated)

. pac res

. graph export "earnings_pac.png", replace
(file earnings_pac.png written in PNG format)

. ac res

. graph export "earnings_ac.png", replace
(file earnings_ac.png written in PNG format)

.
```

```
. drop res

.
. reg d_ln_employment l1d_ln_employment l3d_ln_employment l12d_ln_employment //
> /
> l1d_ln_awkly_hours l12d_ln_rt_merch m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12

      Source |       SS           df       MS      Number of obs   =       156
-------------+----------------------------------   F(16, 139)      =     35.29
       Model |  .015099942        16  .000943746   Prob > F        =    0.0000
    Residual |  .003716946       139  .000026741   R-squared       =    0.8025
-------------+----------------------------------   Adj R-squared   =    0.7797
       Total |  .018816888       155  .000121399   Root MSE        =    .00517

-------------------------------------------------------------------------------
> -----
   d_ln_employment |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Inte
> rval]
-------------------+-----------------------------------------------------------
> -----
 l1d_ln_employment |   .0376232   .0755784     0.50   0.619    -.1118087     .1
> 87055
 l3d_ln_employment |   .2860505   .0784123     3.65   0.000     .1310155    .44
> 10856
l12d_ln_employment |   .3155754    .074868     4.22   0.000     .1675481    .46
> 36026
l1d_ln_awkly_hours |  -.0869467   .0322813    -2.69   0.008    -.1507726   -.02
> 31208
  l12d_ln_rt_merch |   .0664325   .0322815     2.06   0.041     .0026062    .13
> 02588
                m2 |   .0131438   .0032253     4.08   0.000     .0067669    .01
> 95207
                m3 |   .0124148   .0029701     4.18   0.000     .0065424    .01
> 82872
                m4 |    .010674    .003468     3.08   0.003     .0038171    .01
> 75309
                m5 |   .0045726   .0026625     1.72   0.088    -.0006916    .00
> 98369
                m6 |   .0042313   .0025811     1.64   0.103    -.0008721    .00
> 93347
                m7 |    .004966   .0026984     1.84   0.068    -.0003692    .01
> 03011
                m8 |   .0131196   .0033133     3.96   0.000     .0065686    .01
> 96706
                m9 |   .0065989   .0029367     2.25   0.026     .0007925    .01
> 24052
               m10 |   .0196179   .0039392     4.98   0.000     .0118294    .02
> 74064
               m11 |    .015497   .0042681     3.63   0.000     .0070583    .02
> 39358
               m12 |   .0148873   .0038054     3.91   0.000     .0073635    .02
> 24112
             _cons |   -.009688   .0025185    -3.85   0.000    -.0146676   -.00
> 47085
-------------------------------------------------------------------------------
> -----

. predict res, residuals
(207 missing values generated)

. pac res

. graph export "employment_pac.png", replace
(file employment_pac.png written in PNG format)

. ac res
. graph export "employment_ac.png", replace
(file employment_ac.png written in PNG format)

.
.
.
. *Rolling window program - just for w=68, employment
. scalar drop _all

. quietly forval w=68(4)68 {

. *End of rolling window program
.
. summ nobs2 // checking all had a full window

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       nobs2 |        143    57.61538    16.47316         14         68

. *get error info for normal interval
. summ errsq2

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      errsq2 |        142    .0000373     .000089   2.02e-12   .0007366

. scalar rwrmse2=r(mean)^0.5

. scalar list rwrmse2
   rwrmse2 =  .00610728

. *Forecast for employment
. reg d.ln_employment l1d_ln_employment l3d_ln_employment ///
> l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_merch ///
> m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m6,2020m2)

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(16, 52)       =     15.24
       Model |   .00595879        16  .000372424   Prob > F        =    0.0000
    Residual |  .001270627        52  .000024435   R-squared       =    0.8242
-------------+----------------------------------   Adj R-squared   =    0.7702
       Total |  .007229417        68  .000106315   Root MSE        =    .00494

------------------------------------------------------------------------------
> -----
   D.ln_employment |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Inte
> rval]
-------------------+----------------------------------------------------------
> -----
 l1d_ln_employment |  -.1900882   .1503165    -1.26   0.212    -.4917204     .1
> 11544
 l3d_ln_employment |   .0483803    .124648     0.39   0.700    -.2017443    .29
> 85048
l12d_ln_employment |   .0836436   .1287955     0.65   0.519    -.1748035    .34
> 20907
l1d_ln_awkly_hours |  -.0884313   .0474929    -1.86   0.068    -.1837328    .00
> 68701
  l12d_ln_rt_merch |    .084568   .0519211     1.63   0.109    -.0196194    .18
> 87553
                m2 |   .0115027   .0044514     2.58   0.013     .0025703    .02
> 04351
                m3 |    .012899   .0044765     2.88   0.006     .0039163    .02
> 18817
                m4 |   .0025334   .0052315     0.48   0.630    -.0079643    .01
> 30311
```

```
                m5 |  -.0011418   .0040992    -0.28   0.782    -.0093674    .00
> 70838
                m6 |  -.0017142   .0040959    -0.42   0.677    -.0099332    .00
> 65048
                m7 |  -.0019606   .0046562    -0.42   0.675     -.011304    .00
> 73829
                m8 |   .0075548   .0053467     1.41   0.164     -.003174    .01
> 82837
                m9 |  -.0018172   .0045772    -0.40   0.693    -.0110019    .00
> 73675
               m10 |   .0165592   .0064113     2.58   0.013      .003694    .02
> 94244
               m11 |   .0159071     .00741     2.15   0.037     .0010379    .03
> 07762
               m12 |   .0122285   .0062804     1.95   0.057     -.000374    .02
> 48309
             _cons |  -.0033567   .0041107    -0.82   0.418    -.0116055    .00
> 48922
-------------------------------------------------------------------------------
> -----

. predict temp if tin(2020m3,2020m3)
(option xb assumed; fitted values)
(362 missing values generated)

. replace pred2=temp if tin(2020m3,2020m3)
(1 real change made)

. drop temp

. gen np_employment = exp(l.ln_employment+pred2+(rwrmse2^2)/2)
(220 missing values generated)

. gen ubn2=exp(l.ln_employment+pred2+1.96*rwrmse2+(rwrmse2^2)/2)
(220 missing values generated)

. gen lbn2=exp(l.ln_employment+pred2-1.96*rwrmse2+(rwrmse2^2)/2)
(220 missing values generated)

. list date np_employment lbn2 ubn2 if tin(2020m3,2020m3)

     +----------------------------------------+
     |   date   np_emp~t      lbn2       ubn2 |
     |----------------------------------------|
363. | 2020m3   289.6809   286.234   293.1693 |
     +----------------------------------------+

. tsline np_employment lbn2 ubn2 employment if tin(2019m4,2020m3)

.
. *Fan chart - Employment (rwrmse2)
. reg d.ln_employment l1d_ln_employment l3d_ln_employment ///
> l12d_ln_employment l1d_ln_awkly_hours l12d_ln_rt_merch ///
> m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m6,2020m2)

      Source |       SS           df       MS      Number of obs   =        69
-------------+----------------------------------   F(16, 52)       =     15.24
       Model |   .00595879        16  .000372424   Prob > F        =    0.0000
    Residual |  .001270627        52  .000024435   R-squared       =    0.8242
-------------+----------------------------------   Adj R-squared   =    0.7702
       Total |  .007229417        68  .000106315   Root MSE        =    .00494

-------------------------------------------------------------------------------
> -----
   D.ln_employment |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Inte
> rval]
```

```
-------------------+----------------------------------------------------------
> -----
 l1d_ln_employment |  -.1900882   .1503165    -1.26   0.212    -.4917204     .1
> 11544
 l3d_ln_employment |   .0483803    .124648     0.39   0.700    -.2017443    .29
> 85048
l12d_ln_employment |   .0836436   .1287955     0.65   0.519    -.1748035    .34
> 20907
l1d_ln_awkly_hours |  -.0884313   .0474929    -1.86   0.068    -.1837328    .00
> 68701
  l12d_ln_rt_merch |    .084568   .0519211     1.63   0.109    -.0196194    .18
> 87553
                m2 |   .0115027   .0044514     2.58   0.013     .0025703    .02
> 04351
                m3 |    .012899   .0044765     2.88   0.006     .0039163    .02
> 18817
                m4 |   .0025334   .0052315     0.48   0.630    -.0079643    .01
> 30311
                m5 |  -.0011418   .0040992    -0.28   0.782    -.0093674    .00
> 70838
                m6 |  -.0017142   .0040959    -0.42   0.677    -.0099332    .00
> 65048
                m7 |  -.0019606   .0046562    -0.42   0.675     -.011304    .00
> 73829
                m8 |   .0075548   .0053467     1.41   0.164     -.003174    .01
> 82837
                m9 |  -.0018172   .0045772    -0.40   0.693    -.0110019    .00
> 73675
               m10 |   .0165592   .0064113     2.58   0.013      .003694    .02
> 94244
               m11 |   .0159071     .00741     2.15   0.037     .0010379    .03
> 07762
               m12 |   .0122285   .0062804     1.95   0.057     -.000374    .02
> 48309
             _cons |  -.0033567   .0041107    -0.82   0.418    -.0116055    .00
> 48922
------------------------------------------------------------------------------
> -----

. predict fpd_employment
(option xb assumed; fitted values)
(206 missing values generated)

. gen fp_employment=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment) if d
> ate==tm(2020m3)
(362 missing values generated)

. 
. gen ub1=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment+rwrmse2) if dat
> e==tm(2020m3)
(362 missing values generated)

. gen lb1=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment-rwrmse2) if dat
> e==tm(2020m3)
(362 missing values generated)

. 
. gen ub2=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment+2*rwrmse2) if d
> ate==tm(2020m3)
(362 missing values generated)

. gen lb2=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment-2*rwrmse2) if d
> ate==tm(2020m3)
(362 missing values generated)

. 
```

```
. gen ub3=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment+3*rwrmse2) if d
> ate==tm(2020m3)
(362 missing values generated)

. gen lb3=exp((rwrmse2^2)/2)*exp(l.ln_employment+fpd_employment-3*rwrmse2) if d
> ate==tm(2020m3)
(362 missing values generated)

.
. drop fpd_employment

.
. gen y=employment if tin(2019m1,2020m2)
(349 missing values generated)

. replace y=fp_employment if date>tm(2020m2)
(1 real change made)

.
. gen yub1=employment if date==tm(2020m2)
(362 missing values generated)

. replace yub1=ub1 if date>tm(2020m2)
(1 real change made)

. gen ylb1=employment if date==tm(2020m2)
(362 missing values generated)

. replace ylb1=lb1 if date>tm(2020m2)
(1 real change made)

.
. gen yub2=employment if date==tm(2020m2)
(362 missing values generated)

. replace yub2=ub2 if date>tm(2020m2)
(1 real change made)

. gen ylb2=employment if date==tm(2020m2)
(362 missing values generated)

. replace ylb2=lb2 if date>tm(2020m2)
(1 real change made)

.
. gen yub3=employment if date==tm(2020m2)
(362 missing values generated)

. replace yub3=ub3 if date>tm(2020m2)
(1 real change made)

. gen ylb3=employment if date==tm(2020m2)
(362 missing values generated)

. replace ylb3=lb3 if date>tm(2020m2)
(1 real change made)

.
. cd ..
C:\Users\Josh\Desktop\School\3rd\TimeSeries\Project

.
. twoway (tsrline yub3 yub2 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
>         (tsrline yub2 yub1 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
>         (tsrline yub1 y if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
>         (tsrline y ylb1 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
>         (tsrline ylb1 ylb2 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
>         (tsrline ylb2 ylb3 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
>         (tsline employment y if tin(2019m1,2020m3) , ///
>         lcolor(gs6) lwidth(thick) ), scheme(s1mono) legend(off) ///
>         title("MSA Employment Forecast") legend(off) ///
>         xtitle("") ylabel(,grid) ///
>         note("Launch date is 2020m2" "Bands at 1, 2, and 3 sigma")

. graph export "Emp_FanChart.png", replace
(file Emp_FanChart.png written in PNG format)

.
.
. *Rolling window program - just for w=88, awkly_earnings
. scalar drop _all

. quietly forval w=88(4)88 {

. *End of rolling window program
.
. summ nobs10 // checking all had a full window

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      nobs10 |        143    62.44056    28.44754          3         88

. *get error info for normal interval
. summ errsq10

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
     errsq10 |        142    .0003868    .0006579   4.87e-09   .0053473

. scalar rwrmse10=r(mean)^0.5

. scalar list rwrmse10
  rwrmse10 =  .01966677

. *Forecast for awkly_earnings
. reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings ///
> m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2012m10,2020m2)

      Source |       SS           df       MS      Number of obs   =        89
-------------+----------------------------------   F(13, 75)       =      3.61
       Model |  .012455583        13  .000958122   Prob > F        =    0.0002
    Residual |  .019925466        75  .000265673   R-squared       =    0.3847
-------------+----------------------------------   Adj R-squared   =    0.2780
       Total |  .032381049        88  .000367966   Root MSE        =     .0163

------------------------------------------------------------------------------
> ---------
   d_ln_awkly_earnings |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
> Interval]
----------------------+-------------------------------------------------------
> ---------
 l1d_ln_awkly_earnings |  -.3508674   .1048547    -3.35   0.001    -.5597487
>  -.141986
l12d_ln_ahrly_earnings |   .3053894   .1428897     2.14   0.036     .0207385
>  .5900403
                    m2 |   .0130025   .0086593     1.50   0.137    -.0042477
```

```
>  .0302528
                    m3 |   .0018513   .0084586     0.22   0.827    -.0149991
>  .0187016
                    m4 |   .0025141   .0090793     0.28   0.783    -.0155728
>  .0206009
                    m5 |   -.005247   .0085011    -0.62   0.539     -.022182
>  .0116881
                    m6 |  -.0044902   .0089294    -0.50   0.617    -.0222785
>   .013298
                    m7 |   .0044093   .0088482     0.50   0.620    -.0132173
>  .0220359
                    m8 |   .0057477   .0086256     0.67   0.507    -.0114352
>  .0229307
                    m9 |  -.0048992   .0089272    -0.55   0.585     -.022683
>  .0128847
                   m10 |    .007048   .0083935     0.84   0.404    -.0096727
>  .0237687
                   m11 |   .0052311   .0082375     0.64   0.527    -.0111787
>   .021641
                   m12 |   .0137919   .0088932     1.55   0.125    -.0039242
>   .031508
                 _cons |  -.0000724   .0060985    -0.01   0.991    -.0122213
>  .0120765
------------------------------------------------------------------------------
> ---------

. predict temp if tin(2020m3,2020m3)
(option xb assumed; fitted values)
(362 missing values generated)

. replace pred10=temp if tin(2020m3,2020m3)
(1 real change made)

. drop temp

. gen np_awkly_earnings = exp(l.ln_awkly_earnings+pred10+(rwrmse10^2)/2)
(220 missing values generated)

. gen ubn10=exp(l.ln_awkly_earnings+pred10+1.96*rwrmse10+(rwrmse10^2)/2)
(220 missing values generated)

. gen lbn10=exp(l.ln_awkly_earnings+pred10-1.96*rwrmse10+(rwrmse10^2)/2)
(220 missing values generated)

. list date np_awkly_earnings lbn10 ubn10 if tin(2020m3,2020m3)

     +-----------------------------------------+
     |   date   np_awk~s      lbn10      ubn10 |
     |-----------------------------------------|
363. | 2020m3    881.092   847.7749   915.7184 |
     +-----------------------------------------+

. tsline np_awkly_earnings lbn10 ubn10 awkly_earnings if tin(2019m4,2020m3)

. 
. drop ub1 lb1 ub2 lb2 ub3 lb3

. *Fan chart - Avg Weekly Earnings (rwrmse10)
. reg d_ln_awkly_earnings l1d_ln_awkly_earnings l12d_ln_ahrly_earnings ///
> m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2012m10,2020m2)

      Source |       SS           df       MS      Number of obs   =        89
-------------+----------------------------------   F(13, 75)       =      3.61
       Model |  .012455583        13  .000958122   Prob > F        =    0.0002
    Residual |  .019925466        75  .000265673   R-squared       =    0.3847
-------------+----------------------------------   Adj R-squared   =    0.2780
```

```
       Total |  .032381049        88  .000367966   Root MSE        =     .0163

------------------------------------------------------------------------------
> ---------
   d_ln_awkly_earnings |      Coef.   Std. Err.      t    P>|t|     [95% Conf.
> Interval]
----------------------+-------------------------------------------------------
> ---------
 l1d_ln_awkly_earnings |  -.3508674   .1048547    -3.35   0.001    -.5597487
>  -.141986
l12d_ln_ahrly_earnings |   .3053894   .1428897     2.14   0.036     .0207385
>  .5900403
                    m2 |   .0130025   .0086593     1.50   0.137    -.0042477
>  .0302528
                    m3 |   .0018513   .0084586     0.22   0.827    -.0149991
>  .0187016
                    m4 |   .0025141   .0090793     0.28   0.783    -.0155728
>  .0206009
                    m5 |   -.005247   .0085011    -0.62   0.539     -.022182
>  .0116881
                    m6 |  -.0044902   .0089294    -0.50   0.617    -.0222785
>   .013298
                    m7 |   .0044093   .0088482     0.50   0.620    -.0132173
>  .0220359
                    m8 |   .0057477   .0086256     0.67   0.507    -.0114352
>  .0229307
                    m9 |  -.0048992   .0089272    -0.55   0.585     -.022683
>  .0128847
                   m10 |    .007048   .0083935     0.84   0.404    -.0096727
>  .0237687
                   m11 |   .0052311   .0082375     0.64   0.527    -.0111787
>   .021641
                   m12 |   .0137919   .0088932     1.55   0.125    -.0039242
>   .031508
                 _cons |  -.0000724   .0060985    -0.01   0.991    -.0122213
>  .0120765
------------------------------------------------------------------------------
> ---------

. predict fpd_awkly_earnings
(option xb assumed; fitted values)
(217 missing values generated)

. gen fp_awkly_earnings=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_e
> arnings) if date==tm(2020m3)
(362 missing values generated)

. 
. gen ub1=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings+rwrmse
> 10) if date==tm(2020m3)
(362 missing values generated)

. gen lb1=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings-rwrmse
> 10) if date==tm(2020m3)
(362 missing values generated)

. 
. gen ub2=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings+2*rwrm
> se10) if date==tm(2020m3)
(362 missing values generated)

. gen lb2=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings-2*rwrm
> se10) if date==tm(2020m3)
(362 missing values generated)

. 
```

```
. gen ub3=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings+3*rwrm
> se10) if date==tm(2020m3)
(362 missing values generated)

. gen lb3=exp((rwrmse10^2)/2)*exp(l.ln_awkly_earnings+fpd_awkly_earnings-3*rwrm
> se10) if date==tm(2020m3)
(362 missing values generated)


.
. drop fpd_awkly_earnings


.
. drop y yub1 yub2 ylb1 ylb2 yub3 ylb3

. gen y=awkly_earnings if tin(2019m1,2020m2)
(349 missing values generated)

. replace y=fp_awkly_earnings if date>tm(2020m2)
(1 real change made)


.
. gen yub1=awkly_earnings if date==tm(2020m2)
(362 missing values generated)

. replace yub1=ub1 if date>tm(2020m2)
(1 real change made)

. gen ylb1=awkly_earnings if date==tm(2020m2)
(362 missing values generated)

. replace ylb1=lb1 if date>tm(2020m2)
(1 real change made)


.
. gen yub2=awkly_earnings if date==tm(2020m2)
(362 missing values generated)

. replace yub2=ub2 if date>tm(2020m2)
(1 real change made)

. gen ylb2=awkly_earnings if date==tm(2020m2)
(362 missing values generated)

. replace ylb2=lb2 if date>tm(2020m2)
(1 real change made)


.
. gen yub3=awkly_earnings if date==tm(2020m2)
(362 missing values generated)

. replace yub3=ub3 if date>tm(2020m2)
(1 real change made)

. gen ylb3=awkly_earnings if date==tm(2020m2)
(362 missing values generated)

. replace ylb3=lb3 if date>tm(2020m2)
(1 real change made)


.
. twoway (tsrline yub3 yub2 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
>         (tsrline yub2 yub1 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
>         (tsrline yub1 y if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
>         (tsrline y ylb1 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
>         (tsrline ylb1 ylb2 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
>         (tsrline ylb2 ylb3 if tin(2020m2,2020m3), ///
>         recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
>         (tsline awkly_earnings y if tin(2019m1,2020m3) , ///
>         lcolor(gs6) lwidth(thick) ), scheme(s1mono) legend(off) ///
>         title("MSA Avg Weekly Earnings Forecast") legend(off) ///
>         xtitle("") ylabel(,grid) ///
>         note("Launch date is 2020m2" "Bands at 1, 2, and 3 sigma")

. graph export "awkly_earnings_FanChart.png", replace
(file awkly_earnings_FanChart.png written in PNG format)

.
.
. log close
      name:  <unnamed>
       log:  C:\Users\Josh\Desktop\School\3rd\TimeSeries\Project\ProjectLog.smc
> l
  log type:  smcl
 closed on:  16 Apr 2020, 22:50:56
--------------------------------------------------------------------------------
```