Below is a sketch of crucial content. Your report should present this and explain it in the form of a professional report.

GSREG was used to search through nearly 700,000 models to forecast non-farm employment in Florida one month ahead. All variables as log first differences. To keep the number of models to process down to something that could run overnight, instead of over days, I made the following choices (yours may differ):
1) fixed 11 month indicators
2) for predictors I left out US building permits, keeping the Florida labor force, Florida building permits, and the US prime age employment to population ratio
3) considered only lags 1-4, 12, and 24 of the dependent variable
4) considered only lags 1-4 and 12 of the independent variables
5) considered only up to 9 predictors (beyond the intercept and month indicators)

The table below presents information on the GSREG results. The panel labeled "Percent Present in Top X Models" shows how often (per 100 models) each variable was found in the top 100, 200, 500, and 1000 GSREG results. The panel labeled "GSREG Model Rank and Variables" gives the best model of different sizes (3-7 predictors) and the variables included. "Y" indicated the variable was included. I estimate each of these using the rolling window procedure from class. The panel labeled "Benchmark Models" provides variables included in three benchmark models I also estimated using the rolling window procedure for reference. The first of these is purely AR. The second included all variables that were present in at least 250 of the top 1000 GSREG models. The third includes all predictor variables.

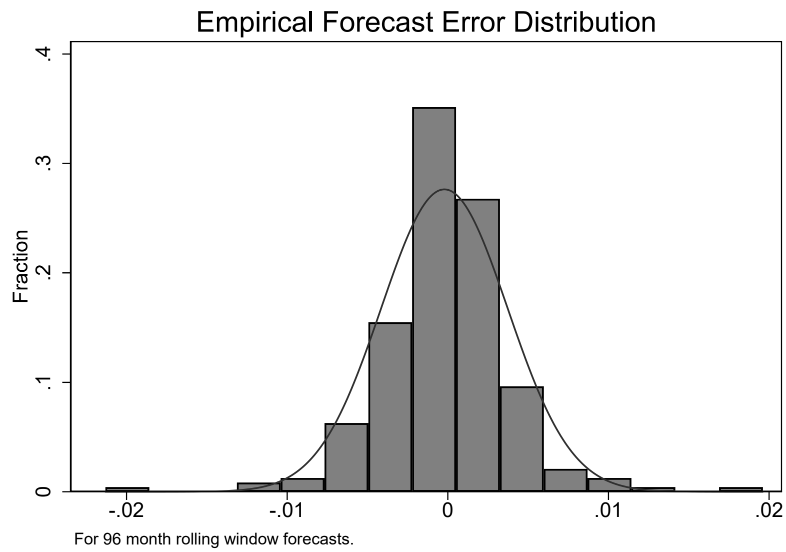
Summary of GSREG and Rolling Window Results

		Percent Present in Top X Models				GSREG Model Rank and Variables					Benchmark Models		
Variable	Lag	100	200	500	1000	1	2	4	7	37			
FL Non-Farm Emp	1	1	1	5	7						Y		Y
	2	18	20	25	28						Y	Y	Y
	3	100	100	100	100	Y	Y	Y	Y	Y	Y	Y	Y
	4	79	73	71	67	Y	Y	Y		Y	Y	Y	Y
	12	100	100	100	100	Y	Y	Y	Y	Y	Y	Y	Y
	24	100	100	100	100	Y	Y	Y	Y	Y	Y	Y	Y
FL Labor Force	1	0	0	0	1								Y
	2	22	22	30	34							Y	Y
	3	18	22	26	30					Y		Y	Y
	4	10	14	18	22								Y
	12	0	1	1	4								Y
US Emp:Pop Ratio	1	0	0	0	0								Y
	2	10	15	20	25							Y	Y
	3	15	22	26	29							Y	Y
	4	33	36	40	42		Y	Y		Y		Y	Y
	12	0	0	0	0								Y
FL Building Permits	1	20	20	27	28							Y	Y
	2	31	34	36	40			Y		Y		Y	Y
	3	15	15	20	22								Y
	4	8	13	16	20								Y
	12	0	1	2	4								Y
# Variables Included:						4	5	6	3	7	6	12	21
Best Window Size						72	96	96	72	96	72	72	96
Best Window RMSE						.00358	.00356	.00358	.00357	.00359	.00367	.00368	.00406

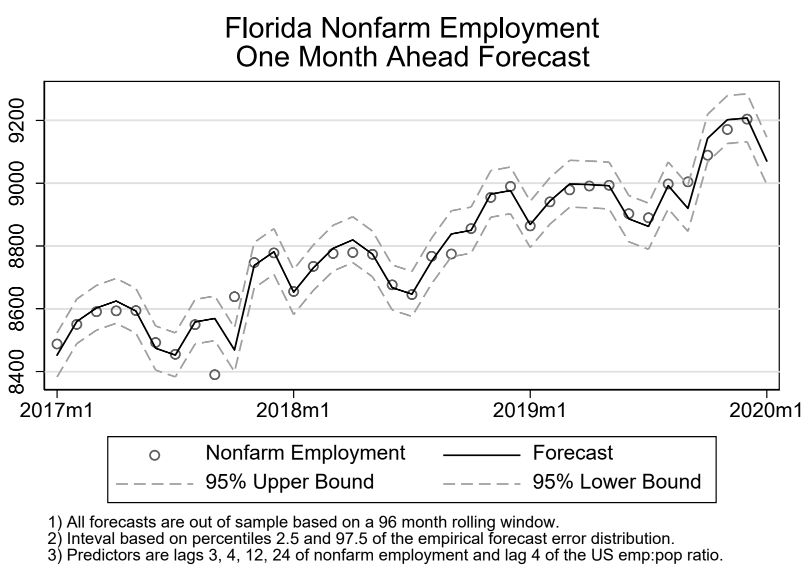
The lower rows of the table above provide results for each model from the rolling window procedure. The best model from the rolling window procedure was the second ranked model form GSREG. I use this for my forecast. The differences were small, and an argument could be made for the GSREG models ranked 1, 4, or 7 as well. The model fit for the actual forecast is then:

reg d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr m2…m12.

The residuals from the rolling window procedure fitting that model fit to all windows of 8 years possible since 1990m1 are used to model the error distribution. The forecast error distribution is shown in the figure on the top right.

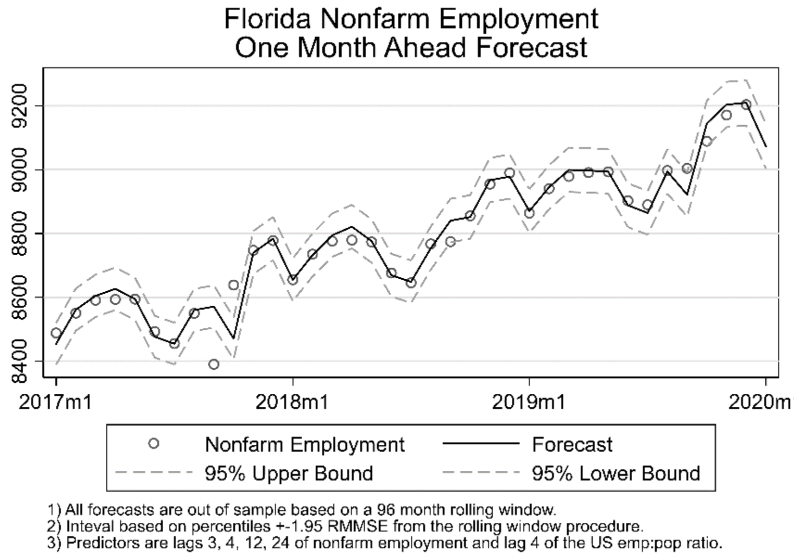


The point forecast for the one month change in the log of nonfarm employment from 2019m12 to 2020m1 is based on the last available eight-year window, 2012m1 to 2019m12. Using an empirical approach for the mean correction factor upon exponentiating the log of predicted nonfarm employment and for the 95% forecast interval, the forecast for January 2020 is nonfarm employment of 9,070,723, with a 95% forecast interval from 8,996,228 to 9,146,527. The middle figure to the right provides a visual depiction of the empirical forecast accuracy and the forecast for January 2020.

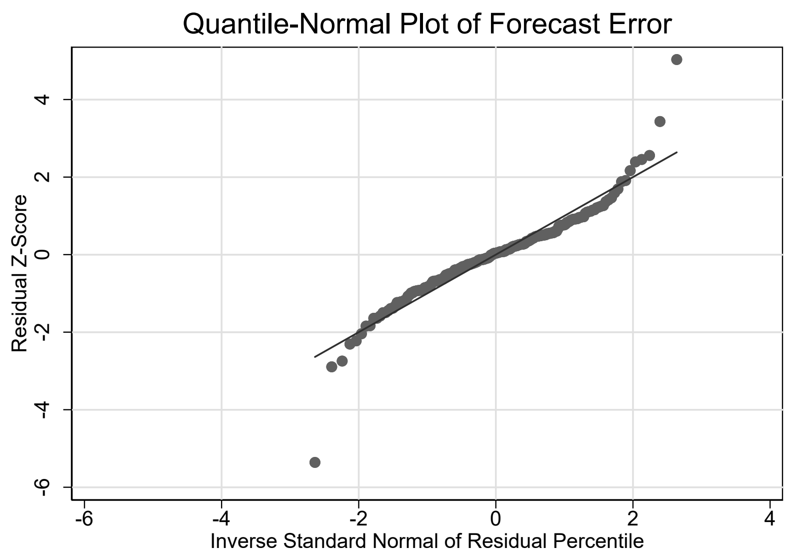


Assuming normality gives a forecast of 9,072,653, with lower and upper bounds at 9,142,932 and 9,002,915. This point and interval forecast are represented top figure on the next page.

Which is better? Imposing normality, if true, provides more accuracy. Not imposing it is more robust when the error distribution is non-normal. If the distribution is near normal, the two would only differ substantially if sparse data in the tails caused the empirical estimates of percentiles 2.5 and 97.5 to be imprecisely estimated.



The error distribution on the previous page appears relatively symmetric, but the degree of divergence form normal is difficult to detect in a histogram. The figure below shows the quantile-normal plot (from Statistics 1). The tails are clearly thicker than the normal distribution, so the normal assumption is not accurate. On the other hand, data in the tails is indeed sparse. So, we are somewhat lucky here that the two agree, so we don't have to choose.



Appendix A: Do File

```
*Problem Set 5 Solution

clear
set more off
cd "Your Working Directory here"
log using "Problem Set 5 Work", replace


** data prep
import delimited using "us and florida economic time series.txt"
rename observation_date datestring
gen dateday=date(datestring,"YMD")
gen date=mofd(dateday)
format date %tm
tsset date
tsappend, add(1)
generate month=month(dofm(date))
tabulate month, generate(m)
keep if date>=tm(1990m1)
rename flbppriv fl_bp
rename fllfn fl_lf
rename flnan fl_nonfarm
rename lnu02300000_20200110 us_epr
gen lnflnonfarm=ln( fl_nonfarm)
gen lnfllf=ln( fl_lf)
gen lnusepr = ln(us_epr)
gen lnflbp=ln( fl_bp)

*generate differences and lags thereof for use with gsreg
gen dlnflnonfarm=d.lnflnonfarm
gen ldlnflnonfarm=ld.lnflnonfarm
gen l2dlnflnonfarm=l2d.lnflnonfarm
gen l3dlnflnonfarm=l3d.lnflnonfarm
gen l4dlnflnonfarm=l4d.lnflnonfarm
gen l5dlnflnonfarm=l3d.lnflnonfarm
gen l6dlnflnonfarm=l4d.lnflnonfarm
gen l12dlnflnonfarm=l12d.lnflnonfarm
gen l24dlnflnonfarm=l24d.lnflnonfarm

gen ldlnusepr=ld.lnusepr
gen l2dlnusepr=l2d.lnusepr
gen l3dlnusepr=l3d.lnusepr
gen l4dlnusepr=l4d.lnusepr
gen l12dlnusepr=l12d.lnusepr

gen ldlnflbp=ld.lnflbp
gen l2dlnflbp=l2d.lnflbp
gen l3dlnflbp=l3d.lnflbp
gen l4dlnflbp=l4d.lnflbp
gen l12dlnflbp=l12d.lnflbp

gen ldlnfllf=ld.lnfllf
gen l2dlnfllf=l2d.lnfllf
gen l3dlnfllf=l3d.lnfllf
gen l4dlnfllf=l4d.lnfllf
gen l12dlnfllf=l12d.lnfllf
```

```
tab month, generate(md)


*Turn gsreg on by uncommenting it only when you want it to run

/*
gsreg dlnflnonfarm ldlnflnonfarm l2dlnflnonfarm l3dlnflnonfarm ///
      l4dlnflnonfarm l12dlnflnonfarm l24dlnflnonfarm ///
      ldlnfllf l2dlnfllf l3dlnfllf l4dlnfllf l12dlnfllf ///
      ldlnusepr l2dlnusepr l3dlnusepr l4dlnusepr l12dlnusepr ///
      ldlnflbp l2dlnflbp l3dlnflbp l4dlnflbp l12dlnflbp , ///
      nocount results(ps5models2.dta) replace ///
    fix(md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12) ncomb(1,9) ///
      aic outsample(24) nindex( -1 aic -1 bic -1 rmse_out) samesample
*/



/*
Checking the gsreg output, the best model with 3, 4, 5, 5, or 7 variables:
GRSEG Rank 1: d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm
GSREG Rank 2: d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr
GSREG Rank 4: d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr
l(2)d.lnflbp
GSREG Rank 7: d.lnflnonfarm l(3,12,24)d.lnflnonfarm
GSREG Rank 37: d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(3)d.lnfllf ///
            l(4)d.lnusepr l(2)d.lnflbp

AS benchmarks, I also consider:
A purely AR model with all lags of y searched:
      reg d.lnflnonfarm l(1/4,12,24)d.lnflnonfarm
A model with all the variables searched over:
      reg d.lnflnonfarm l(1/4,12,24)d.lnflnonfarm l(1/4)d.lnfllf
            l(1/4)d.lnusepr l(1/4)d.lnflbp
A model with all the variables that most commonly appeared in the top 1000
      reg d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(2,3)d.lnfllf ///
            l(2,3,4)d.lnusepr l(1,2)d.lnflbp
*/




************************
*Three benchmark models


*Rolling window program
scalar drop _all
quietly forval w=48(12)180 {
/* 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=565/719 {
      /* 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.lnflnonfarm l(1/4,12,24)d.lnflnonfarm l(1/4)d.lnfllf ///
            l(1/4)d.lnusepr l(1/4)d.lnflbp ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            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.lnflnonfarm)^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
scalar drop _all
quietly forval w=48(12)180 {
/* 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=565/719 {
      /* 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.lnflnonfarm l(1/4,12,24)d.lnflnonfarm ///
      md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            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.lnflnonfarm)^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
scalar drop _all
quietly forval w=48(12)180 {
/* 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=565/719 {
      /* 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.lnflnonfarm l(2/4,12,24)d.lnflnonfarm l(2,3)d.lnfllf  ///
            l(2/4)d.lnusepr l(1/2)d.lnflbp ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            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.lnflnonfarm)^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
```

```
***************************

*Five GSREG models


*Rolling window program
scalar drop _all
quietly forval w=48(12)180 {
/* 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=565/719 {
      /* 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.lnflnonfarm l(3,4,12,24)d.lnflnonfarm ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            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.lnflnonfarm)^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
scalar drop _all
quietly forval w=48(12)180 {
/* 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=565/719 {
      /* 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.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            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.lnflnonfarm)^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
scalar drop _all
quietly forval w=48(12)180 {
/* 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=565/719 {
      /* 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.lnflnonfarm l(3,4,12,24)d.lnflnonfarm ///
            l(4)d.lnusepr l(2)d.lnflbp ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
```

```
            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.lnflnonfarm)^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
scalar drop _all
quietly forval w=48(12)180 {
/* 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=565/719 {
      /* 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.lnflnonfarm l(3,12,24)d.lnflnonfarm ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            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.lnflnonfarm)^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
scalar drop _all
quietly forval w=48(12)180 {
/* 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=565/719 {
      /* 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.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(3)d.lnfllf  ///
            l(4)d.lnusepr l(2)d.lnflbp ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            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.lnflnonfarm)^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


*/

/* GSREG Model 2 at 8 years is the best, by a hair, on RWRMSE.
Could easily choose GSREG 1, 4, or 7 too.
Run Rolling Window again, just for w=96, for GSREG 2
Don't clear for next loop,  to get info on this model. */

*Rolling window program
scalar drop _all
gen pred=. // out of sample prediction
```

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

      quietly forval t=481/719 {
      /* 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'- 96 // 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.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            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 res=d.lnflnonfarm-pred
gen errsq=res^2 // generating squared errors
summ errsq // getting the mean of the squared errors
scalar RWrmse96=r(mean)^.5 // getting the rmse for window width i
summ nobs // getting min and max obs used
scalar RWminobs96=r(min) // min obs used in the window width
scalar RWmaxobs96=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


*Forecast from selected model

reg d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr ///
            md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
            if tin(2012m1,2019m12)
predict temp if date==tm(2020m1)
replace pred=temp if date==tm(2020m1)

*Empirical forecast and interval
gen expres=exp(res)
summ expres
gen epy=exp(l.lnflnonfarm+pred)*r(mean)
_pctile res, percentiles(2.5,97.5)
gen eub=epy*exp(r(r2))
gen elb=epy*exp(r(r1))

twoway (scatter fl_nonfarm date if tin(2017m1,2019m12) , m(Oh) ) ///
      (tsline epy eub elb if tin(2017m1,2020m1) , ///
            lpattern(solid dash dash) lcolor(black gs10 gs10) ) , ///
      saving(ps5_fcst, replace) scheme(s1mono) ylabel(,grid) xtitle("") ///
      legend(label(1 "Nonfarm Employment") label(2 "Forecast") ///
            label(3 "95% Upper Bound") label(4 "95% Lower Bound") ) ///
```

```
      title("Florida Nonfarm Employment" "One Month Ahead Forecast") ///
      note("1) All forecasts are out of sample based on a 96 month rolling
window." ///
      "2) Inteval based on percentiles 2.5 and 97.5 of the empirical forecast
error distribution." ///
      "3) Predictors are lags 3, 4, 12, 24 of nonfarm employment and lag 4 of
the US emp:pop ratio." )

graph export ps5empfcst.emf, replace

list epy eub elb if date==tm(2020m1)

*Normal forecast and interval
gen npy=exp(l.lnflnonfarm+pred+(RWrmse96^2)/2)
gen nub=npy*exp(1.96*RWrmse96)
gen nlb=npy/exp(1.96*RWrmse96)

twoway (scatter fl_nonfarm date if tin(2017m1,2019m12) , m(Oh) ) ///
      (tsline npy nub nlb if tin(2017m1,2020m1) , ///
            lpattern(solid dash dash) lcolor(black gs10 gs10) ) , ///
      saving(ps5_fcst, replace) scheme(s1mono) ylabel(,grid) xtitle("") ///
      legend(label(1 "Nonfarm Employment") label(2 "Forecast") ///
            label(3 "95% Upper Bound") label(4 "95% Lower Bound") ) ///
      title("Florida Nonfarm Employment" "One Month Ahead Forecast") ///
      note("1) All forecasts are out of sample based on a 96 month rolling
window." ///
      "2) Inteval based on percentiles +-1.95 RMMSE from the rolling window
procedure." ///
      "3) Predictors are lags 3, 4, 12, 24 of nonfarm employment and lag 4 of
the US emp:pop ratio." )

graph export ps5normfcst.emf, replace

list npy nub nlb if date==tm(2020m1)


hist res, frac scheme(s1mono) title("Empirical Forecast Error Distribution")
///
      xtitle("") note("For 96 month rolling window forecasts.")
graph export ps5errdist.emf

clear
log close
```

Appendix B: Log File

```
-------------------------------------------------------------------------------------
      log:  Problem Set 5 Work.smcl
 log type:  smcl
opened on:  10 Apr 2020, 18:13:25

.
. ** data prep
. import delimited using "us and florida economic time series.txt"
(5 vars, 972 obs)

. rename observation_date datestring

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

. gen date=mofd(dateday)

. format date %tm

. tsset date
        time variable:  date, 1939m1 to 2019m12
                delta:  1 month

. tsappend, add(1)

. generate month=month(dofm(date))

. tabulate month, generate(m)

      month |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         82        8.43        8.43
          2 |         81        8.32       16.75
          3 |         81        8.32       25.08
          4 |         81        8.32       33.40
          5 |         81        8.32       41.73
          6 |         81        8.32       50.05
          7 |         81        8.32       58.38
          8 |         81        8.32       66.70
          9 |         81        8.32       75.03
         10 |         81        8.32       83.35
         11 |         81        8.32       91.68
         12 |         81        8.32      100.00
------------+-----------------------------------
      Total |        973      100.00

. keep if date>=tm(1990m1)
(612 observations deleted)

. rename flbppriv fl_bp

. rename fllfn fl_lf

. rename flnan fl_nonfarm

. rename lnu02300000_20200110 us_epr

. gen lnflnonfarm=ln( fl_nonfarm)
(1 missing value generated)

. gen lnfllf=ln( fl_lf)
(1 missing value generated)
```

```
. gen lnusepr = ln(us_epr)
(1 missing value generated)

. gen lnflbp=ln( fl_bp)
(1 missing value generated)

.
. *generate differences and lags thereof for use with gsreg
. gen dlnflnonfarm=d.lnflnonfarm
(2 missing values generated)

. gen ldlnflnonfarm=ld.lnflnonfarm
(2 missing values generated)

. gen l2dlnflnonfarm=l2d.lnflnonfarm
(3 missing values generated)

. gen l3dlnflnonfarm=l3d.lnflnonfarm
(4 missing values generated)

. gen l4dlnflnonfarm=l4d.lnflnonfarm
(5 missing values generated)

. gen l5dlnflnonfarm=l3d.lnflnonfarm
(4 missing values generated)

. gen l6dlnflnonfarm=l4d.lnflnonfarm
(5 missing values generated)

. gen l12dlnflnonfarm=l12d.lnflnonfarm
(13 missing values generated)

. gen l24dlnflnonfarm=l24d.lnflnonfarm
(25 missing values generated)

.
. gen ldlnusepr=ld.lnusepr
(2 missing values generated)

. gen l2dlnusepr=l2d.lnusepr
(3 missing values generated)

. gen l3dlnusepr=l3d.lnusepr
(4 missing values generated)

. gen l4dlnusepr=l4d.lnusepr
(5 missing values generated)

. gen l12dlnusepr=l12d.lnusepr
(13 missing values generated)

.
. gen ldlnflbp=ld.lnflbp
(2 missing values generated)

. gen l2dlnflbp=l2d.lnflbp
(3 missing values generated)

. gen l3dlnflbp=l3d.lnflbp
(4 missing values generated)

. gen l4dlnflbp=l4d.lnflbp
(5 missing values generated)
```

```
. gen l12dlnflbp=l12d.lnflbp
(13 missing values generated)

.
. gen ldlnfllf=ld.lnfllf
(2 missing values generated)

. gen l2dlnfllf=l2d.lnfllf
(3 missing values generated)

. gen l3dlnfllf=l3d.lnfllf
(4 missing values generated)

. gen l4dlnfllf=l4d.lnfllf
(5 missing values generated)

. gen l12dlnfllf=l12d.lnfllf
(13 missing values generated)

.
.
. tab month, generate(md)

      month |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         31        8.59        8.59
          2 |         30        8.31       16.90
          3 |         30        8.31       25.21
          4 |         30        8.31       33.52
          5 |         30        8.31       41.83
          6 |         30        8.31       50.14
          7 |         30        8.31       58.45
          8 |         30        8.31       66.76
          9 |         30        8.31       75.07
         10 |         30        8.31       83.38
         11 |         30        8.31       91.69
         12 |         30        8.31      100.00
------------+-----------------------------------
      Total |        361      100.00

.
.
. *Turn gsreg on by uncommenting it only when you want it to run
.
. /*
> gsreg dlnflnonfarm ldlnflnonfarm l2dlnflnonfarm l3dlnflnonfarm ///
>         l4dlnflnonfarm l12dlnflnonfarm l24dlnflnonfarm ///
>         ldlnfllf l2dlnfllf l3dlnfllf l4dlnfllf l12dlnfllf ///
>         ldlnusepr l2dlnusepr l3dlnusepr l4dlnusepr l12dlnusepr ///
>         ldlnflbp l2dlnflbp l3dlnflbp l4dlnflbp l12dlnflbp , ///
>         nocount results(ps5models2.dta) replace ///
>     fix(md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12) ncomb(1,9) ///
>         aic outsample(24) nindex( -1 aic -1 bic -1 rmse_out) samesample
> */
.
.
.
. /*
> Checking the gsreg output, the best model with 3, 4, 5, 5, or 7 variables:
> GRSEG Rank 1: d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm
> GSREG Rank 2: d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr
> GSREG Rank 4: d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr l(2)d.lnflbp
```

```
> GSREG Rank 7: d.lnflnonfarm l(3,12,24)d.lnflnonfarm
> GSREG Rank 37: d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(3)d.lnfllf ///
>                l(4)d.lnusepr l(2)d.lnflbp
>
> AS benchmarks, I also consider:
> A purely AR model with all lags of y searched:
>         reg d.lnflnonfarm l(1/4,12,24)d.lnflnonfarm
> A model with all the variables searched over:
>         reg d.lnflnonfarm l(1/4,12,24)d.lnflnonfarm l(1/4)d.lnfllf
>                l(1/4)d.lnusepr l(1/4)d.lnflbp
> A model with all the variables that most commonly appeared in the top 1000
>         reg d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(2,3)d.lnfllf ///
>                l(2,3,4)d.lnusepr l(1,2)d.lnflbp
> */
.
.
.
.
.
. ************************
. *Three benchmark models
.
.
. *Rolling window program
. scalar drop _all

. quietly forval w=48(12)180 {

. scalar list // list the RMSE and min and max obs for each window width
RWmaxobs180 =        180
RWminobs180 =        180
 RWrmse180 =  .00430156
RWmaxobs168 =        168
RWminobs168 =        168
 RWrmse168 =  .00427944
RWmaxobs156 =        156
RWminobs156 =        156
 RWrmse156 =  .00422851
RWmaxobs144 =        144
RWminobs144 =        144
 RWrmse144 =  .00440661
RWmaxobs132 =        132
RWminobs132 =        132
 RWrmse132 =   .0043515
RWmaxobs120 =        120
RWminobs120 =        120
 RWrmse120 =  .00434587
RWmaxobs108 =        108
RWminobs108 =        108
 RWrmse108 =  .00426457
RWmaxobs96 =         96
RWminobs96 =         96
  RWrmse96 =   .0040682
RWmaxobs84 =         84
RWminobs84 =         84
  RWrmse84 =  .00415002
RWmaxobs72 =         72
RWminobs72 =         72
  RWrmse72 =  .00446223
RWmaxobs60 =         60
RWminobs60 =         60
  RWrmse60 =   .0048816
RWmaxobs48 =         48
```

```
RWminobs48 =         48
  RWrmse48 =  .00597765

. *End of rolling window program
.
.
. *Rolling window program
. scalar drop _all

. quietly forval w=48(12)180 {

. scalar list // list the RMSE and min and max obs for each window width
RWmaxobs180 =        180
RWminobs180 =        180
 RWrmse180 =  .00414008
RWmaxobs168 =        168
RWminobs168 =        168
 RWrmse168 =  .00414015
RWmaxobs156 =        156
RWminobs156 =        156
 RWrmse156 =  .00411297
RWmaxobs144 =        144
RWminobs144 =        144
 RWrmse144 =  .00424732
RWmaxobs132 =        132
RWminobs132 =        132
 RWrmse132 =  .00419212
RWmaxobs120 =        120
RWminobs120 =        120
 RWrmse120 =  .00416155
RWmaxobs108 =        108
RWminobs108 =        108
 RWrmse108 =   .0041845
RWmaxobs96 =         96
RWminobs96 =         96
  RWrmse96 =  .00376715
RWmaxobs84 =         84
RWminobs84 =         84
  RWrmse84 =  .00367764
RWmaxobs72 =         72
RWminobs72 =         72
  RWrmse72 =  .00379896
RWmaxobs60 =         60
RWminobs60 =         60
  RWrmse60 =  .00413651
RWmaxobs48 =         48
RWminobs48 =         48
  RWrmse48 =  .00439839

. *End of rolling window program
.
.
.
. *Rolling window program
. scalar drop _all

. quietly forval w=48(12)180 {

. scalar list // list the RMSE and min and max obs for each window width
RWmaxobs180 =        180
RWminobs180 =        180
 RWrmse180 =  .00406522
RWmaxobs168 =        168
```

```
RWminobs168 =        168
 RWrmse168 =  .00406959
RWmaxobs156 =        156
RWminobs156 =        156
 RWrmse156 =  .00406944
RWmaxobs144 =        144
RWminobs144 =        144
 RWrmse144 =  .00416772
RWmaxobs132 =        132
RWminobs132 =        132
 RWrmse132 =  .00406955
RWmaxobs120 =        120
RWminobs120 =        120
 RWrmse120 =  .00399029
RWmaxobs108 =        108
RWminobs108 =        108
 RWrmse108 =  .00384776
RWmaxobs96 =          96
RWminobs96 =          96
  RWrmse96 =  .00373216
RWmaxobs84 =          84
RWminobs84 =          84
  RWrmse84 =  .00369406
RWmaxobs72 =          72
RWminobs72 =          72
  RWrmse72 =  .00384486
RWmaxobs60 =          60
RWminobs60 =          60
  RWrmse60 =  .00424954
RWmaxobs48 =          48
RWminobs48 =          48
  RWrmse48 =  .00505078

. *End of rolling window program
.
.
.
. ***************************
.
. *Five GSREG models
.
.
. *Rolling window program
. scalar drop _all

. quietly forval w=48(12)180 {

. scalar list // list the RMSE and min and max obs for each window width
RWmaxobs180 =        180
RWminobs180 =        180
 RWrmse180 =  .00385964
RWmaxobs168 =        168
RWminobs168 =        168
 RWrmse168 =  .00384515
RWmaxobs156 =        156
RWminobs156 =        156
 RWrmse156 =  .00382718
RWmaxobs144 =        144
RWminobs144 =        144
 RWrmse144 =  .00394805
RWmaxobs132 =        132
RWminobs132 =        132
 RWrmse132 =  .00386368
```

```
RWmaxobs120 =        120
RWminobs120 =        120
 RWrmse120 =  .00380051
RWmaxobs108 =        108
RWminobs108 =        108
 RWrmse108 =  .00370836
RWmaxobs96 =         96
RWminobs96 =         96
  RWrmse96 =  .00359017
RWmaxobs84 =         84
RWminobs84 =         84
  RWrmse84 =  .00358536
RWmaxobs72 =         72
RWminobs72 =         72
  RWrmse72 =  .00374057
RWmaxobs60 =         60
RWminobs60 =         60
  RWrmse60 =  .00406565
RWmaxobs48 =         48
RWminobs48 =         48
  RWrmse48 =  .00431512

. *End of rolling window program
.
.
.
.
. *Rolling window program
. scalar drop _all

. quietly forval w=48(12)180 {

. scalar list // list the RMSE and min and max obs for each window width
RWmaxobs180 =        180
RWminobs180 =        180
 RWrmse180 =   .0038773
RWmaxobs168 =        168
RWminobs168 =        168
 RWrmse168 =  .00387424
RWmaxobs156 =        156
RWminobs156 =        156
 RWrmse156 =  .00385154
RWmaxobs144 =        144
RWminobs144 =        144
 RWrmse144 =  .00397404
RWmaxobs132 =        132
RWminobs132 =        132
 RWrmse132 =  .00388264
RWmaxobs120 =        120
RWminobs120 =        120
 RWrmse120 =  .00378879
RWmaxobs108 =        108
RWminobs108 =        108
 RWrmse108 =  .00367789
RWmaxobs96 =         96
RWminobs96 =         96
  RWrmse96 =  .00357342
RWmaxobs84 =         84
RWminobs84 =         84
  RWrmse84 =   .0035722
RWmaxobs72 =         72
RWminobs72 =         72
  RWrmse72 =   .0037286
```

```
RWmaxobs60 =         60
RWminobs60 =         60
  RWrmse60 =  .00402898
RWmaxobs48 =         48
RWminobs48 =         48
  RWrmse48 =  .00439626

. *End of rolling window program
.
.
.
.
. *Rolling window program
. scalar drop _all

. quietly forval w=48(12)180 {

. scalar list // list the RMSE and min and max obs for each window width
RWmaxobs180 =        180
RWminobs180 =        180
 RWrmse180 =  .00387455
RWmaxobs168 =        168
RWminobs168 =        168
 RWrmse168 =  .00387169
RWmaxobs156 =        156
RWminobs156 =        156
 RWrmse156 =  .00385123
RWmaxobs144 =        144
RWminobs144 =        144
 RWrmse144 =  .00397518
RWmaxobs132 =        132
RWminobs132 =        132
 RWrmse132 =  .00388061
RWmaxobs120 =        120
RWminobs120 =        120
 RWrmse120 =   .0037861
RWmaxobs108 =        108
RWminobs108 =        108
 RWrmse108 =  .00368681
RWmaxobs96 =         96
RWminobs96 =         96
  RWrmse96 =  .00358762
RWmaxobs84 =         84
RWminobs84 =         84
  RWrmse84 =  .00359497
RWmaxobs72 =         72
RWminobs72 =         72
  RWrmse72 =  .00374231
RWmaxobs60 =         60
RWminobs60 =         60
  RWrmse60 =  .00406703
RWmaxobs48 =         48
RWminobs48 =         48
  RWrmse48 =  .00449081

. *End of rolling window program
.
.
.
.
. *Rolling window program
. scalar drop _all
```

```
. quietly forval w=48(12)180 {

. scalar list // list the RMSE and min and max obs for each window width
RWmaxobs180 =        180
RWminobs180 =        180
 RWrmse180 =  .00392161
RWmaxobs168 =        168
RWminobs168 =        168
 RWrmse168 =  .00391121
RWmaxobs156 =        156
RWminobs156 =        156
 RWrmse156 =  .00388805
RWmaxobs144 =        144
RWminobs144 =        144
 RWrmse144 =  .00402227
RWmaxobs132 =        132
RWminobs132 =        132
 RWrmse132 =  .00396475
RWmaxobs120 =        120
RWminobs120 =        120
 RWrmse120 =  .00393111
RWmaxobs108 =        108
RWminobs108 =        108
 RWrmse108 =  .00380615
RWmaxobs96 =         96
RWminobs96 =         96
  RWrmse96 =  .00360662
RWmaxobs84 =         84
RWminobs84 =         84
  RWrmse84 =  .00357853
RWmaxobs72 =         72
RWminobs72 =         72
  RWrmse72 =  .00374813
RWmaxobs60 =         60
RWminobs60 =         60
  RWrmse60 =  .00407029
RWmaxobs48 =         48
RWminobs48 =         48
  RWrmse48 =  .00428177

. *End of rolling window program
.
.
.
.
. *Rolling window program
. scalar drop _all

. quietly forval w=48(12)180 {

. scalar list // list the RMSE and min and max obs for each window width
RWmaxobs180 =        180
RWminobs180 =        180
 RWrmse180 =  .00388793
RWmaxobs168 =        168
RWminobs168 =        168
 RWrmse168 =  .00387754
RWmaxobs156 =        156
RWminobs156 =        156
 RWrmse156 =  .00386765
RWmaxobs144 =        144
RWminobs144 =        144
 RWrmse144 =  .00397908
```

```
RWmaxobs132 =        132
RWminobs132 =        132
 RWrmse132 =  .00388438
RWmaxobs120 =        120
RWminobs120 =        120
 RWrmse120 =  .00379984
RWmaxobs108 =        108
RWminobs108 =        108
 RWrmse108 =  .00369883
RWmaxobs96 =          96
RWminobs96 =          96
  RWrmse96 =  .00360272
RWmaxobs84 =          84
RWminobs84 =          84
  RWrmse84 =  .00361184
RWmaxobs72 =          72
RWminobs72 =          72
  RWrmse72 =  .00376642
RWmaxobs60 =          60
RWminobs60 =          60
  RWrmse60 =  .00410608
RWmaxobs48 =          48
RWminobs48 =          48
  RWrmse48 =  .00450792

. *End of rolling window program
.
.
. */
.
. /* GSREG Model 2 at 8 years is the best, by a hair, on RWRMSE.
> Could easily choose GSREG 1, 4, or 7 too.
> Run Rolling Window again, just for w=96, for GSREG 2
> Don't clear for next loop,  to get info on this model. */
.
. *Rolling window program
. scalar drop _all

. gen pred=. // out of sample prediction
(361 missing values generated)

. gen nobs=. // number of observations in the window for each forecast point
(361 missing values generated)

.         quietly forval t=481/719 {

. gen res=d.lnflnonfarm-pred
(122 missing values generated)

. gen errsq=res^2 // generating squared errors
(122 missing values generated)

. summ errsq // getting the mean of the squared errors

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       errsq |        239    .0000155    .0000437   1.00e-12   .0004544

. scalar RWrmse96=r(mean)^.5 // getting the rmse for window width i

. summ nobs // getting min and max obs used

    Variable |        Obs        Mean    Std. Dev.       Min        Max
```

```
-------------+----------------------------------------------------------
        nobs |        239          96           0          96          96

. scalar RWminobs96=r(min) // min obs used in the window width

. scalar RWmaxobs96=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
RWmaxobs96 =         96
RWminobs96 =         96
  RWrmse96 =  .00393691

. *End of rolling window program
.
.
. *Forecast from selected model
.
. reg d.lnflnonfarm l(3,4,12,24)d.lnflnonfarm l(4)d.lnusepr ///
>                 md2 md3 md4 md5 md6 md7 md8 md9 md10 md11 md12 ///
>                 if tin(2012m1,2019m12)

      Source |       SS           df       MS      Number of obs   =        96
-------------+----------------------------------   F(16, 79)       =     43.44
       Model |  .007261762        16   .00045386   Prob > F        =    0.0000
    Residual |    .0008253        79  .000010447   R-squared       =    0.8979
-------------+----------------------------------   Adj R-squared   =    0.8773
       Total |  .008087062        95  .000085127   Root MSE        =    .00323

------------------------------------------------------------------------------
D.           |
 lnflnonfarm |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
 lnflnonfarm |
        L3D. |  -.0113479   .1164495    -0.10   0.923    -.2431348    .2204391
        L4D. |   .0160473   .1189138     0.13   0.893    -.2206446    .2527392
       L12D. |  -.1600824   .1073525    -1.49   0.140    -.3737621    .0535974
       L24D. |  -.0263326   .1050477    -0.25   0.803    -.2354249    .1827596
             |
     lnusepr |
        L4D. |   .2154534   .1643551     1.31   0.194    -.1116873    .5425941
             |
         md2 |   .0278338   .0043228     6.44   0.000     .0192294    .0364381
         md3 |   .0247999   .0040589     6.11   0.000     .0167209    .0328788
         md4 |   .0208837   .0046251     4.52   0.000     .0116778    .0300897
         md5 |   .0194558    .004109     4.73   0.000     .0112771    .0276345
         md6 |   .0020004   .0021337     0.94   0.351    -.0022466    .0062474
         md7 |   .0125271   .0026549     4.72   0.000     .0072426    .0178115
         md8 |   .0321801   .0049768     6.47   0.000     .0222741    .0420861
         md9 |   .0162684   .0040677     4.00   0.000     .0081719     .024365
        md10 |   .0317492   .0054092     5.87   0.000     .0209825    .0425159
        md11 |   .0314892   .0046256     6.81   0.000     .0222822    .0406962
        md12 |   .0233693   .0040487     5.77   0.000     .0153105    .0314281
       _cons |  -.0176253   .0030979    -5.69   0.000    -.0237914   -.0114591
------------------------------------------------------------------------------

. predict temp if date==tm(2020m1)
(option xb assumed; fitted values)
(360 missing values generated)
. replace pred=temp if date==tm(2020m1)
(1 real change made)
.
```

```
. *Empirical forecast and interval
. gen expres=exp(res)
(122 missing values generated)

. summ expres

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      expres |        239     .999795    .0039382   .9789086   1.019806

. gen epy=exp(l.lnflnonfarm+pred)*r(mean)
(121 missing values generated)

. _pctile res, percentiles(2.5,97.5)

. gen eub=epy*exp(r(r2))
(121 missing values generated)

. gen elb=epy*exp(r(r1))
(121 missing values generated)
.
. twoway (scatter fl_nonfarm date if tin(2017m1,2019m12) , m(Oh) ) ///
>         (tsline epy eub elb if tin(2017m1,2020m1) , ///
>                 lpattern(solid dash dash) lcolor(black gs10 gs10) ) , ///
>         saving(ps5_fcst, replace) scheme(s1mono) ylabel(,grid) xtitle("") ///
>         legend(label(1 "Nonfarm Employment") label(2 "Forecast") ///
>                 label(3 "95% Upper Bound") label(4 "95% Lower Bound") ) ///
>         title("Florida Nonfarm Employment" "One Month Ahead Forecast") ///
>         note("1) All forecasts are out of sample based on a 96 month rolling
window." ///
>         "2) Inteval based on percentiles 2.5 and 97.5 of the empirical forecast
error distribution." ///
>         "3) Predictors are lags 3, 4, 12, 24 of nonfarm employment and lag 4 of the
US emp:pop ratio." )
(file ps5_fcst.gph saved)

.
. graph export ps5empfcst.emf, replace
(file C:\Users\jdewey\Documents\A S20 Time Series\Problem Sets\ps5empfcst.emf written
in Enhanced Metafile format)
.
. list epy eub elb if date==tm(2020m1)

     +--------------------------------+
     |      epy        eub        elb |
     |--------------------------------|
361. | 9070.723   9146.527   8996.228 |
     +--------------------------------+
.
. *Normal forecast and interval
. gen npy=exp(l.lnflnonfarm+pred+(RWrmse96^2)/2)
(121 missing values generated)

. gen nub=npy*exp(1.96*RWrmse96)
(121 missing values generated)

. gen nlb=npy/exp(1.96*RWrmse96)
(121 missing values generated)

.
. twoway (scatter fl_nonfarm date if tin(2017m1,2019m12) , m(Oh) ) ///
>         (tsline npy nub nlb if tin(2017m1,2020m1) , ///
>                 lpattern(solid dash dash) lcolor(black gs10 gs10) ) , ///
```

```
>         saving(ps5_fcst, replace) scheme(s1mono) ylabel(,grid) xtitle("") ///
>         legend(label(1 "Nonfarm Employment") label(2 "Forecast") ///
>                 label(3 "95% Upper Bound") label(4 "95% Lower Bound") ) ///
>         title("Florida Nonfarm Employment" "One Month Ahead Forecast") ///
>         note("1) All forecasts are out of sample based on a 96 month rolling
window." ///
>         "2) Inteval based on percentiles +-1.95 RMMSE from the rolling window
procedure." ///
>         "3) Predictors are lags 3, 4, 12, 24 of nonfarm employment and lag 4 of the
US emp:pop ratio." )
(file ps5_fcst.gph saved)

.
. graph export ps5normfcst.emf, replace
(file C:\Users\jdewey\Documents\A S20 Time Series\Problem Sets\ps5normfcst.emf written
in Enhanced Metafile format)

.
. list npy nub nlb if date==tm(2020m1)

     +--------------------------------+
     |      npy        nub        nlb |
     |--------------------------------|
361. | 9072.653   9142.932   9002.915 |
     +--------------------------------+

.
.
. hist res, frac normal scheme(s1mono)  ///
>         title("Empirical Forecast Error Distribution") ///
>         xtitle("") note("For 96 month rolling window forecasts.")
(bin=15, start=-.02131703, width=.0027286)

. graph export ps5errdist.emf , replace
(file C:\Users\jdewey\Documents\A S20 Time Series\Problem Sets\ps5errdist.emf written
in Enhanced Metafile format)
.
. summ res

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         res |        239   -.0002128    .0039394   -.021317    .019612

. gen nres=(res-r(mean))/r(sd)
(122 missing values generated)

. qnorm nres, scheme(s1mono)  ///
>         title("Quantile-Normal Plot of Forecast Error") ///
>         xtitle("Inverse Standard Normal of Residual Percentile") ///
>         ytitle("Residual Z-Score") ///
>         xlabel(-6(2)4,grid) ylabel(-6(2)4,grid) ///
>         note("For 96 month rolling window forecasts.")

. graph export ps5qnorm.emf , replace
(file C:\Users\jdewey\Documents\A S20 Time Series\Problem Sets\ps5qnorm.emf written in
Enhanced Metafile format)
.
. clear

. log close
  log type:  smcl
 closed on:  10 Apr 2020, 18:21:52
-------------------------------------------------------------------------------------
```