Time Series Modeling and Forecasting **Problem Set 6 - Solution**

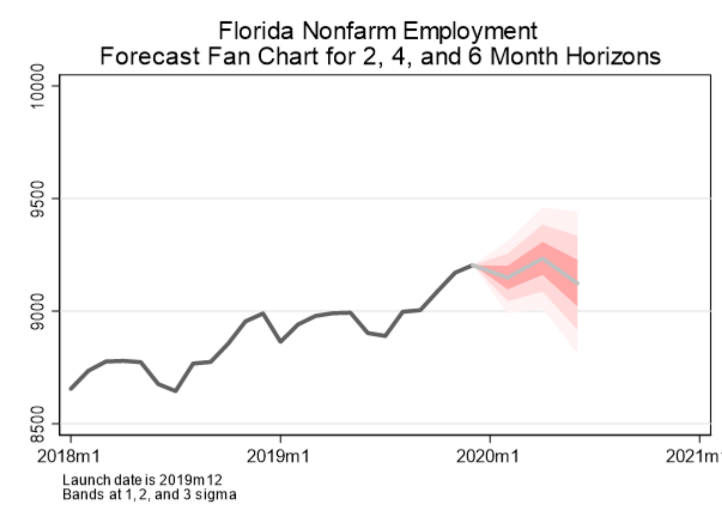
1. <u>One at a time forecasts multiple periods ahead</u>.

An extensive search using GSREG, selecting based on AIC, BIC, and out of sample (last 24 months) RMSE was used to select three candidate models for further analysis using rolling window estimation. The following was near the top for all horizons:

reg d.lnflnonfarm d.lnflnonfarm(6,9,12,24) l(6,9)d.lnusepr m2-m12

See the comments in the do file for more detail if desired. Therefore, I just used that model for all three horizons. The rolling window procedure showed that a window of 6 years, 72 months, resulted in the smallest forecast rmse.

The fanchart to the right and the table below shows the forecast and the normal forecast intervals at 1, 2, and 3 sigma (forecast error rmse), based on the most recent 72 months of data. The-3 sigma interval corresponds to a 99.7% CI under the assumption the errors are normal, and such an interval would contain the actual outcome at least 89% of the time regardless of the underlying distribution. Hence, regardless of the actual distribution, it is unlikely employment will lie outside of the 3-sigma band.

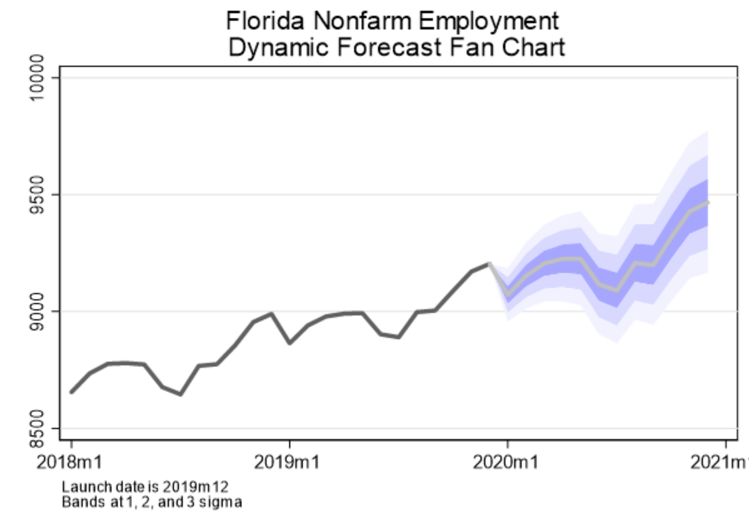


Florida Nonfarm Employment Forecast: February, April, and June 2020

Month of 2020	Interval Lower Bound μ-3σ	μ-2σ	μ-σ	Point Forecast	Interval Upper Bound μ+σ	μ+2σ	μ+3σ
February	8.990	9.042	9.095	9.149	9.202	9.256	9.311
April	9.013	9.086	9.160	9.234	9.309	9.384	9.460
June	8.814	8.916	9.019	9.123	9.229	9.336	9.444

2. <u>Dynamic Autoregressive Forecast</u>. I used GSREG to search over a large possible number of AR models to predict one period ahead, using all subsets of lags 1-12 and 24. Three of the top performing models were explored ore rigorously with the rolling window procedure. Of them, the best was the following, over a window of 96 months, or eight years:

arima d.lnflnonfarm m2-m12, ar(3,9,12)



The resulting dynamic forecast is summarized in the fan chart and table below. More details are in the comments in the do file.

Monthly Forecasts of 2020 Florida Nonfarm Employment

Month of 2020	Interval Lower Bound μ-3σ	μ-2σ	μ-σ	Point Forecast	Interval Upper Bound μ+σ	μ+2σ	μ+3σ
January	8.957	8.995	9.033	9.071	9.109	9.147	9.186
February	9.013	9.060	9.106	9.153	9.200	9.248	9.295
March	9.043	9.097	9.151	9.206	9.260	9.315	9.371
April	9.044	9.104	9.165	9.226	9.287	9.349	9.411
May	9.027	9.093	9.159	9.226	9.293	9.361	9.429
June	8.905	8.975	9.046	9.117	9.189	9.261	9.334
July	8.864	8.939	9.014	9.090	9.167	9.244	9.322
August	8.966	9.046	9.127	9.208	9.291	9.374	9.457
September	8.943	9.027	9.112	9.198	9.285	9.373	9.461
October	9.047	9.136	9.227	9.318	9.410	9.503	9.597
November	9.142	9.236	9.332	9.428	9.526	9.624	9.723
December	9.167	9.266	9.365	9.466	9.568	9.671	9.775

3. Compare and Discuss. The results are quite similar, as shown in the table below.

Forecast Comparison

Month of 2020	Point Forecast		μ-3σ		μ+3σ	
	Direct	Dynamic	Direct	Dynamic	Direct	Dynamic
February	9.149	9.153	8.990	9.013	9.311	9.295
April	9.234	9.226	9.013	9.044	9.460	9.411
June	9.123	9.117	8.814	8.905	9.444	9.334

Since the dynamic forecasting method is more subject to compounding errors, we would expect the forecast interval to be wider, but it is not. From the direct estimate of the 6 month change, the RMSE (for the log change) was 0.0115. From the dynamic model, 6 months out it was 0.0079 (adding the successive MSEs and taking the square root). This seems too low! What follows was not really required for full credit, but is worth thinking through carefully.

I ran the rolling window procedure for the entire sample period, using the OLS version of the model, to get the RMSE for forecasting the log change out one period. The RMSE was 0.0048. So, 6 months out, rmse should be about 0.0048*6^.5=0.0118. This is higher than 0.0115 from estimating the 6 month change directly, as it should be. And that does not even factor in what would happen if we ran through this 6 times dynamically with Rolling window, which would likely give a yet larger RMSE due to compounding the modeling error component, not just the residual component. The true RMSE would be even higher.

So, at the 6-month mark, the dynamic approach is underestimating the 95% ci upper bound by a factor BIGGER than exp(2*(0.0118-0.0079)) = 1.0078, and the lower interval lower by a factor smaller than 1/1.0078. With employment at the end of 2019 at 9.204 M, the CI upper bound should be at least 0.072M higher and the lower bound at least 0.072 lower.

Bottom line: is the CI is quite a bit too small with the dynamic technique! A quick dynamic forecast is fine if the stakes are low, and the fanchart is very suggestive. However, if the stakes are high, take the time to find the best model for the times of interest and use rolling window estimation and validation.

Appendix A: Log File

```
*Problem Set 6 Solution

clear
set more off
cd "C:\Users\jdewey\Documents\A S20 Time Series\Problem Sets\"
log using "Problem Set 6 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(12)
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)

*Note, we have already examined PAC, AC, etc and decided to difference

*generate variables for gsreg and multiple steps out

gen dlnflnonfarm=d.lnflnonfarm
gen dh2lnflnonfarm=lnflnonfarm-l2.lnflnonfarm
gen dh4lnflnonfarm=lnflnonfarm-l4.lnflnonfarm
gen dh6lnflnonfarm=lnflnonfarm-l6.lnflnonfarm

gen ldlnflnonfarm=ld.lnflnonfarm
gen l2dlnflnonfarm=l2d.lnflnonfarm
gen l3dlnflnonfarm=l3d.lnflnonfarm
gen l4dlnflnonfarm=l4d.lnflnonfarm
gen l6dlnflnonfarm=l6d.lnflnonfarm
gen l9dlnflnonfarm=l9d.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 l6dlnusepr=l6d.lnusepr
gen l9dlnusepr=l9d.lnusepr
gen l12dlnusepr=l12d.lnusepr

gen ldlnflbp=ld.lnflbp
gen l2dlnflbp=l2d.lnflbp
```

```
gen l3dlnflbp=l3d.lnflbp
gen l4dlnflbp=l4d.lnflbp
gen l6dlnflbp=l6d.lnflbp
gen l9dlnflbp=l9d.lnflbp
gen l12dlnflbp=l12d.lnflbp


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

/* From problem set 5, we know labor force was not very useful for one
step forecasts. So, I dropped those. I added lags 6 and 9 to consideration
given how long out we are forecasting.
I limited GSREG to combinations of up to 6 for time. Further ahead forecasts
usually, but not necessarily always, call for more parsimonious models. */


/*
gsreg dh2lnflnonfarm l2dlnflnonfarm l3dlnflnonfarm l4dlnflnonfarm ///
      l6dlnflnonfarm l9dlnflnonfarm l12dlnflnonfarm l24dlnflnonfarm ///
      l2dlnusepr l3dlnusepr l4dlnusepr l6dlnusepr l9dlnusepr l12dlnusepr ///
      l2dlnflbp l3dlnflbp l4dlnflbp l6dlnflbp l9dlnflbp l12dlnflbp , ///
      nocount results(ps6modelsh2.dta) replace ///
    fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) ncomb(1,6) ///
      aic outsample(24) nindex( -1 aic -1 bic -1 rmse_out) samesample

gsreg dh4lnflnonfarm l4dlnflnonfarm ///
      l6dlnflnonfarm l9dlnflnonfarm l12dlnflnonfarm l24dlnflnonfarm ///
      l4dlnusepr l6dlnusepr l9dlnusepr l12dlnusepr ///
      l4dlnflbp l6dlnflbp l9dlnflbp l12dlnflbp, ///
      nocount results(ps6modelsh4.dta) replace ///
    fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) ncomb(1,6) ///
      aic outsample(24) nindex( -1 aic -1 bic -1 rmse_out) samesample

gsreg dh6lnflnonfarm ///
      l6dlnflnonfarm l9dlnflnonfarm l12dlnflnonfarm l24dlnflnonfarm ///
      l6dlnusepr l9dlnusepr l12dlnusepr ///
      l6dlnflbp l9dlnflbp l12dlnflbp, ///
      nocount results(ps6modelsh6.dta) replace ///
    fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) ncomb(1,6) ///
      aic outsample(24) nindex( -1 aic -1 bic -1 rmse_out) samesample

*/




/*
Checking the gsreg output I selected these:

For H=2
1) dh2lnflnonfarm d.lnflnonfarm(6,9,12,24) l(6,9)d.lnusepr
2) dh2lnflnonfarm d.lnflnonfarm(6,9,12,24) l(6)d.lnusepr
15) dh2lnflnonfarm d.lnflnonfarm(6,9,12,24)

For H=4
1) dh4lnflnonfarm d.lnflnonfarm(4,6,9,12) l(6,9)d.lnusepr
2) dh4lnflnonfarm d.lnflnonfarm(6,9,12,24) l(6,9)d.lnusepr
32) dh4lnflnonfarm d.lnflnonfarm(6,12) l(6,9)d.lnusepr
```

```
For H=6
1) dh6lnflnonfarm d.lnflnonfarm(9,12,24) l(6,9,12)d.lnusepr
2) dh6lnflnonfarm d.lnflnonfarm(9,12) l(6,9,12)d.lnusepr
15) dh6lnflnonfarm d.lnflnonfarm(6,9,12,24) l(6,9)d.lnusepr


HOWEVER, due to strong agreement on l(6,9)d.lnusepr and that lag 2 of the AR
term never appears and lag 4 is only on one of the models, and lag 24 is
in most, I will just run
      reg d.lnflnonfarm(6,9,12,24) l(6,9)d.lnusepr m2-m12
for all horisons to pick the window

*/

/*
***********************
*For H=2

*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 dh2lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
            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-dh2lnflnonfarm)^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
```

```
*For H=4

*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 dh4lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
		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-dh4lnflnonfarm)^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


*For H=6

*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 dh6lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
            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-dh6lnflnonfarm)^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


*/

/* 72 months, or 6 years, performs best for all
Run Rolling Window again, just for w=72,
Don't clear for next loop,  to get info on this model for each H. */

scalar drop _all

*H=2
*Rolling window program
gen pred2=. // out of sample prediction
gen nobs2=. // number of observations in the window for each forecast point

      quietly forval t=457/719 {
      /* 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'- 72 // 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 dh2lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
            m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
            if date>=wstart & date<=wend // restricts the model to the window
      replace nobs2=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 res2=dh2lnflnonfarm-pred2
gen err2sq=res2^2 // generating squared errors
summ err2sq // getting the mean of the squared errors
scalar rwrmseh2=r(mean)^.5 // getting the rmse for window width i
summ nobs2 // getting min and max obs used
scalar rwminobsh2=r(min) // min obs used in the window width
scalar rwmaxobsh2=r(max) // max obs used in the window width
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program


*H=4
*Rolling window program
gen pred4=. // out of sample prediction
gen nobs4=. // number of observations in the window for each forecast point

      quietly forval t=457/719 {
      /* 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'- 72 // 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 dh4lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
            m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
            if date>=wstart & date<=wend // restricts the model to the window
      replace nobs4=e(N) if date==`t' // number of observations used
      predict ptemp // temporary predicted values
      replace pred4=ptemp if date==`t' // saving the single forecast value
      drop ptemp wstart wend // clear these to prepare for the next loop
      }
gen res4=dh4lnflnonfarm-pred4
gen err4sq=res4^2 // generating squared errors
summ err4sq // getting the mean of the squared errors
scalar rwrmseh4=r(mean)^.5 // getting the rmse for window width i
summ nobs4 // getting min and max obs used
scalar rwminobsh4=r(min) // min obs used in the window width
scalar rwmaxobsh4=r(max) // max obs used in the window width
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program


*H=6
*Rolling window program
gen pred6=. // out of sample prediction
gen nobs6=. // number of observations in the window for each forecast point

      quietly forval t=457/719 {
```

```
      /* 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'- 72 // 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 dh6lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
            m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
            if date>=wstart & date<=wend // restricts the model to the window
      replace nobs6=e(N) if date==`t' // number of observations used
      predict ptemp // temporary predicted values
      replace pred6=ptemp if date==`t' // saving the single forecast value
      drop ptemp wstart wend // clear these to prepare for the next loop
      }
gen res6=dh6lnflnonfarm-pred6
gen err6sq=res6^2 // generating squared errors
summ err6sq // getting the mean of the squared errors
scalar rwrmseh6=r(mean)^.5 // getting the rmse for window width i
summ nobs6 // getting min and max obs used
scalar rwminobsh6=r(min) // min obs used in the window width
scalar rwmaxobsh6=r(max) // max obs used in the window width
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program


*Forecast from selected model

reg dh2lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
            m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m1,2019m12)
predict pd if date==tm(2020m2)
gen pflnonfarm=exp((rwrmseh2^2)/2)*exp(l2.lnflnonfarm+pd) if date==tm(2020m2)
gen ub1=exp((rwrmseh2^2)/2)*exp(l2.lnflnonfarm+pd+1*rwrmseh2) if
date==tm(2020m2)
gen lb1=exp((rwrmseh2^2)/2)*exp(l2.lnflnonfarm+pd-1*rwrmseh2) if
date==tm(2020m2)
gen ub2=exp((rwrmseh2^2)/2)*exp(l2.lnflnonfarm+pd+2*rwrmseh2) if
date==tm(2020m2)
gen lb2=exp((rwrmseh2^2)/2)*exp(l2.lnflnonfarm+pd-2*rwrmseh2) if
date==tm(2020m2)
gen ub3=exp((rwrmseh2^2)/2)*exp(l2.lnflnonfarm+pd+3*rwrmseh2) if
date==tm(2020m2)
gen lb3=exp((rwrmseh2^2)/2)*exp(l2.lnflnonfarm+pd-3*rwrmseh2) if
date==tm(2020m2)
drop pd

reg dh4lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
            m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m1,2019m12)
predict pd if date==tm(2020m4)
replace pflnonfarm=exp((rwrmseh4^2)/2)*exp(l4.lnflnonfarm+pd) if
date==tm(2020m4)
replace ub1=exp((rwrmseh4^2)/2)*exp(l4.lnflnonfarm+pd+1*rwrmseh4) if
date==tm(2020m4)
```

```
replace lb1=exp((rwrmseh4^2)/2)*exp(l4.lnflnonfarm+pd-1*rwrmseh4) if
date==tm(2020m4)
replace ub2=exp((rwrmseh4^2)/2)*exp(l4.lnflnonfarm+pd+2*rwrmseh4) if
date==tm(2020m4)
replace lb2=exp((rwrmseh4^2)/2)*exp(l4.lnflnonfarm+pd-2*rwrmseh4) if
date==tm(2020m4)
replace ub3=exp((rwrmseh4^2)/2)*exp(l4.lnflnonfarm+pd+3*rwrmseh4) if
date==tm(2020m4)
replace lb3=exp((rwrmseh4^2)/2)*exp(l4.lnflnonfarm+pd-3*rwrmseh4) if
date==tm(2020m4)
drop pd

reg dh6lnflnonfarm l(6,9,12,24)d.lnflnonfarm l(6,9)d.lnusepr ///
            m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 if tin(2014m1,2019m12)
predict pd if date==tm(2020m6)
replace pflnonfarm=exp((rwrmseh6^2)/2)*exp(l6.lnflnonfarm+pd) if
date==tm(2020m6)
replace ub1=exp((rwrmseh6^2)/2)*exp(l6.lnflnonfarm+pd+1*rwrmseh6) if
date==tm(2020m6)
replace lb1=exp((rwrmseh6^2)/2)*exp(l6.lnflnonfarm+pd-1*rwrmseh6) if
date==tm(2020m6)
replace ub2=exp((rwrmseh6^2)/2)*exp(l6.lnflnonfarm+pd+2*rwrmseh6) if
date==tm(2020m6)
replace lb2=exp((rwrmseh6^2)/2)*exp(l6.lnflnonfarm+pd-2*rwrmseh6) if
date==tm(2020m6)
replace ub3=exp((rwrmseh6^2)/2)*exp(l6.lnflnonfarm+pd+3*rwrmseh6) if
date==tm(2020m6)
replace lb3=exp((rwrmseh6^2)/2)*exp(l6.lnflnonfarm+pd-3*rwrmseh6) if
date==tm(2020m6)
drop pd


replace pflnonfarm=fl_nonfarm if date==tm(2019m12)
replace ub1=fl_nonfarm if date==tm(2019m12)
replace ub2=fl_nonfarm if date==tm(2019m12)
replace ub3=fl_nonfarm if date==tm(2019m12)
replace lb1=fl_nonfarm if date==tm(2019m12)
replace lb2=fl_nonfarm if date==tm(2019m12)
replace lb3=fl_nonfarm if date==tm(2019m12)


*Table
list date pflnonfarm lb3 lb2 lb1 ub1 ub2 ub3 if tin(2020m1,2020m12)

*Fan Charts

twoway (tsrline ub3 ub2 if tin(2018m1,2020m12), ///
      recast(rarea) fcolor(red) fintensity(5) lwidth(none) ) ///
      (tsrline ub2 ub1 if tin(2018m1,2020m12), ///
      recast(rarea) fcolor(red) fintensity(15) lwidth(none) ) ///
      (tsrline ub1 pflnonfarm if tin(2018m1,2020m12), ///
      recast(rarea) fcolor(red) fintensity(35) lwidth(none) ) ///
      (tsrline pflnonfarm lb1 if tin(2018m1,2020m12), ///
      recast(rarea) fcolor(red) fintensity(35) lwidth(none) ) ///
      (tsrline lb1 lb2 if tin(2018m1,2020m12), ///
      recast(rarea) fcolor(red) fintensity(15) lwidth(none) ) ///
      (tsrline lb2 lb3 if tin(2018m1,2020m12), ///
```

```
      recast(rarea) fcolor(red) fintensity(5) lwidth(none) ) ///
      (tsline fl_nonfarm pflnonfarm if tin(2018m1,2020m12) , ///
      lcolor(gs6 gs12) lwidth(thick thick) ), scheme(s1mono) legend(off) ///
      title("Florida Nonfarm Employment"  ///
      "Forecast Fan Chart for 2, 4, and 6 Month Horizons") legend(off) ///
      ylab(8500(500)10000) xtitle("") ylabel(,grid)  ///
      note("Launch date is 2019m12" "Bands at 1, 2, and 3 sigma")

      graph export "Fan Chart One at a time.emf", replace


*Dynamic Forecast

*Using GSREG to select an AR model. Considering lags 1-12 and 24

/*
gsreg dlnflnonfarm  l24dlnflnonfarm, dlags(1/12) ///
      nocount results(ps6modelsar.dta) replace ///
    fix(m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12) ncomb(1,6) ///
      aic outsample(24) nindex( -1 aic -1 bic -1 rmse_out) samesample
*/

/*
#1: d.lnflnonfarm l(3,9,10,12)d.lnflnonrarm
#2: d.lnflnonfarm l(3,6,9,10,12)d.lnflnonrarm
#3: d.lnflnonfarm l(3,9,12)d.lnflnonrarm
*/


************************

*GSREG 1
*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(3,9,10,12)d.lnflnonfarm ///
            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.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 2
*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(3,9,10,12)d.lnflnonfarm ///
            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.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 3
```

```
*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(3,9,12)d.lnflnonfarm ///
            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.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 3 for 96 months seems best

arima d.lnflnonfarm m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
      if tin(2012m11,2019m12) , ar(3,9,12)
predict pdlny , dynamic(mofd(tm(2019m12)))  // starts dynamics 2019m12
predict mse, mse // variance of forecast each period

gen py=fl_nonfarm if date==tm(2019m12) // last known point to start fan chart
replace py=l.py*exp(pdlny+mse/2) if date>tm(2019m12) // moves ahead
recursively

/*treating errors as realizations of independent shocks,
the varianve of a forecast at h is the sum of the variaces of all the prior
forecasts of d.lny (since we keep adding up) and the current one. So: */
gen totmse=mse if date==tm(2019m12)
replace totmse=l.totmse+mse if date>tm(2019m12)
gen rmsedyn=totmse^.5
```

```
*Use this for the fan chart bounds:
gen yub1=py*exp(totmse^.5)
gen yub2=py*exp(2*totmse^.5)
gen yub3=py*exp(3*totmse^.5)
gen ylb1=py/exp(totmse^.5)
gen ylb2=py/exp(2*totmse^.5)
gen ylb3=py/exp(3*totmse^.5)

replace yub1=fl_nonfarm if date==tm(2019m12)
replace yub2=fl_nonfarm if date==tm(2019m12)
replace yub3=fl_nonfarm if date==tm(2019m12)
replace ylb1=fl_nonfarm if date==tm(2019m12)
replace ylb2=fl_nonfarm if date==tm(2019m12)
replace ylb3=fl_nonfarm if date==tm(2019m12)

twoway (tsrline yub3 yub2 if tin(2018m12,), ///
      recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
      (tsrline yub2 yub1 if tin(2018m12,), ///
      recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
      (tsrline yub1 py if tin(2018m12,), ///
      recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
      (tsrline py ylb1 if tin(2018m12,), ///
      recast(rarea) fcolor(blue) fintensity(35) lwidth(none) ) ///
      (tsrline ylb1 ylb2 if tin(2018m12,), ///
      recast(rarea) fcolor(blue) fintensity(15) lwidth(none) ) ///
      (tsrline ylb2 ylb3 if tin(2018m12,), ///
      recast(rarea) fcolor(blue) fintensity(5) lwidth(none) ) ///
      (tsline fl_nonfarm py if tin(2018m1,) , ///
      lcolor(gs6 gs12) lwidth(thick thick) ), scheme(s1mono) legend(off) ///
      title("Florida Nonfarm Employment"  ///
      "Dynamic Forecast Fan Chart") legend(off) ///
      xtitle("") ylabel(,grid) ///
      note("Launch date is 2019m12" "Bands at 1, 2, and 3 sigma") ///

      graph export dynamicfanchart.emf , replace


list date pflnonfarm lb3 lb2 lb1 ub1 ub2 ub3 if tin(2020m1,2020m12)
list date py ylb3 ylb2 ylb1 yub1 yub2 yub3 if tin(2020m1,2020m12)
list date pflnonfarm py lb3 ylb3 ub3 yub3 if tin(2020m1,2020m12)


/*
From the direct estimate of the 6 month change, the RMSE was 0.0115.
From the dynamic model, 6 months out it was 0.0079.
This seems too low!

Not required, but it is interesting to compare the RWRMSE to the RMSE
In the dynamic model
Should the arima routine, not regression, to evaluate it
since we use it for the dynamic part
But, that will make this take A LOT LOT LOT longer
Use reg just to get estimate of 1st part RMSE
*/
```

```
*Rolling window program
gen predar=. // out of sample prediction
gen nobsar=. // 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  */
      arima d.lnflnonfarm l(3,9,12) m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
            if date>=wstart & date<=wend   // restricts to the window
      replace nobsar=e(N) if date==`t' // number of observations used
      predict ptemp // temporary predicted values
      replace predar=ptemp if date==`t' // saving the single forecast value
      drop ptemp wstart wend // clear these to prepare for the next loop
      }
gen resar=d.lnflnonfarm-predar
gen errarsq=resar^2 // generating squared errors
summ errarsq // getting the mean of the squared errors
scalar rwrmsear=r(mean)^.5 // getting the rmse for window width i
summ nobsar // getting min and max obs used
scalar rwminobshar=r(min) // min obs used in the window width
scalar rwmaxobshar=r(max) // max obs used in the window width
scalar list // list the RMSE and min and max obs for each window width
*End of rolling window program

summ fl_nonfarm if date==tm(2019m12)

/*
So, 6 months out, rmse should be about 0.0048*6^.5=0.0118.
This is higher than 0.0115 from estimating the 6 month change directy.
And, it does not even factor in what would happen if we ran through this
6 times dynamically with Rolling window, it is just from cross validatin
the one period forecast!
The true RMSE would be even higher.

So, at the 6 month mark, the dynamic approach is understimating the
95% ci upper bound should be higher by a factor BIGGER than
      exp(2*(0.0108-0.0079)) = 1.0078
And the lower interval lower by a factor smaller than 1/1.0078.
With employment at the end of 2019 at 9204, the CI should bemore than
144 wider (upper about 72 higher and lower about 72 lower) in June 2020.

Bottom line: is the CI is quite a bit too small with the dynamic mode!
```