```
*Time Series Final Project - Deliverable 1 + 2 + 3

/*The project is to forecast the March non-seasonally adjusted estimates of average weekly earnings
and total employment for private employers (total private) for a Florida metropolitan statistical
area */

clear
set more off
cd "C:\Users\Sydney\OneDrive - Florida Polytechnic University\Spring 2022\STA4853 - Time
Series\Project\final_update_proj_data_txt"

log using "FinalProject", replace

import delimited "final_update_proj_data_Monthly.txt"

*Generate a monthly date variable %tm
rename date datestring
generate datec=date(datestring, "YMD")
gen date=mofd(datec)
format date %tm
tsset date
generate quarter = quarter(datec)
generate monthly = month(datec)

*Rename variables
rename smu12367400500000001 TotalPriv
*All Employees: Total Private in Orlando-Kissimmee-Sanford, FL (MSA)

rename smu12367400500000002 WeeklyHrs
*Average Weekly Hours of All Employees: Total Private in Orlando-Kissimmee-Sanford, FL (MSA)

rename smu12367400500000003 HourlyEarn
*Average Hourly Earnings of All Employees: Total Private in Orlando-Kissimmee-Sanford, FL (MSA)

rename smu12367400500000011 WeeklyEarn
*Average Weekly Earnings of All Employees: Total Private in Orlando-Kissimmee-Sanford, FL (MSA)

*Natural log of variables
gen lnTotalPriv = ln(TotalPriv)
gen lnWeeklyHrs = ln(WeeklyHrs)
gen lnHourlyEarn = ln(HourlyEarn)
gen lnWeeklyEarn = ln(WeeklyEarn)

*Summary Statistics
summ TotalPriv WeeklyHrs HourlyEarn WeeklyEarn
summ lnTotalPriv lnWeeklyHrs lnHourlyEarn lnWeeklyEarn

*Tsline
tsline lnTotalPriv, saving(lnTotalPriv_ts, replace)
tsline lnWeeklyEarn, saving(lnWeeklyEarn_ts, replace)

graph combine lnTotalPriv_ts.gph lnWeeklyEarn_ts.gph, saving(lntsline, replace)
graph export "lntsline.png", replace

*AC and PAC
ac lnTotalPriv, saving(ac_lntotal, replace)
pac lnTotalPriv, saving(pac_lntotal, replace)

graph combine ac_lntotal.gph pac_lntotal.gph, saving(ac_pac_lntotal, replace)
graph export "ac_pac_lntotal.png", replace

ac lnWeeklyEarn, saving(ac_lnWeeklyE, replace)
pac lnWeeklyEarn, saving(pac_lnWeeklyE, replace)
```

```
63
64  graph combine ac_lnWeeklyE.gph pac_lnWeeklyE.gph, saving(ac_pac_lnWeeklyE, replace)
65  graph export "ac_pac_lnWeeklyE.png", replace
66
67  *Deliverable 2
68
69  *gen lags for vselect might change to 1/24
70  gen dlnTotalPriv=d.lnTotalPriv
71  quietly forvalues i=1/12 {
72      gen dlnTotalPrivl`i'=l`i'd.lnTotalPriv
73  }
74
75  gen dlnWeeklyEarn=d.lnWeeklyEarn
76  quietly forvalues i=1/12 {
77      gen dlnWeeklyEarnl`i'=l`i'd.lnWeeklyEarn
78  }
79
80  *Use vselect to estimate and evaluate some alt models for total priv and avg weekly earnings
81
82  vselect dlnTotalPriv dlnTotalPrivl* dlnWeeklyEarnl*, best
83
84  *For a reasonable set of models from vselect, also calculate LOOCV
85  /*
86  predictors for each model
87  2: l(1,2)dlnTotalPriv
88  3: l(1,2,12)dlnTotalPriv
89  4: l(1,2,12)dlnTotalPriv l(12)dlnWeeklyEarn
90  5: l(1,2,12)dlnTotalPriv l(1,2)dlnWeeklyEarn
91  6: l(1,2,9,12)dlnTotalPriv l(1,2)dlnWeeklyEarn
92
93  */
94  scalar drop _all
95  loocv reg d.lnTotalPriv l(1,2)d.lnTotalPriv
96  scalar define loormse2=r(rmse)
97
98  loocv reg d.lnTotalPriv l(1,2,12)d.lnTotalPriv
99  scalar define loormse3=r(rmse)
100
101 loocv reg d.lnTotalPriv l(1,2,12)d.lnTotalPriv l(12)d.lnWeeklyEarn
102 scalar define loormse4=r(rmse)
103
104 loocv reg d.lnTotalPriv l(1,2,12)d.lnTotalPriv l(1,2)d.lnWeeklyEarn
105 scalar define loormse5=r(rmse)
106
107 loocv reg d.lnTotalPriv l(1,2,9,12)d.lnTotalPriv l(1,2)d.lnWeeklyEarn
108 scalar define loormse6=r(rmse)
109
110 scalar list loormse2 loormse3 loormse4 loormse5 loormse6
111 *include a simple 12-lag AR model as benchmark
112 reg lnTotalPriv l(1/12).lnWeeklyEarn
113 reg d.lnTotalPriv l(1/12)d.lnTotalPriv l(1/12)d.lnWeeklyHrs i.month date
114
115
116 /* Deliverable 3
117 use the rolling window procedure to make final model selection, including selecting window width.
    Include a simple 12 lag AR only model as a benchmark. Clearly report the best window width and the
    resulting rolling window RMSE for this benchmark model. */
118
119 /*
120 1) The date of the first complete observation is t0.
121 2) The longest lag is L.
122 3) We difference.
123 */
```

```
124
125 summ date if l12d.lnWeeklyEarn~=. & l12d.lnTotalPriv~=.
126 *min 577 + 84 = 661
127
128 *Rolling window program 1 .01755403 window size 84
129 scalar drop _all
130 quietly forval w=12(12)84 {
131 /* w=small(inc)large
132 small is the smallest window
133 inc is the window size increment
134 large is the largest window.
135 (large-small)/inc must be an interger */
136 gen pred=. // out of sample prediction
137 gen nobs=. // number of observations in the window for each forecast point
138     forval t=661/745 {
139     /* t=first/last
140     first is the first date for which you want to make a forecast.
141     first-1 is the end date of the earliest window used to fit the model.
142     first-w, where w is the window width, is the date of the first
143     observation used to fit the model in the earliest window.
144     You must choose first so it is preceded by a full set of
145     lags for the model with the longest lag length to be estimated.
146     last is  the last observation to be forecast. */
147     gen wstart=`t'-`w' // fit window start date
148     gen wend=`t'-1 // fit window end date
149     /* Enter the regression command immediately below.
150     Leave the if statement intact to control the window  */
151     reg d.lnTotalPriv l(1,2)d.lnTotalPriv  ///
152         if date>=wstart & date<=wend // restricts the model to the window
153     replace nobs=e(N) if date==`t' // number of observations used
154     predict ptemp // temporary predicted values
155     replace pred=ptemp if date==`t' // saving the single forecast value
156     drop ptemp wstart wend // clear these to prepare for the next loop
157     }
158 gen errsq=(pred-d.lnWeeklyEarn)^2 // generating squared errors
159 summ errsq // getting the mean of the squared errors
160 scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
161 summ nobs // getting min and max obs used
162 scalar RWminobs`w'=r(min) // in obs used in the window width
163 scalar RWmaxobs`w'=r(max) // max obs used in the window width
164 drop errsq pred nobs // clearing for the next loop
165 }
166 scalar list // list the RMSE and min and max obs for each window width
167 *End of rolling window program
168
169
170 *Rolling window program 2 .02484131, 84
171 scalar drop _all
172 quietly forval w=12(12)84 {
173 /* w=small(inc)large
174 small is the smallest window
175 inc is the window size increment
176 large is the largest window.
177 (large-small)/inc must be an interger */
178 gen pred=. // out of sample prediction
179 gen nobs=. // number of observations in the window for each forecast point
180     forval t=661/745 {
181     /* t=first/last
182     first is the first date for which you want to make a forecast.
183     first-1 is the end date of the earliest window used to fit the model.
184     first-w, where w is the window width, is the date of the first
185     observation used to fit the model in the earliest window.
186     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.lnTotalPriv l(1,2,12)d.lnTotalPriv  ///
        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.lnWeeklyEarn)^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 3  .02494302, 84
scalar drop _all
quietly forval w=12(12)84 {
/* 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=661/745 {
    /* 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.lnTotalPriv l(1,2,12)d.lnTotalPriv l(12)d.lnWeeklyEarn ///
        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.lnWeeklyEarn)^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 4 .02378657, 84
scalar drop _all
quietly forval w=12(12)84 {
/* 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=661/745 {
    /* 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.lnTotalPriv l(1,2,12)d.lnTotalPriv l(1,2)d.lnWeeklyEarn ///
        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.lnWeeklyEarn)^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 5 .02375954, 84
scalar drop _all
quietly forval w=12(12)84 {
/* 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=661/745 {
    /* 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.
```

```
313       last is  the last observation to be forecast. */
314       gen wstart=`t'-`w' // fit window start date
315       gen wend=`t'-1 // fit window end date
316       /* Enter the regression command immediately below.
317       Leave the if statement intact to control the window  */
318       reg d.lnTotalPriv l(1,2,9,12)d.lnTotalPriv l(1,2)d.lnWeeklyEarn ///
319           if date>=wstart & date<=wend // restricts the model to the window
320       replace nobs=e(N) if date==`t' // number of observations used
321       predict ptemp // temporary predicted values
322       replace pred=ptemp if date==`t' // saving the single forecast value
323       drop ptemp wstart wend // clear these to prepare for the next loop
324       }
325   gen errsq=(pred-d.lnWeeklyEarn)^2 // generating squared errors
326   summ errsq // getting the mean of the squared errors
327   scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
328   summ nobs // getting min and max obs used
329   scalar RWminobs`w'=r(min) // in obs used in the window width
330   scalar RWmaxobs`w'=r(max) // max obs used in the window width
331   drop errsq pred nobs // clearing for the next loop
332   }
333   scalar list // list the RMSE and min and max obs for each window width
334
335   *End of rolling window program
336   /*
337   List all model RWrmse here:
338   model 1: .01755403
339   model 2: .02484131
340   model 3: .02450599
341   model 4: .02378657
342   model 5: .02375954
343
344   1: l(1,2)dlnTotalPriv
345   2: l(1,2,12)dlnTotalPriv
346   3: l(1,2,12)dlnTotalPriv l(12)dlnWeeklyEarn
347   4: l(1,2,12)dlnTotalPriv l(1,2)dlnWeeklyEarn
348   5: l(1,2,9,12)dlnTotalPriv l(1,2)dlnWeeklyEarn
349
350
351   Pick best model
352   Model 1 has the lowest rmse at .01755403 but models, 4 and 5 are the next lowest with .02378657 and
      .02375954 and include lags 1,2,12dlnTotalPriv and lags 1,2 dlnWeekly. Model one just has l(1,2)
      dlnTotalPriv. When looking at all models, model 1 has the best BIC, models 4 and 5 do not have much
      going for them.
353   */
354
355   *Make evaluation chart for TotalPriv with best model
356
357
358   *Rolling window program -- Inner Loop Only
359   *min is 577, best model performed with window width of: 84, 577 +84 =661 to 746
360
361   scalar drop _all
362   gen pred=. // out of sample prediction
363   gen nobs=. // number of observations in the window for each forecast point
364       quietly forval t=661/746 {
365       /* t=first/last
366       first is the first date for which you want to make a forecast.
367       first-1 is the end date of the earliest window used to fit the model.
368       first-w, where w is the window width, is the date of the first
369       observation used to fit the model in the earliest window.
370       You must choose first so it is preceded by a full set of
371       lags for the model with the longest lag length to be estimated.
372       last is  the last observation to be forecast. */
```

```
373       gen wstart=`t'-84 // fit window start date
374       gen wend=`t'-1 // fit window end date
375       /* Enter the regression command immediately below.
376       Leave the if statement intact to control the window  */
377       reg d.lnTotalPriv l(1,2)d.lnTotalPriv  ///
378           if date>=wstart & date<=wend // restricts the model to the window
379       replace nobs=e(N) if date==`t' // number of observations used
380       predict ptemp // temporary predicted values
381       replace pred=ptemp if date==`t' // saving the single forecast value
382       drop ptemp wstart wend // clear these to prepare for the next loop
383       }
384
385   **End of selected rolling window implementation
386
387
388   /*
389   *gen future dependent variables at different horizons
390   gen gh1lnTotalPriv = lnTotalPriv-l1.TotalPriv
391   gen gh2lnTotalPriv = lnTotalPriv-l2.TotalPriv
392   gen gh3lnTotalPriv = lnTotalPriv-l3.TotalPriv
393   gen gh4lnTotalPriv = lnTotalPriv-l4.TotalPriv
394
395   */
396
397
398   scalar rwrmse = .01755403
399
400   reg d.lnTotalPriv l(1,2)d.lnTotalPriv if tin(,2022m3)
401   predict pd
402   gen pTotalPriv=exp((rwrmse^2)/2)*exp(l.lnTotalPriv+pd) if date==tm(2022m3)
403   gen ub1=exp((rwrmse^2)/2)*exp(l.lnTotalPriv+pd+1*rwrmse) if date==tm(2022m3)
404   gen lb1=exp((rwrmse^2)/2)*exp(l.lnTotalPriv+pd-1*rwrmse) if date==tm(2022m3)
405   gen ub2=exp((rwrmse^2)/2)*exp(l.lnTotalPriv+pd+2*rwrmse) if date==tm(2022m3)
406   gen lb2=exp((rwrmse^2)/2)*exp(l.lnTotalPriv+pd-2*rwrmse) if date==tm(2022m3)
407   gen ub3=exp((rwrmse^2)/2)*exp(l.lnTotalPriv+pd+3*rwrmse) if date==tm(2022m3)
408   gen lb3=exp((rwrmse^2)/2)*exp(l.lnTotalPriv+pd-3*rwrmse) if date==tm(2022m3)
409   drop pd
410
411
412   replace pTotalPriv=TotalPriv if date==tm(2022m3)
413   replace ub1=TotalPriv if date==tm(2022m3)
414   replace ub2=TotalPriv if date==tm(2022m3)
415   replace ub3=TotalPriv if date==tm(2022m3)
416   replace lb1=TotalPriv if date==tm(2022m3)
417   replace lb2=TotalPriv if date==tm(2022m3)
418   replace lb3=TotalPriv if date==tm(2022m3)
419
420   tsline TotalPriv pTotalPriv lb3 lb2 lb1 ub1 ub2 ub3 ///
421       if tin(2018m3,2022m3) , legend(off) ///
422       lpattern( solid solid longdash dash shortdash shortdash dash longdash) ///
423       lcolor(black blue red orange gray gray orange red)
424
425   twoway (tsrline ub3 ub2 if tin(2018m3,2022m3), ///
426       recast(rarea) fcolor(red) fintensity(5) lwidth(none) ) ///
427       (tsrline ub2 ub1 if tin(2018m3,2022m3), ///
428       recast(rarea) fcolor(red) fintensity(15) lwidth(none) ) ///
429       (tsrline ub1 pTotalPriv if tin(2018m3,2022m3), ///
430       recast(rarea) fcolor(red) fintensity(35) lwidth(none) ) ///
431       (tsrline pTotalPriv lb1 if tin(2018m3,2022m3), ///
432       recast(rarea) fcolor(red) fintensity(35) lwidth(none) ) ///
433       (tsrline lb1 lb2 if tin(2018m3,2022m3), ///
434       recast(rarea) fcolor(red) fintensity(15) lwidth(none) ) ///
435       (tsrline lb2 lb3 if tin(2018m3,2022m3), ///
```

```
436       recast(rarea) fcolor(red) fintensity(5) lwidth(none) ) ///
437       (tsline TotalPriv if tin(2018m3,2022m3) , ///
438       lcolor(gs6) lwidth(thick) ) ///
439       (tsline pTotalPriv if tin(2018m3,2022m3) , ///
440       lcolor(gs12) lwidth(thick) ) ///
441       (scatter TotalPriv date if tin(2018m3,2022m3) , ms(Oh) mcolor(gs6) ) , ///
442       scheme(s1mono) legend(off)
443
444   graph export "Fan Chart.pdf", replace
445
446
447
448
449   *Examine Error Distribution
450   gen res=d.lnTotalPriv-pred
451   hist res, frac normal saving(ps5reshist, replace) scheme(s1mono)
452   swilk res
453   sktest res
454
455   *Run model on last window of 84 months (7 years)
456   reg d.lnTotalPriv l(1,2)d.lnTotalPriv
457   predict pdlny if date==tm(2022m3)
458   replace pdlny =pred if date<tm(2022m3)
459
460
461
462   *Normal Interval
463   gen ressq=res^2 // generating squared errors
464   summ ressq // getting the mean of the squared errors
465   gen pyn=exp(pdlny+l.lnTotalPriv+0.5*r(mean))
466   gen ubyn=pyn*exp(1.96*r(mean)^0.5)
467   gen lbyn=pyn*exp(-1.96*r(mean)^0.5)
468
469
470   twoway (tsline ubyn pyn  lbyn if tin(2021m3,2022m3)) ///
471    (scatter TotalPriv date if tin(2021m3,2022m3), ms(+) ) ///
472    (scatter pyn date if tin(2021m3,2022m3), ms(oh) ) , ///
473    scheme(s1mono) title("Normal") legend(off)
474   graph save TotalPriv.gph, replace
475
476
477   *Empirical Interval
478
479   gen expres=exp(res)
480   summ expres // mean is the multiplicative correction factor
481   gen pye=r(mean)*exp(pdlny+l.lnTotalPriv)
482   _pctile expres, percentile(2.5,97.5) //  corrections for the bounds
483   return list
484   gen lbye=r(r1)*pye
485   gen ubye=r(r2)*pye
486
487   twoway (tsline ubye pye lbye if tin((2021m3,2022m3)) ///
488    (scatter TotalPriv date if tin((2021m3,2022m3), ms(+) ) ///
489    (scatter pye date if tin((2021m3,2022m3), ms(oh)) , ///
490    scheme(s1mono) title("Empirical") legend(off))
491   graph save TotalPrive.gph, replace
492
493   graph combine TotalPriv.gph TotalPrive.gph, ///
494    scheme(s1mono) title("TotalPriv") ///
495    t2title("Rolling Window Forecast 95% Intervals")
496   graph save ps5combined.emf, replace
497
498   list pyn lbyn ubyn pye lbye ubye if date==tm(2022m3)
```

```
499
500
501   *Show 90% and 99% intervals
502
503   *Normal Interval
504   summ ressq // getting the mean of the squared errors
505   gen ubyn90=pyn*exp(1.64*r(mean)^0.5)
506   gen lbyn90=pyn*exp(-1.64*r(mean)^0.5)
507   gen ubyn99=pyn*exp(2.58*r(mean)^0.5)
508   gen lbyn99=pyn*exp(-2.58*r(mean)^0.5)
509
510   twoway (tsline ubyn99 ubyn90 pyn lbyn90 lbyn99 if tin(2020m1,2022m3)) ///
511    (scatter TotalPriv date if tin(2020m1,2022m3), ms(+) ) ///
512    (scatter pyn date if tin(2022m3,2022m3), ms(oh) ) , ///
513    scheme(s1mono) title("Normal") legend(off)
514   graph save TotalPriv9099.gph, replace
515
516   *Empirical Interval
517   summ expres // mean is the multiplicative correction factor
518   _pctile expres, percentile(5,95) //  corrections for the bounds
519   return list
520   gen lbye90=r(r1)*pye
521   gen ubye90=r(r2)*pye
522   _pctile expres, percentile(.5,99.5) //  corrections for the bounds
523   return list
524   gen lbye99=r(r1)*pye
525   gen ubye99=r(r2)*pye
526
527   twoway (tsline ubye99 ubye90 pye lbye90 lbye99 if tin(2020m1,2022m3)) ///
528    (scatter TotalPriv date if tin(2020m1,2022m3), ms(+) ) ///
529    (scatter pye date if tin(2022m3,2022m3), ms(oh) ) , ///
530    scheme(s1mono) title("Empirical") legend(off)
531   graph save TotalPriv9099e.gph, replace
532
533   graph combine TotalPriv9099e.gph TotalPriv9099.gph, ///
534    scheme(s1mono) title("TotalPriv") ///
535    t2title("Rolling Window Forecast 90% and 99% Intervals")
536   graph save ps5combined9099.emf, replace
537
538
539
540
541   log close
542
```