```
  ___  ____  ____  ____  ____(R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/
  Statistics/Data analysis

   User: Project
```

```
      name:  <unnamed>
       log:  E:\Time Series\Project\Project_2.smcl
  log type:  smcl
 opened on:   2 May 2022, 16:05:58

1 .
2 . *Import data from csv file.
3 . import delimited "Project_Monthly.txt"
  (encoding automatically selected: ISO-8859-1)
  (12 vars, 266 obs)

4 .
5 . *Check to make sure data is imported.
6 . describe

  Contains data
   Observations:           266
      Variables:            12
```

Variable name	Storage type	Display format	Value label	Variable label
date	str10	%10s		**DATE**
apus35b72610	float	%9.0g		**APUS35B72610**
apus35b74716	float	%9.0g		**APUS35B74716**
miam112lfn	long	%12.0g		**MIAM112LFN**
miam112urn	float	%9.0g		**MIAM112URN**
smu123310005~01	float	%9.0g		**SMU12331000500000001**
smu1233100050~2	float	%9.0g		**SMU12331000500000002**
smu1233100050~3	float	%9.0g		**SMU12331000500000003**
smu123310005~11	float	%9.0g		**SMU12331000500000011**
v10	byte	%8.0g		
v11	byte	%8.0g		
v12	byte	%8.0g		

Sorted by:
Note: Dataset has changed since last saved.

```
7 . summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
date	**0**				
apus35b72610	**265**	**.112083**	**.0133757**	**.08**	**.164**
apus35b74716	**266**	**2.885305**	**.7793685**	**1.356**	**4.352**
miam112lfn	**266**	**2845569**	**207946.7**	**2475966**	**3193207**
miam112urn	**266**	**5.5**	**2.565666**	**2.1**	**13.8**
smu123310~01	**266**	**2084.412**	**183.9617**	**1834.6**	**2470.5**
smu1233100~2	**182**	**35.06538**	**.5882454**	**32.3**	**36.3**
smu1233100~3	**182**	**23.50538**	**1.898119**	**20.98**	**29.5**
smu123310~11	**182**	**824.0697**	**66.1051**	**746.95**	**1029.55**
v10	**0**				
v11	**0**				
v12	**0**				

```
. 
. *Rename the four variables.
. 
. *Average Weekly Earnings of All Employees: Total Private in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
. rename smu12331000500000011 WeeklyWage

. 
. *Average Hourly Earnings of All Employees: Total Private in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
. rename smu12331000500000003 HourlyWage

. 
. *Average Weekly Hours of All Employees: Total Private in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
. rename smu12331000500000002 WeeklyHrs

. 
. *All Employees: Total Private in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
. rename smu12331000500000001 AllEmployees

. 
. *Average Price: Electricity per Kilowatt-Hour in Miami-Fort Lauderdale-West Palm Beach, FL (CBSA)
. rename apus35b72610 Average_Price_Elec

. 
. *Average Price: Gasoline, Unleaded Premium (Cost per Gallon/3.785 Liters) in Miami-Fort Lauderdale-West Palm Beach, Fl
. rename apus35b74716 Average_Price_Gas

. 
. *Civilian Labor Force in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
. rename  miam112lfn Labor_Force

. 
. *Unemployment Rate in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
. rename miam112urn Unem_Rate

. 
. *Generate a monthly date variable (make its display format monthly time, %tm)
. generate datestring=date(date,"YMD")

. gen datec = mofd(datestring)

. format datec %tm

. tsset datec

Time variable: datec, 2000m1 to 2022m2
        Delta: 1 month

. 
. keep if tin(,2022m2)
(0 observations deleted)

. 
. *add January 2020 to the data,
. tsappend, add(1)
```

```
45 . gen month=month(dofm(datec))

46 .
47 . *Generate dummy month indicators
48 . tabulate month, generate(m)

      month |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         23        8.61        8.61
          2 |         23        8.61       17.23
          3 |         23        8.61       25.84
          4 |         22        8.24       34.08
          5 |         22        8.24       42.32
          6 |         22        8.24       50.56
          7 |         22        8.24       58.80
          8 |         22        8.24       67.04
          9 |         22        8.24       75.28
         10 |         22        8.24       83.52
         11 |         22        8.24       91.76
         12 |         22        8.24      100.00
------------+-----------------------------------
      Total |        267      100.00

49 .
50 . *Generate natural logs of the variables to be used in the analysis
51 . gen lnWeeklyWage=ln(WeeklyWage)
   (85 missing values generated)

52 .
53 . gen lnHourlyWage=ln(HourlyWage)
   (85 missing values generated)

54 .
55 . gen lnWeeklyHrs=ln(WeeklyHrs)
   (85 missing values generated)

56 .
57 . /*
   > *tsline plots
   > tsline lnWeeklyWage, title("tsline lnWeeklyWage") saving("tsline4", replace)
   >
   > tsline lnHourlyWage, title("tsline lnHourlyWage") saving("tsline5", replace)
   >
   > tsline lnWeeklyHrs, title("tsline lnWeeklyHrs") saving("tsline6", replace)
   >
   > graph combine "tsline4" "tsline5" "tsline6", rows(2)
   > graph export "tsline2.emf", replace
   >
   > *AC
   > ac lnWeeklyWage, title("Autocorrelogram lnWeeklyWage") saving("ac4", replace)
   >
   > ac lnHourlyWage, title("Autocorrelogram lnHourlyWage") saving("ac5", replace)
   >
   > ac lnWeeklyHrs, title("Autocorrelogram lnWeeklyHrs") saving("ac6", replace)
   >
   > graph combine "ac4" "ac5" "ac6", rows(2)
   > graph export "dependence3.emf", replace
   >
   > *PAC
   > pac lnWeeklyWage, title("Partial Autocorrelogram lnWeeklyWage") saving("pac4", replace)
   >
   > pac lnHourlyWage, title("Partial Autocorrelogram lnHourlyWage") saving("pac5", replace)
   >
   > pac lnWeeklyHrs, title("Partial Autocorrelogram lnWeeklyHrs") saving("pac6", replace)
   >
   > graph combine "pac4" "pac5" "pac6", rows(2)
   > graph export "dependence4.emf", replace
```

```
>
>
> *Generate lags for vselect
> gen dlnWeeklyWage = d.lnWeeklyWage
>
> quietly forvalues i = 1/12 {
>         gen dlnHourlyWagel`i'= l`i'd.lnHourlyWage
> }
>
> quietly forvalues i = 1/12 {
>         gen dlnWeeklyHrsl`i'= l`i'd.lnWeeklyHrs
> }
>
> quietly forvalues i = 1/12 {
>         gen dlnWeeklyWagel`i'= l`i'd.lnWeeklyWage
> }
>
>
> *Vselecting the models for WeeklyWage
> vselect dlnWeeklyWage dlnWeeklyWagel* dlnHourlyWagel*  dlnWeeklyHrsl*, best fix( m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12)
>
> *Check LOOCV for them
> scalar drop _all
>
> reg d.lnWeeklyWage l(1/12)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
> estat ic // getting ic
>                 scalar define df1=el(r(S),1,4) // saving model df
>                 scalar define aic1=el(r(S),1,5) // saving aic
>                 scalar define bic1=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(12)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
>                 scalar define loormse1=r(rmse)
>
> reg d.lnWeeklyWage l(3)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
> estat ic // getting ic
>                 scalar define df2=el(r(S),1,4) // saving model df
>                 scalar define aic2=el(r(S),1,5) // saving aic
>                 scalar define bic2=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(3)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
>                 scalar define loormse2=r(rmse)
>
> reg d.lnWeeklyWage l(1, 2)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
> estat ic // getting ic
>                 scalar define df3=el(r(S),1,4) // saving model df
>                 scalar define aic3=el(r(S),1,5) // saving aic
>                 scalar define bic3=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(1, 2)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
>                 scalar define loormse3=r(rmse)
>
> reg d.lnWeeklyWage l(1, 2, 10)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
> estat ic // getting ic
>                 scalar define df4=el(r(S),1,4) // saving model df
>                 scalar define aic4=el(r(S),1,5) // saving aic
>                 scalar define bic4=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(1, 2, 10)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
>                 scalar define loormse4=r(rmse)
>
> reg d.lnWeeklyWage l(1, 2, 9 , 10)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
> estat ic // getting ic
```

```
>                 scalar define df5=el(r(S),1,4) // saving model df
>                 scalar define aic5=el(r(S),1,5) // saving aic
>                 scalar define bic5=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(1, 2, 9, 10)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>         if tin(2008m1,2022m2)
>                 scalar define loormse5=r(rmse)
>
> reg d.lnWeeklyWage l(1, 2)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12
>         if tin(2008m1,2022m2)
> estat ic // getting ic
>                 scalar define df6=el(r(S),1,4) // saving model df
>                 scalar define aic6=el(r(S),1,5) // saving aic
>                 scalar define bic6=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(1, 2)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m
>         if tin(2008m1,2022m2)
>                 scalar define loormse6=r(rmse)
>
> reg d.lnWeeklyWage l(1, 2)d.lnWeeklyWage l(6, 9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11
>         if tin(2008m1,2022m2)
> estat ic // getting ic
>                 scalar define df7=el(r(S),1,4) // saving model df
>                 scalar define aic7=el(r(S),1,5) // saving aic
>                 scalar define bic7=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(1, 2)d.lnWeeklyWage l(6, 9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9 m1
>         if tin(2008m1,2022m2)
>                 scalar define loormse7=r(rmse)
>
> reg d.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m
>         if tin(2008m1,2022m2)
> estat ic // getting ic
>                 scalar define df8=el(r(S),1,4) // saving model df
>                 scalar define aic8=el(r(S),1,5) // saving aic
>                 scalar define bic8=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9
>         if tin(2008m1,2022m2)
>                 scalar define loormse8=r(rmse)
>
> reg d.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(7, 10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9 m1
>         if tin(2008m1,2022m2)
> estat ic // getting ic
>                 scalar define df9=el(r(S),1,4) // saving model df
>                 scalar define aic9=el(r(S),1,5) // saving aic
>                 scalar define bic9=el(r(S),1,6) // saving bic
> loocv reg d.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(7, 10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8
>         if tin(2008m1,2022m2)
>                 scalar define loormse9=r(rmse)
>
> *Creating a comparision table
> matrix drop _all
> matrix fit1=(df1,aic1,bic1,loormse1)
> matrix fit2=(df2,aic2,bic2,loormse2)
> matrix fit3=(df3,aic3,bic3,loormse3)
> matrix fit4=(df4,aic4,bic4,loormse4)
> matrix fit5=(df5,aic5,bic5,loormse5)
> matrix fit6=(df6,aic6,bic6,loormse6)
> matrix fit7=(df7,aic7,bic7,loormse7)
> matrix fit8=(df8,aic8,bic8,loormse8)
> matrix fit9=(df9,aic9,bic9,loormse9)
>
> matrix FIT=fit1\fit2\fit3\fit4\fit5\fit6\fit7\fit8\fit9
> matrix rownames FIT= "Model 12-Lag AR" "Model 1" "Model 2" "Model 3" "Model 4" "Model 5" "Model 6" "Model 7" "Model 8"
> matrix colnames FIT=df AIC BIC LOOCV
> matrix list FIT
>
> summ datec if l12d.lnWeeklyWage~=. & l11d.lnWeeklyHrs~=. & l10d.lnHourlyWage~=.
>
> summ datec if datec==tm(2022m2)
```

```
> 
> scalar drop _all
> quietly forval w=48(12)144 {
> /* 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=721/745 {
>         /* 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.lnWeeklyWage l(1/12)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
>                 if datec>=wstart & datec<=wend // restricts the model to the window
>         replace nobs=e(N) if datec==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if datec==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d.lnWeeklyWage)^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) // min 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 if datec==tm(2022m2)
> 
> scalar drop _all
> quietly forval w=48(12)144 {
> /* 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=721/745 {
>         /* 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.lnWeeklyWage l(1, 2)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9 m10
>                 if datec>=wstart & datec<=wend // restricts the model to the window
>         replace nobs=e(N) if datec==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if datec==`t' // saving the single forecast value
```

```
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d.lnWeeklyWage)^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) // min 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 if datec==tm(2022m2)
> 
> scalar drop _all
> quietly forval w=48(12)144 {
> /* 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=721/745 {
>         /* 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.lnWeeklyWage l(1, 2)d.lnWeeklyWage l(6, 9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9
>                 if datec>=wstart & datec<=wend // restricts the model to the window
>         replace nobs=e(N) if datec==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if datec==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d.lnWeeklyWage)^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) // min 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 if datec==tm(2022m2)
> 
> scalar drop _all
> quietly forval w=48(12)144 {
> /* 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=721/745 {
>         /* 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.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8
>                 if datec>=wstart & datec<=wend // restricts the model to the window
>         replace nobs=e(N) if datec==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if datec==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d.lnWeeklyWage)^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) // min 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 if datec==tm(2022m2)
>
> scalar drop _all
> quietly forval w=48(12)144 {
> /* 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=721/745 {
>         /* 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.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(7, 10)d.lnHourlyWage m2 m3 m4 m5 m6 m7
>                 if datec>=wstart & datec<=wend // restricts the model to the window
>         replace nobs=e(N) if datec==`t' // number of observations used
>         predict ptemp // temporary predicted values
>         replace pred=ptemp if datec==`t' // saving the single forecast value
>         drop ptemp wstart wend // clear these to prepare for the next loop
>         }
> gen errsq=(pred-d.lnWeeklyWage)^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) // min 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 if datec==tm(2022m2)
>
> /*
> Model 12-Lag: RWrmse132 =  .01332579
> Model 5: RWrmse72 = .01285895
> Model 6: RWrmse72 = .01334716
> Model 7: RWrmse72 = .01295601
```

```
   > Model 8: RWrmse72 = .01278424
   > */
   >
   > */
58 . *Rolling window program -- Inner Loop Only
59 .
60 . *So, the obs to fit are now 493+180=581 to 745.
61 .
62 . scalar drop _all

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

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

65 .         quietly forval t=673/745 {

66 . **End of selected rolling window implementation
67 .
68 . *Examine Error Distribution
69 . gen res=d.lnWeeklyWage-pred
   (194 missing values generated)

70 . hist res, frac normal saving(errhist2, replace) scheme(s1mono)
   (bin=8, start=-.02859622, width=.0072284)
   file errhist2.gph saved

71 . swilk res

                   Shapiro-Wilk W test for normal data

       Variable │       Obs       W           V         z       Prob>z
   ─────────────┼──────────────────────────────────────────────────────
            res │        73    0.99375      0.398    -2.007    0.97762

72 . sktest res

   Skewness and kurtosis tests for normality
                                                         ───── Joint test ─────
       Variable │       Obs   Pr(skewness)   Pr(kurtosis)   Adj chi2(2)  Prob>chi2
   ─────────────┼─────────────────────────────────────────────────────────────────
            res │        73        0.5035         0.8787          0.48     0.7870

73 .
74 . /*Run model on last window of 72 months (6 years)
   > to get most recent predictions and forecast*/
75 . reg d.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(7, 10)d.lnHourlyWage m2 m3 m4 m5 m6 m7 m8 m9 m1
   >         if tin(2017m2,2022m2)

         Source │       SS           df       MS      Number of obs   =        61
   ─────────────┼──────────────────────────────────   F(19, 41)       =      3.40
          Model │  .008932779        19  .000470146   Prob > F        =    0.0005
       Residual │   .00567227        41  .000138348   R-squared       =    0.6116
   ─────────────┼──────────────────────────────────   Adj R-squared   =    0.4316
          Total │  .014605049        60  .000243417   Root MSE        =    .01176
```

D. lnWeeklyWage	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
lnWeeklyWage						
LD.	-.5511667	.129799	-4.25	0.000	-.8133011	-.2890322
L2D.	-.4887498	.1385908	-3.53	0.001	-.7686397	-.2088599
L5D.	.1006807	.1368171	0.74	0.466	-.1756271	.3769884
L6D.	.1121592	.1310186	0.86	0.397	-.1524383	.3767566
lnWeeklyHrs						
L9D.	.1126246	.1214973	0.93	0.359	-.1327442	.3579934
L11D.	.2869692	.1167229	2.46	0.018	.0512425	.522696
lnHourlyWage						
L7D.	.2592664	.1493541	1.74	0.090	-.0423603	.5608932
L10D.	-.1694602	.1602092	-1.06	0.296	-.4930094	.1540889
m2	.0115008	.0090324	1.27	0.210	-.0067405	.029742
m3	-.0073127	.00805	-0.91	0.369	-.0235701	.0089446
m4	.0081322	.0084912	0.96	0.344	-.0090162	.0252806
m5	.0002674	.0080405	0.03	0.974	-.0159708	.0165056
m6	.0009581	.0085775	0.11	0.912	-.0163645	.0182806
m7	.0033654	.0089657	0.38	0.709	-.0147413	.021472
m8	.0122641	.008243	1.49	0.144	-.0043831	.0289112
m9	-.0039747	.0080977	-0.49	0.626	-.0203283	.0123789
m10	.0052396	.0084164	0.62	0.537	-.0117576	.0222368
m11	-.0039734	.009159	-0.43	0.667	-.0224705	.0145236
m12	.0198346	.0081022	2.45	0.019	.003472	.0361973
_cons	.001987	.006359	0.31	0.756	-.0108553	.0148292

```
76 . predict pdlnWeeklyWage if datec==tm(2022m3) // generate point forecast
   (option xb assumed; fitted values)
   (266 missing values generated)

77 .  // generate point forecast
78 . replace pdlnWeeklyWage=pred if datec<tm(2022m3)
   (73 real changes made)

79 .
80 . *Normal Interval
81 . gen ressq=res^2 // generating squared errors
   (194 missing values generated)

82 . summ ressq // getting the mean of the squared errors
```

Variable	Obs	Mean	Std. dev.	Min	Max
ressq	73	.0001466	.0001867	2.40e-07	.0008545

```
83 . gen pWeeklyWagen=exp(pdlnWeeklyWage+l.lnWeeklyWage+0.5*r(mean))
   (193 missing values generated)

84 .
```

```
 85 . *95% Interval
 86 . gen ubWeeklyWagen=pWeeklyWagen*exp(1.96*r(mean)^0.5)
    (193 missing values generated)

 87 . gen lbWeeklyWagen=pWeeklyWagen*exp(-1.96*r(mean)^0.5)
    (193 missing values generated)

 88 .
 89 . *90% Interval
 90 . gen ubWeeklyWagen90=pWeeklyWagen*exp(1.64*r(mean)^0.5)
    (193 missing values generated)

 91 . gen lbWeeklyWagen90=pWeeklyWagen*exp(-1.64*r(mean)^0.5)
    (193 missing values generated)

 92 .
 93 . *99% Interval
 94 . gen ubWeeklyWagen99=pWeeklyWagen*exp(2.58*r(mean)^0.5)
    (193 missing values generated)

 95 . gen lbWeeklyWagen99=pWeeklyWagen*exp(-2.58*r(mean)^0.5)
    (193 missing values generated)

 96 .
 97 . *Graphing the intervals
 98 . twoway (tsline ubWeeklyWagen lbWeeklyWagen pWeeklyWagen if tin(2017m2,2022m3)) ///
    >         (scatter WeeklyWage datec if tin(2017m2,2020m1), ms(Oh) ) ///
    >         (scatter WeeklyWage datec if tin(2020m2,2022m3), ms(T) ) , ///
    >         scheme(s1mono) title("Average Weekly Wage") ///
    >         t2title("Rolling Window Forecast Interval (Normal)") legend(order(1 "ubWeeklyWagen1" ///
    >                 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2" 7 "Weekly

 99 .         graph save WeeklyWagen.gph, replace
    file WeeklyWagen.gph saved

100 .
101 . twoway (tsline ubWeeklyWagen90 lbWeeklyWagen90 pWeeklyWagen if tin(2017m2,2022m3)) ///
    >         (scatter WeeklyWage datec if tin(2017m2,2020m1), ms(Oh) ) ///
    >         (scatter WeeklyWage datec if tin(2020m2,2022m3), ms(T) ) , ///
    >         scheme(s1mono) title("Average Weekly Wage") ///
    >         t2title("Rolling Window Forecast Interval (Normal)") legend(order(1 "ubWeeklyWagen1" ///
    >                 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2" 7 "Weekly

102 .         graph save WeeklyWagen90.gph, replace
    file WeeklyWagen90.gph saved

103 .
104 . twoway (tsline ubWeeklyWagen99 lbWeeklyWagen99 pWeeklyWagen if tin(2017m2,2022m3)) ///
    >         (scatter WeeklyWage datec if tin(2017m2,2020m1), ms(Oh) ) ///
    >         (scatter WeeklyWage datec if tin(2020m2,2022m3), ms(T) ) , ///
    >         scheme(s1mono) title("Average Weekly Wage") ///
    >         t2title("Rolling Window Forecast Interval (Normal)") legend(order(1 "ubWeeklyWagen1" ///
    >                 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2" 7 "Weekly

105 .         graph save WeeklyWagen99.gph, replace
    file WeeklyWagen99.gph saved
```

```
106 .
107 . *Empirical Interval
108 . gen experr=exp(res)
    (194 missing values generated)

109 . summ experr // mean is the multiplicative correction factor

        Variable |        Obs        Mean    Std. dev.       Min        Max
    -------------+---------------------------------------------------------
          experr |         73     1.00365    .0116782   .9718088   1.029662

110 .
111 . gen pWeeklyWagee=r(mean)*exp(l.lnWeeklyWage+pdlnWeeklyWage)
    (193 missing values generated)

112 .
113 . *95% Interval
114 . _pctile experr, percentile(2.5,97.5) //  corrections for the bounds

115 . return list

    scalars:
                     r(r1) =  .9816916584968567
                     r(r2) =  1.025489449501038

116 .
117 . gen lbWeeklyWagee=pWeeklyWagee*r(r1)
    (193 missing values generated)

118 . gen ubWeeklyWagee=pWeeklyWagee*r(r2)
    (193 missing values generated)

119 .
120 . *90% Interval
121 . _pctile experr, percentile(5,95) //  corrections for the bounds

122 . return list

    scalars:
                     r(r1) =  .98403000831604
                     r(r2) =  1.022963762283325

123 .
124 . gen lbWeeklyWagee90=r(r1)*pWeeklyWagee
    (193 missing values generated)

125 . gen ubWeeklyWagee90=r(r2)*pWeeklyWagee
    (193 missing values generated)

126 .
127 . *99% Interval
128 . _pctile experr, percentile(.5,99.5) //  corrections for the bounds

129 . return list

    scalars:
                     r(r1) =  .9718087911605835
                     r(r2) =  1.029662370681763
```

```
130 .
131 . gen lbWeeklyWagee99=r(r1)*pWeeklyWagee
    (193 missing values generated)

132 . gen ubWeeklyWagee99=r(r2)*pWeeklyWagee
    (193 missing values generated)

133 .
134 .
135 . twoway (tsline ubWeeklyWagee lbWeeklyWagee pWeeklyWagee if tin(2015m1,2022m3)) ///
    >         (scatter WeeklyWage datec if tin(2017m2,2022m2), ms(Oh) ) ///
    >         (scatter WeeklyWage datec if tin(2022m3,2022m3), ms(T) ) , ///
    >         scheme(s1mono) title("Average Weekly Wage") ///
    >         t2title("Rolling Window Forecast Interval (Empirical)") legend(order(1 "ubWeeklyWagen1" ///
    >                 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2" 7 "Weekly

136 .         graph save WeeklyWagee.gph, replace
    file WeeklyWagee.gph saved

137 .
138 . twoway (tsline ubWeeklyWagee90 lbWeeklyWagee90 pWeeklyWagee if tin(2015m1,2022m3)) ///
    >         (scatter WeeklyWage datec if tin(2017m2,2022m2), ms(Oh) ) ///
    >         (scatter WeeklyWage datec if tin(2022m3,2022m3), ms(T) ) , ///
    >         scheme(s1mono) title("Average Weekly Wage") ///
    >         t2title("Rolling Window Forecast Interval (Empirical)") legend(order(1 "ubWeeklyWagen1" ///
    >                 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2" 7 "Weekly

139 .         graph save WeeklyWagee90.gph, replace
    file WeeklyWagee90.gph saved

140 .
141 . twoway (tsline ubWeeklyWagee99 lbWeeklyWagee99 pWeeklyWagee if tin(2015m1,2022m3)) ///
    >         (scatter WeeklyWage datec if tin(2017m2,2022m2), ms(Oh) ) ///
    >         (scatter WeeklyWage datec if tin(2022m3,2022m3), ms(T) ) , ///
    >         scheme(s1mono) title("Average Weekly Wage") ///
    >         t2title("Rolling Window Forecast Interval (Empirical)") legend(order(1 "ubWeeklyWagen1" ///
    >                 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2" 7 "Weekly

142 .         graph save WeeklyWagee99.gph, replace
    file WeeklyWagee99.gph saved

143 .
144 . *Compare normal and empirical bounds
145 . twoway (scatter WeeklyWage datec, ms(Oh) ) ///
    >         (tsline lbWeeklyWagen ubWeeklyWagen lbWeeklyWagee ubWeeklyWagee, ///
    >                 lpattern( solid solid "-###" "-###") ///
    >         lcolor(gs8%40 gs8%40 gs1 gs1) ///
    >         lwidth(vthick vthick thick thick) ) ///
    >         if tin(2020m1,2022m3) , tline(`=scalar(break)') scheme(s1mono) ///
    >         ylabel( , grid) xlabel( , grid) ///
    >         title("Miami-Fort Lauderdale-West Palm Beach WeeklyWage") ///
    >         t2title("95% Forecast Interval Comparison") ///
    >         legend(order(1 "Actual" ///
    >                 2 "Normal Bounds" 4 "Empirical Bounds" ) holes(2) )
    break not found
```

```
146 .         graph save WeeklyWageCombined.gph, replace
    file WeeklyWageCombined.gph saved

147 .
148 . twoway (scatter WeeklyWage datec, ms(Oh) ) ///
    >         (tsline lbWeeklyWagen90 ubWeeklyWagen90 lbWeeklyWagee90 ubWeeklyWagee90, ///
    >                 lpattern( solid solid "-###" "-###") ///
    >         lcolor(gs8%40 gs8%40 gs1 gs1) ///
    >         lwidth(vthick vthick thick thick) ) ///
    >         if tin(2020m1,2022m3) , tline(`=scalar(break)') scheme(s1mono) ///
    >         ylabel( , grid) xlabel( , grid) ///
    >         title("Miami-Fort Lauderdale-West Palm Beach WeeklyWage") ///
    >         t2title("90% Forecast Interval Comparison") ///
    >         legend(order(1 "Actual" ///
    >                 2 "Normal Bounds" 4 "Empirical Bounds" ) holes(2) )
    break not found

149 .         graph save WeeklyWageCombined90.gph, replace
    file WeeklyWageCombined90.gph saved

150 .
151 . twoway (scatter WeeklyWage datec, ms(Oh) ) ///
    >         (tsline lbWeeklyWagen99 ubWeeklyWagen99 lbWeeklyWagee99 ubWeeklyWagee99, ///
    >                 lpattern( solid solid "-###" "-###") ///
    >         lcolor(gs8%40 gs8%40 gs1 gs1) ///
    >         lwidth(vthick vthick thick thick) ) ///
    >         if tin(2020m1,2022m3) , tline(`=scalar(break)') scheme(s1mono) ///
    >         ylabel( , grid) xlabel( , grid) ///
    >         title("Miami-Fort Lauderdale-West Palm Beach WeeklyWage") ///
    >         t2title("99% Forecast Interval Comparison") ///
    >         legend(order(1 "Actual" ///
    >                 2 "Normal Bounds" 4 "Empirical Bounds" ) holes(2) )
    break not found

152 .         graph save WeeklyWageCombined99.gph, replace
    file WeeklyWageCombined99.gph saved

153 .
154 . list lbWeeklyWagen pWeeklyWagen ubWeeklyWagen lbWeeklyWagee pWeeklyWagee ubWeeklyWagee if datec==tm(2022m3)
```

	lbWeek~n	pWeekl~n	ubWeek~n	lbWeek~e	pWeekl~e	ubWeek~e
267.	**994.6629**	**1018.55**	**1043.01**	**1003.478**	**1022.193**	**1048.248**

```
155 .
156 . Stop
    command Stop is unrecognized
    r(199);

    end of do-file

    r(199);

157 .
```