```
**Appendix A.2

* Zhi Zheng
* Time Series Modeling and Forecasting
* Spring 2022
* Project AverWeeklyWage

clear
set more off

*set the working directory to wherever you have the data"
cd "E:\Time Series\Project"

*create log
log using "Project_2", replace

*Import data from csv file.
import delimited "Project_Monthly.txt"

*Check to make sure data is imported.
describe
summarize

*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 (CBSA)
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

keep if tin(,2022m2)

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

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

*Generate dummy month indicators
tabulate month, generate(m)

*Generate natural logs of the variables to be used in the analysis
gen lnWeeklyWage=ln(WeeklyWage)

gen lnHourlyWage=ln(HourlyWage)

gen lnWeeklyHrs=ln(WeeklyHrs)

/*
*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 m11 m12 ///
```

```
    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 m12 ///
    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 m10 m11 m12 ///
    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 m11 m12 ///
    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 m10 m11 m12 ///
    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 m10 m11 m12 ///
    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 m9 m10 m11 m12 ///
    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 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(6, 9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage 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, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(10)d.lnHourlyWage 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, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(7, 10)d.lnHourlyWage 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)

/*
Model 12-Lag: RWrmse132 =  .01332579
Model 5: RWrmse72 = .01285895
Model 6: RWrmse72 = .01334716
Model 7: RWrmse72 = .01295601
Model 8: RWrmse72 = .01278424
*/

*/
*Rolling window program -- Inner Loop Only

*So, the obs to fit are now 493+180=581 to 745.

scalar drop _all
gen pred=. // out of sample prediction
gen nobs=. // number of observations in the window for each forecast point
    quietly forval t=673/745 {

    /* 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'-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 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 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
    }
**End of selected rolling window implementation

*Examine Error Distribution
gen res=d.lnWeeklyWage-pred
hist res, frac normal saving(errhist2, replace) scheme(s1mono)
swilk res
sktest res

/*Run model on last window of 72 months (6 years)
to get most recent predictions and forecast*/
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 m10 m11 m12 ///
    if tin(2017m2,2022m2)
predict pdlnWeeklyWage if datec==tm(2022m3) // generate point forecast
 // generate point forecast
```

```
replace pdlnWeeklyWage=pred if datec<tm(2022m3)

*Normal Interval
gen ressq=res^2 // generating squared errors
summ ressq // getting the mean of the squared errors
gen pWeeklyWagen=exp(pdlnWeeklyWage+l.lnWeeklyWage+0.5*r(mean))

*95% Interval
gen ubWeeklyWagen=pWeeklyWagen*exp(1.96*r(mean)^0.5)
gen lbWeeklyWagen=pWeeklyWagen*exp(-1.96*r(mean)^0.5)

*90% Interval
gen ubWeeklyWagen90=pWeeklyWagen*exp(1.64*r(mean)^0.5)
gen lbWeeklyWagen90=pWeeklyWagen*exp(-1.64*r(mean)^0.5)

*99% Interval
gen ubWeeklyWagen99=pWeeklyWagen*exp(2.58*r(mean)^0.5)
gen lbWeeklyWagen99=pWeeklyWagen*exp(-2.58*r(mean)^0.5)

*Graphing the intervals
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 "WeeklyWage") holes(2) )
    graph save WeeklyWagen.gph, replace

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 "WeeklyWage") holes(2) )
    graph save WeeklyWagen90.gph, replace

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 "WeeklyWage") holes(2) )
    graph save WeeklyWagen99.gph, replace

*Empirical Interval
gen experr=exp(res)
summ experr // mean is the multiplicative correction factor

gen pWeeklyWagee=r(mean)*exp(l.lnWeeklyWage+pdlnWeeklyWage)

*95% Interval
_pctile experr, percentile(2.5,97.5) //  corrections for the bounds
return list

gen lbWeeklyWagee=pWeeklyWagee*r(r1)
gen ubWeeklyWagee=pWeeklyWagee*r(r2)

*90% Interval
_pctile experr, percentile(5,95) //  corrections for the bounds
return list
```

```

gen lbWeeklyWagee90=r(r1)*pWeeklyWagee
gen ubWeeklyWagee90=r(r2)*pWeeklyWagee

*99% Interval
_pctile experr, percentile(.5,99.5) //  corrections for the bounds
return list

gen lbWeeklyWagee99=r(r1)*pWeeklyWagee
gen ubWeeklyWagee99=r(r2)*pWeeklyWagee


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 "WeeklyWage") holes(2) )
    graph save WeeklyWagee.gph, replace

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 "WeeklyWage") holes(2) )
    graph save WeeklyWagee90.gph, replace

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 "WeeklyWage") holes(2) )
    graph save WeeklyWagee99.gph, replace

*Compare normal and empirical bounds
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) )
    graph save WeeklyWageCombined.gph, replace

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) )
    graph save WeeklyWageCombined90.gph, replace

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) )
    graph save WeeklyWageCombined99.gph, replace

list lbWeeklyWagen pWeeklyWagen ubWeeklyWagen lbWeeklyWagee pWeeklyWagee ubWeeklyWagee if datec==tm(2022m3)

Stop
```