

```

1  **Appendix A.2
2
3  * Zhi Zheng
4  * Time Series Modeling and Forecasting
5  * Spring 2022
6  * Project AverWeeklyWage
7
8  clear
9  set more off
10
11 *set the working directory to wherever you have the data"
12 cd "E:\Time Series\Project"
13
14 *create log
15 log using "Project_2", replace
16
17 *Import data from csv file.
18 import delimited "Project_Monthly.txt"
19
20 *Check to make sure data is imported.
21 describe
22 summarize
23
24 *Rename the four variables.
25
26 *Average Weekly Earnings of All Employees: Total Private in Miami-Fort Lauderdale-West Palm Beach,
27 FL (MSA)
28 rename smu12331000500000011 WeeklyWage
29
30 *Average Hourly Earnings of All Employees: Total Private in Miami-Fort Lauderdale-West Palm Beach,
31 FL (MSA)
32 rename smu12331000500000003 HourlyWage
33
34 *Average Weekly Hours of All Employees: Total Private in Miami-Fort Lauderdale-West Palm Beach, FL
35 (MSA)
36 rename smu12331000500000002 WeeklyHrs
37
38 *All Employees: Total Private in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
39 rename smu12331000500000001 AllEmployees
40
41 *Average Price: Electricity per Kilowatt-Hour in Miami-Fort Lauderdale-West Palm Beach, FL (CBSA)
42 rename apus35b72610 Average_Price_Elec
43
44 *Average Price: Gasoline, Unleaded Premium (Cost per Gallon/3.785 Liters) in Miami-Fort
45 Lauderdale-West Palm Beach, FL (CBSA)
46 rename apus35b74716 Average_Price_Gas
47
48 *Civilian Labor Force in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
49 rename miam1121fn Labor_Force
50
51 *Unemployment Rate in Miami-Fort Lauderdale-West Palm Beach, FL (MSA)
52 rename miam112urn Unem_Rate
53
54 *Generate a monthly date variable (make its display format monthly time, %tm)
55 generate datestring=date(date,"YMD")
56 gen datec = mofd(datestring)
57 format datec %tm
58 tsset datec
59
60 keep if tin(,2022m2)
61
62 *add January 2020 to the data,
63 tsappend, add(1)

```

```

60 gen month=month(dofm(datec))
61
62 *Generate dummy month indicators
63 tabulate month, generate(m)
64
65 *Generate natural logs of the variables to be used in the analysis
66 gen lnWeeklyWage=ln(WeeklyWage)
67
68 gen lnHourlyWage=ln(HourlyWage)
69
70 gen lnWeeklyHrs=ln(WeeklyHrs)
71
72 /*
73 *tsline plots
74 tsline lnWeeklyWage, title("tsline lnWeeklyWage") saving("tsline4", replace)
75
76 tsline lnHourlyWage, title("tsline lnHourlyWage") saving("tsline5", replace)
77
78 tsline lnWeeklyHrs, title("tsline lnWeeklyHrs") saving("tsline6", replace)
79
80 graph combine "tsline4" "tsline5" "tsline6", rows(2)
81 graph export "tsline2.emf", replace
82
83 *AC
84 ac lnWeeklyWage, title("Autocorrelogram lnWeeklyWage") saving("ac4", replace)
85
86 ac lnHourlyWage, title("Autocorrelogram lnHourlyWage") saving("ac5", replace)
87
88 ac lnWeeklyHrs, title("Autocorrelogram lnWeeklyHrs") saving("ac6", replace)
89
90 graph combine "ac4" "ac5" "ac6", rows(2)
91 graph export "dependence3.emf", replace
92
93 *PAC
94 pac lnWeeklyWage, title("Partial Autocorrelogram lnWeeklyWage") saving("pac4", replace)
95
96 pac lnHourlyWage, title("Partial Autocorrelogram lnHourlyWage") saving("pac5", replace)
97
98 pac lnWeeklyHrs, title("Partial Autocorrelogram lnWeeklyHrs") saving("pac6", replace)
99
100 graph combine "pac4" "pac5" "pac6", rows(2)
101 graph export "dependence4.emf", replace
102
103
104 *Generate lags for vselect
105 gen dlnWeeklyWage = d.lnWeeklyWage
106
107 quietly forvalues i = 1/12 {
108     gen dlnHourlyWage1`i' = l`i'.lnHourlyWage
109 }
110
111 quietly forvalues i = 1/12 {
112     gen dlnWeeklyHrs1`i' = l`i'.lnWeeklyHrs
113 }
114
115 quietly forvalues i = 1/12 {
116     gen dlnWeeklyWage1`i' = l`i'.lnWeeklyWage
117 }
118
119
120 *Vselecting the models for WeeklyWage
121 vselect dlnWeeklyWage dlnWeeklyWage1* dlnHourlyWage1* dlnWeeklyHrs1*, best fix( m2 m3 m4 m5 m6 m7
m8 m9 m10 m11 m12)

```

```

122
123 *Check LOOCV for them
124 scalar drop _all
125
126 reg d.lnWeeklyWage l(1/12)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
127     if tin(2008m1,2022m2)
128 estat ic // getting ic
129     scalar define df1=el(r(S),1,4) // saving model df
130     scalar define aic1=el(r(S),1,5) // saving aic
131     scalar define bic1=el(r(S),1,6) // saving bic
132 loocv reg d.lnWeeklyWage l(12)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
133     if tin(2008m1,2022m2)
134     scalar define loormse1=r(rmse)
135
136 reg d.lnWeeklyWage l(3)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
137     if tin(2008m1,2022m2)
138 estat ic // getting ic
139     scalar define df2=el(r(S),1,4) // saving model df
140     scalar define aic2=el(r(S),1,5) // saving aic
141     scalar define bic2=el(r(S),1,6) // saving bic
142 loocv reg d.lnWeeklyWage l(3)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
143     if tin(2008m1,2022m2)
144     scalar define loormse2=r(rmse)
145
146 reg d.lnWeeklyWage l(1, 2)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
147     if tin(2008m1,2022m2)
148 estat ic // getting ic
149     scalar define df3=el(r(S),1,4) // saving model df
150     scalar define aic3=el(r(S),1,5) // saving aic
151     scalar define bic3=el(r(S),1,6) // saving bic
152 loocv reg d.lnWeeklyWage l(1, 2)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
153     if tin(2008m1,2022m2)
154     scalar define loormse3=r(rmse)
155
156 reg d.lnWeeklyWage l(1, 2, 10)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
157     if tin(2008m1,2022m2)
158 estat ic // getting ic
159     scalar define df4=el(r(S),1,4) // saving model df
160     scalar define aic4=el(r(S),1,5) // saving aic
161     scalar define bic4=el(r(S),1,6) // saving bic
162 loocv reg d.lnWeeklyWage l(1, 2, 10)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
163     if tin(2008m1,2022m2)
164     scalar define loormse4=r(rmse)
165
166 reg d.lnWeeklyWage l(1, 2, 9 , 10)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
167     if tin(2008m1,2022m2)
168 estat ic // getting ic
169     scalar define df5=el(r(S),1,4) // saving model df
170     scalar define aic5=el(r(S),1,5) // saving aic
171     scalar define bic5=el(r(S),1,6) // saving bic
172 loocv reg d.lnWeeklyWage l(1, 2, 9, 10)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
173     if tin(2008m1,2022m2)
174     scalar define loormse5=r(rmse)
175
176 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 ///
177     if tin(2008m1,2022m2)
178 estat ic // getting ic
179     scalar define df6=el(r(S),1,4) // saving model df
180     scalar define aic6=el(r(S),1,5) // saving aic
181     scalar define bic6=el(r(S),1,6) // saving bic
182 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 ///

```

```

183     if tin(2008m1,2022m2)
184         scalar define loormse6=r(rmse)
185
186 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 ///
187     if tin(2008m1,2022m2)
188 estat ic // getting ic
189         scalar define df7=el(r(S),1,4) // saving model df
190         scalar define aic7=el(r(S),1,5) // saving aic
191         scalar define bic7=el(r(S),1,6) // saving bic
192 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 ///
193     if tin(2008m1,2022m2)
194         scalar define loormse7=r(rmse)
195
196 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 ///
197     if tin(2008m1,2022m2)
198 estat ic // getting ic
199         scalar define df8=el(r(S),1,4) // saving model df
200         scalar define aic8=el(r(S),1,5) // saving aic
201         scalar define bic8=el(r(S),1,6) // saving bic
202 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 ///
203     if tin(2008m1,2022m2)
204         scalar define loormse8=r(rmse)
205
206 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 ///
207     if tin(2008m1,2022m2)
208 estat ic // getting ic
209         scalar define df9=el(r(S),1,4) // saving model df
210         scalar define aic9=el(r(S),1,5) // saving aic
211         scalar define bic9=el(r(S),1,6) // saving bic
212 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 ///
213     if tin(2008m1,2022m2)
214         scalar define loormse9=r(rmse)
215
216 *Creating a comparision table
217 matrix drop _all
218 matrix fit1=(df1,aic1,bic1,loormse1)
219 matrix fit2=(df2,aic2,bic2,loormse2)
220 matrix fit3=(df3,aic3,bic3,loormse3)
221 matrix fit4=(df4,aic4,bic4,loormse4)
222 matrix fit5=(df5,aic5,bic5,loormse5)
223 matrix fit6=(df6,aic6,bic6,loormse6)
224 matrix fit7=(df7,aic7,bic7,loormse7)
225 matrix fit8=(df8,aic8,bic8,loormse8)
226 matrix fit9=(df9,aic9,bic9,loormse9)
227
228 matrix FIT=fit1\fit2\fit3\fit4\fit5\fit6\fit7\fit8\fit9
229 matrix rownames FIT= "Model 12-Lag AR" "Model 1" "Model 2" "Model 3" "Model 4" "Model 5" "Model 6"
"Model 7" "Model 8"
230 matrix colnames FIT=df AIC BIC LOOCV
231 matrix list FIT
232
233 summ datec if l12d.lnWeeklyWage~= . & l11d.lnWeeklyHrs~= . & l10d.lnHourlyWage~= .
234
235 summ datec if datec==tm(2022m2)
236
237 scalar drop _all
238 quietly forval w=48(12)144 {

```

```

239 /* w=small(inc)large
240 small is the smallest window
241 inc is the window size increment
242 large is the largest window.
243 (large-small)/inc must be an interger */
244 gen pred=. // out of sample prediction
245 gen nobs=. // number of observations in the window for each forecast point
246 forval t=721/745 {
247 /* t=first/last
248 first is the first date for which you want to make a forecast.
249 first-1 is the end date of the earliest window used to fit the model.
250 first-w, where w is the window width, is the date of the first
251 observation used to fit the model in the earliest window.
252 You must choose first so it is preceded by a full set of
253 lags for the model with the longest lag length to be estimated.
254 last is the last observation to be forecast. */
255 gen wstart=`t'-'w' // fit window start date
256 gen wend=`t'-1 // fit window end date
257 /* Enter the regression command immediately below.
258 Leave the if statement intact to control the window */
259 reg d.lnWeeklyWage l(1/12)d.lnWeeklyWage m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
260 if datec>=wstart & datec<=wend // restricts the model to the window
261 replace nobs=e(N) if datec==`t' // number of observations used
262 predict ptemp // temporary predicted values
263 replace pred=ptemp if datec==`t' // saving the single forecast value
264 drop ptemp wstart wend // clear these to prepare for the next loop
265 }
266 gen errsq=(pred-d.lnWeeklyWage)^2 // generating squared errors
267 summ errsq // getting the mean of the squared errors
268 scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
269 summ nobs // getting min and max obs used
270 scalar RWminobs`w'=r(min) // min obs used in the window width
271 scalar RWmaxobs`w'=r(max) // max obs used in the window width
272 drop errsq pred nobs // clearing for the next loop
273 }
274 scalar list // list the RMSE and min and max obs for each window width if datec==tm(2022m2)
275
276 scalar drop _all
277 quietly forval w=48(12)144 {
278 /* w=small(inc)large
279 small is the smallest window
280 inc is the window size increment
281 large is the largest window.
282 (large-small)/inc must be an interger */
283 gen pred=. // out of sample prediction
284 gen nobs=. // number of observations in the window for each forecast point
285 forval t=721/745 {
286 /* t=first/last
287 first is the first date for which you want to make a forecast.
288 first-1 is the end date of the earliest window used to fit the model.
289 first-w, where w is the window width, is the date of the first
290 observation used to fit the model in the earliest window.
291 You must choose first so it is preceded by a full set of
292 lags for the model with the longest lag length to be estimated.
293 last is the last observation to be forecast. */
294 gen wstart=`t'-'w' // fit window start date
295 gen wend=`t'-1 // fit window end date
296 /* Enter the regression command immediately below.
297 Leave the if statement intact to control the window */
298 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 ///
299 if datec>=wstart & datec<=wend // restricts the model to the window
300 replace nobs=e(N) if datec==`t' // number of observations used

```

```

301     predict ptemp // temporary predicted values
302     replace pred=ptemp if datec==`t' // saving the single forecast value
303     drop ptemp wstart wend // clear these to prepare for the next loop
304 }
305 gen errsq=(pred-d.lnWeeklyWage)^2 // generating squared errors
306 summ errsq // getting the mean of the squared errors
307 scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
308 summ nobs // getting min and max obs used
309 scalar RWminobs`w'=r(min) // min obs used in the window width
310 scalar RWmaxobs`w'=r(max) // max obs used in the window width
311 drop errsq pred nobs // clearing for the next loop
312 }
313 scalar list // list the RMSE and min and max obs for each window width if datec==tm(2022m2)
314
315 scalar drop _all
316 quietly forval w=48(12)144 {
317     /* w=small(inc)large
318     small is the smallest window
319     inc is the window size increment
320     large is the largest window.
321     (large-small)/inc must be an interger */
322     gen pred=. // out of sample prediction
323     gen nobs=. // number of observations in the window for each forecast point
324     forval t=721/745 {
325         /* t=first/last
326         first is the first date for which you want to make a forecast.
327         first-1 is the end date of the earliest window used to fit the model.
328         first-w, where w is the window width, is the date of the first
329         observation used to fit the model in the earliest window.
330         You must choose first so it is preceded by a full set of
331         lags for the model with the longest lag length to be estimated.
332         last is the last observation to be forecast. */
333         gen wstart=`t'-'w' // fit window start date
334         gen wend=`t'-1 // fit window end date
335         /* Enter the regression command immediately below.
336         Leave the if statement intact to control the window */
337         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 ///
338         if datec>=wstart & datec<=wend // restricts the model to the window
339         replace nobs=e(N) if datec==`t' // number of observations used
340         predict ptemp // temporary predicted values
341         replace pred=ptemp if datec==`t' // saving the single forecast value
342         drop ptemp wstart wend // clear these to prepare for the next loop
343     }
344     gen errsq=(pred-d.lnWeeklyWage)^2 // generating squared errors
345     summ errsq // getting the mean of the squared errors
346     scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
347     summ nobs // getting min and max obs used
348     scalar RWminobs`w'=r(min) // min obs used in the window width
349     scalar RWmaxobs`w'=r(max) // max obs used in the window width
350     drop errsq pred nobs // clearing for the next loop
351 }
352 scalar list // list the RMSE and min and max obs for each window width if datec==tm(2022m2)
353
354 scalar drop _all
355 quietly forval w=48(12)144 {
356     /* w=small(inc)large
357     small is the smallest window
358     inc is the window size increment
359     large is the largest window.
360     (large-small)/inc must be an interger */
361     gen pred=. // out of sample prediction
362     gen nobs=. // number of observations in the window for each forecast point

```



```

363     forval t=721/745 {
364         /* t=first/last
365         first is the first date for which you want to make a forecast.
366         first-1 is the end date of the earliest window used to fit the model.
367         first-w, where w is the window width, is the date of the first
368         observation used to fit the model in the earliest window.
369         You must choose first so it is preceded by a full set of
370         lags for the model with the longest lag length to be estimated.
371         last is the last observation to be forecast. */
372         gen wstart=`t'-'w' // fit window start date
373         gen wend=`t'-1 // fit window end date
374         /* Enter the regression command immediately below.
375         Leave the if statement intact to control the window */
376         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 ///
377         if datec>=wstart & datec<=wend // restricts the model to the window
378         replace nobse=e(N) if datec==`t' // number of observations used
379         predict ptemp // temporary predicted values
380         replace pred=ptemp if datec==`t' // saving the single forecast value
381         drop ptemp wstart wend // clear these to prepare for the next loop
382     }
383     gen errsq=(pred-d.lnWeeklyWage)^2 // generating squared errors
384     summ errsq // getting the mean of the squared errors
385     scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
386     summ nobse // getting min and max obs used
387     scalar RWminobs`w'=r(min) // min obs used in the window width
388     scalar RWmaxobs`w'=r(max) // max obs used in the window width
389     drop errsq pred nobse // clearing for the next loop
390 }
391 scalar list // list the RMSE and min and max obs for each window width if datec==tm(2022m2)
392
393 scalar drop _all
394 quietly forval w=48(12)144 {
395     /* w=small(inc)large
396     small is the smallest window
397     inc is the window size increment
398     large is the largest window.
399     (large-small)/inc must be an interger */
400     gen pred=. // out of sample prediction
401     gen nobse=. // number of observations in the window for each forecast point
402     forval t=721/745 {
403         /* t=first/last
404         first is the first date for which you want to make a forecast.
405         first-1 is the end date of the earliest window used to fit the model.
406         first-w, where w is the window width, is the date of the first
407         observation used to fit the model in the earliest window.
408         You must choose first so it is preceded by a full set of
409         lags for the model with the longest lag length to be estimated.
410         last is the last observation to be forecast. */
411         gen wstart=`t'-'w' // fit window start date
412         gen wend=`t'-1 // fit window end date
413         /* Enter the regression command immediately below.
414         Leave the if statement intact to control the window */
415         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 ///
416         if datec>=wstart & datec<=wend // restricts the model to the window
417         replace nobse=e(N) if datec==`t' // number of observations used
418         predict ptemp // temporary predicted values
419         replace pred=ptemp if datec==`t' // saving the single forecast value
420         drop ptemp wstart wend // clear these to prepare for the next loop
421     }
422     gen errsq=(pred-d.lnWeeklyWage)^2 // generating squared errors
423     summ errsq // getting the mean of the squared errors

```

```

424 scalar RWrmse`w'=r(mean)^.5 // getting the rmse for window width i
425 summ nobs // getting min and max obs used
426 scalar RWminobs`w'=r(min) // min obs used in the window width
427 scalar RWmaxobs`w'=r(max) // max obs used in the window width
428 drop errsq pred nobs // clearing for the next loop
429 }
430 scalar list // list the RMSE and min and max obs for each window width if datec==tm(2022m2)
431
432 /*
433 Model 12-Lag: RWrmse132 = .01332579
434 Model 5: RWrmse72 = .01285895
435 Model 6: RWrmse72 = .01334716
436 Model 7: RWrmse72 = .01295601
437 Model 8: RWrmse72 = .01278424
438 */
439
440 */
441 *Rolling window program -- Inner Loop Only
442
443 *So, the obs to fit are now 493+180=581 to 745.
444
445 scalar drop _all
446 gen pred=. // out of sample prediction
447 gen nobs=. // number of observations in the window for each forecast point
448 quietly forval t=673/745 {
449
450     /* t=first/last
451     first is the first date for which you want to make a forecast.
452     first-1 is the end date of the earliest window used to fit the model.
453     first-w, where w is the window width, is the date of the first
454     observation used to fit the model in the earliest window.
455     You must choose first so it is preceded by a full set of
456     lags for the model with the longest lag length to be estimated.
457     last is the last observation to be forecast. */
458
459     gen wstart=`t'-72 // fit window start date
460     gen wend=`t'-1 // fit window end date
461
462     /* Enter the regression command immediately below.
463     Leave the if statement intact to control the window */
464     reg d.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(7, 10)d.lnHourlyWage m2 m3
465     m4 m5 m6 m7 m8 m9 m10 m11 m12 ///
466     if datec>=wstart & datec<=wend // restricts the model to the window
467     replace nobs=e(N) if datec==`t' // number of observations used
468     predict ptemp // temporary predicted values
469     replace pred=ptemp if datec==`t' // saving the single forecast value
470     drop ptemp wstart wend // clear these to prepare for the next loop
471 }
472
473 **End of selected rolling window implementation
474
475 *Examine Error Distribution
476 gen res=d.lnWeeklyWage-pred
477 hist res, frac normal saving(errhist2, replace) scheme(s1mono)
478 swilk res
479 sktest res
480
481 /*Run model on last window of 72 months (6 years)
482 to get most recent predictions and forecast*/
483 reg d.lnWeeklyWage l(1, 2, 5, 6)d.lnWeeklyWage l(9, 11)d.lnWeeklyHrs l(7, 10)d.lnHourlyWage m2 m3 m4
484 m5 m6 m7 m8 m9 m10 m11 m12 ///
485 if tin(2017m2,2022m2)
486 predict pdlnWeeklyWage if datec==tm(2022m3) // generate point forecast
487 // generate point forecast

```



```

485 replace pdlnWeeklyWage=pred if datec<tm(2022m3)
486
487 *Normal Interval
488 gen ressq=res^2 // generating squared errors
489 summ ressq // getting the mean of the squared errors
490 gen pWeeklyWagen=exp(pdlnWeeklyWage+1.lnWeeklyWage+0.5*r(mean))
491
492 *95% Interval
493 gen ubWeeklyWagen=pWeeklyWagen*exp(1.96*r(mean)^0.5)
494 gen lbWeeklyWagen=pWeeklyWagen*exp(-1.96*r(mean)^0.5)
495
496 *90% Interval
497 gen ubWeeklyWagen90=pWeeklyWagen*exp(1.64*r(mean)^0.5)
498 gen lbWeeklyWagen90=pWeeklyWagen*exp(-1.64*r(mean)^0.5)
499
500 *99% Interval
501 gen ubWeeklyWagen99=pWeeklyWagen*exp(2.58*r(mean)^0.5)
502 gen lbWeeklyWagen99=pWeeklyWagen*exp(-2.58*r(mean)^0.5)
503
504 *Graphing the intervals
505 twoway (tsline ubWeeklyWagen lbWeeklyWagen pWeeklyWagen if tin(2017m2,2022m3)) ///
506       (scatter WeeklyWage datec if tin(2017m2,2020m1), ms(Oh) ) ///
507       (scatter WeeklyWage datec if tin(2020m2,2022m3), ms(T) ) , ///
508       scheme(s1mono) title("Average Weekly Wage") ///
509       t2title("Rolling Window Forecast Interval (Normal)") legend(order(1 "ubWeeklyWagen1" ///
510       2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2"
511       7 "WeeklyWage") holes(2) )
512       graph save WeeklyWagen.gph, replace
513
514 twoway (tsline ubWeeklyWagen90 lbWeeklyWagen90 pWeeklyWagen if tin(2017m2,2022m3)) ///
515       (scatter WeeklyWage datec if tin(2017m2,2020m1), ms(Oh) ) ///
516       (scatter WeeklyWage datec if tin(2020m2,2022m3), ms(T) ) , ///
517       scheme(s1mono) title("Average Weekly Wage") ///
518       t2title("Rolling Window Forecast Interval (Normal)") legend(order(1 "ubWeeklyWagen1" ///
519       2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2"
520       7 "WeeklyWage") holes(2) )
521       graph save WeeklyWagen90.gph, replace
522
523 twoway (tsline ubWeeklyWagen99 lbWeeklyWagen99 pWeeklyWagen if tin(2017m2,2022m3)) ///
524       (scatter WeeklyWage datec if tin(2017m2,2020m1), ms(Oh) ) ///
525       (scatter WeeklyWage datec if tin(2020m2,2022m3), ms(T) ) , ///
526       scheme(s1mono) title("Average Weekly Wage") ///
527       t2title("Rolling Window Forecast Interval (Normal)") legend(order(1 "ubWeeklyWagen1" ///
528       2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2"
529       7 "WeeklyWage") holes(2) )
530       graph save WeeklyWagen99.gph, replace
531
532 *Empirical Interval
533 gen experr=exp(res)
534 summ experr // mean is the multiplicative correction factor
535
536 gen pWeeklyWagee=r(mean)*exp(1.lnWeeklyWage+pdlnWeeklyWage)
537
538 *95% Interval
539 _pctile experr, percentile(2.5,97.5) // corrections for the bounds
540 return list
541
542 gen lbWeeklyWagee=pWeeklyWagee*r(r1)
543 gen ubWeeklyWagee=pWeeklyWagee*r(r2)
544
545 *90% Interval
546 _pctile experr, percentile(5,95) // corrections for the bounds
547 return list

```

```

545
546 gen lbWeeklyWagee90=r(r1)*pWeeklyWagee
547 gen ubWeeklyWagee90=r(r2)*pWeeklyWagee
548
549 *99% Interval
550 _pctile experr, percentile(.5,99.5) // corrections for the bounds
551 return list
552
553 gen lbWeeklyWagee99=r(r1)*pWeeklyWagee
554 gen ubWeeklyWagee99=r(r2)*pWeeklyWagee
555
556
557 twoway (tsline ubWeeklyWagee lbWeeklyWagee pWeeklyWagee if tin(2015m1,2022m3)) ///
558 (scatter WeeklyWage datec if tin(2017m2,2022m2), ms(Oh) ) ///
559 (scatter WeeklyWage datec if tin(2022m3,2022m3), ms(T) ) , ///
560 scheme(s1mono) title("Average Weekly Wage") ///
561 t2title("Rolling Window Forecast Interval (Empirical)") legend(order(1 "ubWeeklyWagen1" ///
562 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2"
563 7 "WeeklyWage") holes(2) )
564 graph save WeeklyWagee.gph, replace
565
566 twoway (tsline ubWeeklyWagee90 lbWeeklyWagee90 pWeeklyWagee if tin(2015m1,2022m3)) ///
567 (scatter WeeklyWage datec if tin(2017m2,2022m2), ms(Oh) ) ///
568 (scatter WeeklyWage datec if tin(2022m3,2022m3), ms(T) ) , ///
569 scheme(s1mono) title("Average Weekly Wage") ///
570 t2title("Rolling Window Forecast Interval (Empirical)") legend(order(1 "ubWeeklyWagen1" ///
571 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2"
572 7 "WeeklyWage") holes(2) )
573 graph save WeeklyWagee90.gph, replace
574
575 twoway (tsline ubWeeklyWagee99 lbWeeklyWagee99 pWeeklyWagee if tin(2015m1,2022m3)) ///
576 (scatter WeeklyWage datec if tin(2017m2,2022m2), ms(Oh) ) ///
577 (scatter WeeklyWage datec if tin(2022m3,2022m3), ms(T) ) , ///
578 scheme(s1mono) title("Average Weekly Wage") ///
579 t2title("Rolling Window Forecast Interval (Empirical)") legend(order(1 "ubWeeklyWagen1" ///
580 2 "lbWeeklyWagen1" 3 "pWeeklyWagen1" 4 "ubWeeklyWagen2" 5 "lbWeeklyWagen2" 6 "pWeeklyWagen2"
581 7 "WeeklyWage") holes(2) )
582 graph save WeeklyWagee99.gph, replace
583
584 *Compare normal and empirical bounds
585 twoway (scatter WeeklyWage datec, ms(Oh) ) ///
586 (tsline lbWeeklyWagen ubWeeklyWagen lbWeeklyWagee ubWeeklyWagee, ///
587 lpattern( solid solid "-###" "-###") ///
588 lcolor(gs8%40 gs8%40 gs1 gs1) ///
589 lwidth(vthick vthick thick thick) ) ///
590 if tin(2020m1,2022m3) , tline(`=scalar(break)') scheme(s1mono) ///
591 ylabel( , grid) xlabel( , grid) ///
592 title("Miami-Fort Lauderdale-West Palm Beach WeeklyWage") ///
593 t2title("95% Forecast Interval Comparison") ///
594 legend(order(1 "Actual" ///
595 2 "Normal Bounds" 4 "Empirical Bounds" ) holes(2) )
596 graph save WeeklyWageCombined.gph, replace
597
598 twoway (scatter WeeklyWage datec, ms(Oh) ) ///
599 (tsline lbWeeklyWagen90 ubWeeklyWagen90 lbWeeklyWagee90 ubWeeklyWagee90, ///
600 lpattern( solid solid "-###" "-###") ///
601 lcolor(gs8%40 gs8%40 gs1 gs1) ///
602 lwidth(vthick vthick thick thick) ) ///
603 if tin(2020m1,2022m3) , tline(`=scalar(break)') scheme(s1mono) ///
604 ylabel( , grid) xlabel( , grid) ///
605 title("Miami-Fort Lauderdale-West Palm Beach WeeklyWage") ///
606 t2title("90% Forecast Interval Comparison") ///
607 legend(order(1 "Actual" ///

```

```

605         2 "Normal Bounds" 4 "Empirical Bounds" ) holes(2) )
606     graph save WeeklyWageCombined90.gph, replace
607
608     twoway (scatter WeeklyWage datec, ms(0h) ) ///
609         (tsline lbWeeklyWagen99 ubWeeklyWagen99 lbWeeklyWagee99 ubWeeklyWagee99, ///
610             lpattern( solid solid "-###" "-###") ///
611             lcolor(gs8%40 gs8%40 gs1 gs1) ///
612             lwidth(vthick vthick thick thick) ) ///
613         if tin(2020m1,2022m3) , tline(`=scalar(break)') scheme(s1mono) ///
614         ylabel( , grid) xlabel( , grid) ///
615         title("Miami-Fort Lauderdale-West Palm Beach WeeklyWage") ///
616         t2title("99% Forecast Interval Comparison") ///
617         legend(order(1 "Actual" ///
618             2 "Normal Bounds" 4 "Empirical Bounds" ) holes(2) )
619     graph save WeeklyWageCombined99.gph, replace
620
621     list lbWeeklyWagen pWeeklyWagen ubWeeklyWagen lbWeeklyWagee pWeeklyWagee ubWeeklyWagee if datec==tm(
622         2022m3)
623     Stop

```