Supplementary Material

Solar photovoltaic utilization in electricity generation to tackle climate

Haydar Demirhan

Table of Contents

1. Methods

In this section, the details of the implementation of ARDL bounds testing is given.

1.1. ARDL models and ARDL bounds testing

To conduct ARDL bounds testing, we define the following conditional error correction model (ECM):

$$\begin{aligned}\Delta T_t = {} & c_0 + c_1 t + \pi_{TT} T_{t-1} + \pi_{TE} E_{t-1} + \pi_{TP} \log(P_{t-1}) + \pi_{TS} \log(S_{t-1}) \\ & + \sum_{i=1}^{p_T - 1} \psi'_{T_i} \Delta T_{t-i} + \sum_{i=1}^{p_E - 1} \psi'_{E_i} \Delta E_{t-i} + \sum_{i=1}^{p_P - 1} \psi'_{P_i} \Delta \log(P_{t-i}) \\ & + \sum_{i=1}^{p_S - 1} \psi'_{S_i} \Delta \log(S_{t-i}) + \omega'_E \Delta E_t + \omega'_P \Delta \log(P_t) + \omega'_S \Delta \log(S_t) + u_t \end{aligned} \tag{S1}$$

where $t = 1,2,\dots,72$, c_0 is the intercept, c_1 is trend coefficient, $\pi_{..}$ is long-run correlation coefficient, $\psi'_{..}$ and $\omega'_{.}$ show short-run coefficients, Δ is the difference operator, $\{u_t\}$ is the white-noise error series, p_T is the number of autoregressive lags, p_E, p_P, and p_S define the number of lags to be used for GHG emissions, PV installations, and sunspot numbers series, respectively.
To apply the ARDL bounds testing procedure, the following steps are implemented:

1. Check that all the series are either I(0) or I(1) by using unit-root tests. At this step, we consider the descriptive analyses in Section 23.1 are considered to choose the suitable unit-root test for each series.
2. Determine a suitable lag structure for the ARDL model in Eq. (1). The lag structure that minimizes an information criterion such as AIC or BIC can be used to specify the lag structure of the model. Although it is possible to set the same lag number p for all series, finding a different lag number for each series can provide more accurate results. In this study, instead of setting the same p for all series, we use different lag numbers are specified using. We use AIC and BIC BIC to decide on the optimal lag structure.
3. Fit the ARDL model in Eq. (1) with the lag orders determined at the previous step and fit another ARDL model under the null hypothesis to be able to compute the F-statistic of the ARDL bounds test.
4. Make sure that the errors arising from the model in Eq. (1) are serially independent and homoscedastic. This is one of the key assumptions of the ARDL bounds test of Pesaran et al. (2001). Breusch-Godfrey (BG), Breusch-Pagan-Godfrey (BPG), Shapiro-Wilk, and Ramsey's RESET tests are used to check serial correlation in residuals, the homoskedasticity of residuals, the normality of residuals, and issues with model specification, respectively (see Asteriou and Hall (2016, pp. 124 for BP and pp. 167 for BPG tests)).
5. Conduct a stability check to make sure that model assumptions hold. It is suggested by Pesaran et al. (2001) to use the cumulate sum (CUSUM) plot of recursive residuals and CUSUMSQ plot of squared recursive residuals from the model fitted for stability checking. The R package "dLagM" also generates the moving sum (MOSUM) of recursive residuals to test the stability (Demirhan, 2020). It is expected to have CUSUM and CUSUMSQ values within the 5% confidence levels to conclude the stability.

6. Conduct the ARDL bounds test. The “dLagM” package is used to calculate the F-statistic of the test and implement all diagnostic tests of steps (4) and (5). By this test, the statistical significance of long-run relationships between temperature anomalies and GHG emissions, PV installations, and sunspot numbers are assessed.

If we find a significant cointegration, then the general ECM model given in Eq. (2) is fitted to figure out the long-run equilibrium relationship between the series and their short-run effects on temperature anomalies.

$$\Delta T_t = c_0 + \beta M_{t-1} + \sum_{i=1}^{p_T-1} \psi'_{T_i} \Delta T_{t-i} + \sum_{i=1}^{p_E-1} \psi'_{E_i} \Delta E_{t-i} + \sum_{i=1}^{p_P-1} \psi'_{P_i} \Delta\log(P_{t-i}) + \sum_{i=1}^{p_S-1} \psi'_{S_i} \Delta\log(S_{t-i}) + \omega'_E \Delta E_t + \omega'_P \Delta\log(P_t) + \omega'_S \Delta\log(S_t) + u_t \tag{S2}$$

where the series $\{M_t\}$ is obtained as follows for each of the five cases defined by Pesaran et al. (2001):

- Case 1, 3, and 5: $M_t = T_t - [\pi_{TE}E_t + \pi_{TP}\log(P_t) + \pi_{TS}\log(S_t)]/\pi_{TT}$
- Case 2: $M_t = T_t - [\pi_{TE}E_t + \pi_{TP}\log(P_t) + \pi_{TS}\log(S_t)]/\pi_{TT} - c_0/\pi_{TT}$
- Case 4: $M_t = T_t - [\pi_{TE}E_t + \pi_{TP}\log(P_t) + \pi_{TS}\log(S_t)]/\pi_{TT} - c_1 t/\pi_{TT}$.

The parameter β in Eq. (2) shows how quickly GHG emissions, PV installations, and sunspot numbers return to the long-run equilibrium after a short-run shock (Zhai et al., 2017).

2. R Scripts for Analyses

2.1. Preliminary codes and data

Load required packages, read the data and convert it into time series objects for modeling.
To run the codes independently, please copy and paste them in to an .R file.
Clean R’s memory and load the required packages

```
library(dLagM); library(tseries); library(fUnitRoots)
library(lmtest); library(urca); library(nortest); library(nlme)
```

Load the dataset

```
data <- data.frame(matrix( c( 140, 12.33333333, 0.516666667, 175.8666667 ,
                    139.7, 14.33333333, -0.746666667 , 182.5666667 ,
                    140.4, 16.66666667, -0.24, 187.4333333 ,
                    141.7, 17 , 1.196666667, 183.2333333 ,
                    143.2, 20.66666667, 0.666666667, 169.3333333 ,
                    143.6, 30.66666667, 1.203333333, 151.5666667 ,
                    142.7, 41 , 0.416666667, 135.4    ,
                    142.9, 26 , 0.923333333, 117.8    ,
                    144.1, 72 , 0.963333333, 103.4666667 ,
                    144.5, 83.66666667, 0.45 , 93.86666667 ,
                    147.7, 68.33333333, 0.513333333, 84.33333333 ,
                    145.7, 82 , 0.57 , 74.13333333 ,
                    152.9, 92 , 0.093333333, 66.2     ,
                    152.6, 121.6666667, 0.95 , 60.03333333 ,
                    152.1, 109.3333333, 0.81 , 56.5     ,
                    152.9, 124.3333333, 1.733333333, 49.6     ,
                    151.7, 103.6666667, 0.98 , 45.46666667 ,
                    152.6, 133, 1.073333333, 41.6    ,
                    153, 69 , 0.786666667, 33.86666667 ,
                    154.2, 88 , -0.87, 30.06666667 ,
                    155.7, 84.66666667, 0.85 , 27.5    ,
                    157.7, 130.6666667, 1.223333333, 22.73333333 ,
                    157.4, 126.3333333, 0.89 , 18.7    ,
                    156.2, 150.6666667, 0.456666667, 14.33333333 ,
                    155, 355, 0.906666667, 10.56666667 ,
                    152.8, 528.6666667, 0.76 , 9.166666667 ,
                    154.3, 619, 0.436666667, 6.033333333 ,
                    153.3, 944.6666667, 0.343333333, 5.4 ,
                    155.5, 1217.666667, 0.366666667, 4.2 ,
                    154.2, 1907 , 0.65 , 2.766666667 ,
                    149.5, 2911 , 0.543333333, 3.166666667 ,
                    150.3, 3952.333333, 0.47 , 3.966666667 ,
                    144.9, 5720.333333, 1.563333333, 7.8 ,
                    147.4, 8389.666667, 1.11 , 12.43333333 ,
                    147.4, 10641.33333, 0.646666667, 16.86666667 ,
                    145.2, 16086.66667, 0.77 , 23.46666667 ,
                    143.5, 17740, 0.386666667, 27.93333333 ,
                    141.5, 21603, -0.483333333 , 40.06666667 ,
                    140.9, 29948.33333, -0.026666667 , 51  ,
                    141.6, 50779, -0.836666667 , 69  ,
                    140.7, 23075, 0.653333333, 84  ,
                    139.4, 16448.66667, 0.2, 86.7    ,
                    139.7, 18739, -0.383333333 , 93.43333333 ,
                    138.4, 43082, -0.293333333 , 87.63333333 ,
                    135, 27396.33333, 0.426666667, 83.06666667 ,
                    135.6, 25227.33333, 1.22 , 86.76666667 ,
                    133.3, 15547, 1.186666667, 86.5    ,
                    133.6, 19698.66667, 1.083333333, 90.53333333 ,
                    133, 15531.33333, 2.04 , 104.2   ,
                    133.8, 16029.66667, 1, 114.7666667 ,
                    134.5, 14054, 0.676666667, 120.1333333 ,
                    132.6, 14703.66667, 1.13 , 122.8333333 ,
                    133.7, 15875, 0.64 , 114.3   ,
                    132.9, 14835, 1.716666667, 98.26666667 ,
                    131.6, 12504.33333, 1.016666667, 88.2    ,
                    132.4, 11496.66667, 0.226666667, 77.03333333 ,
                    132.1, 12172, 0.506666667, 65.9    ,
                    132.2, 11012, 2.066666667, 58.63333333 ,
                    133, 9949.666667, 1.24 , 49.56666667 ,
                    130.6, 10353, 1.81 , 42.03333333 ,
                    133.5, 10975.66667, 0.556666667, 34.26666667 ,
                    132.8, 12986, 0.393333333, 29.53333333 ,
                    133.3, 12457.33333, 1.126666667, 25.8    ,
                    133, 13471, 0.496666667, 23.13333333 ,
                    132.3, 15054.33333, 1.476666667, 19.7    ,
                    133, 17366.66667, 1.243333333, 16.06666667 ,
                    133.2, 16242.33333, 1.153333333, 13.4    ,
                    134, 16875.33333, 1.033333333, 9.3 ,
                    132.3, 18805, 0.716666667, 9.033333333 ,
                    133.6, 23054.33333, 1.673333333, 9.333333333 ,
                    133.1, 21419.66667, 2.12 , 8.666666667 ,
                                        133.1, 20282, 0.736666667, 7.533333333 ),
                                      nrow = 72 , ncol = 4, byrow = TRUE) )

                    colnames(data) <- c("Emis", "AvrInst", "TempAnom", "SSN_SWO_SM")
                    head(data)
```

Convert the data into time series objects

```
dataAnom <- ts(data$TempAnom, frequency = 4, start = c(2001,3), end = c(2019,2) )
dataEmis <- ts(data$Emis, frequency = 4, start = c(2001,3), end = c(2019,2))
dataInst <- ts(data$AvrInst, frequency = 4, start = c(2001,3), end = c(2019,2))
dataSSN <- ts(data$SSN_SWO_SM, frequency = 4, start = c(2001,3), end = c(2019,2))

dataEmisRaw <- dataEmis
dataInstRaw <- dataInst
dataSSNRaw <- dataSSN
```

2.2. Codes for Section 3.1. Data description

Plot raw series

```
par(mfrow=c(2,2))
plot(dataAnom, ylab = expression(paste("Mean temperature anomalies (", degree ,
"C)")), main = "Raw temperature anomalies series")
plot(dataEmis, ylab = expression(paste("GHG emissions (Mt ", CO2-e , ")")), main =
"Raw GHG emissions series")
plot(dataInst, ylab = expression(paste("Installed PV capacity (", kW , ")")), main
="Raw average PV installations series")
plot(dataSSN, ylab = "Sunspot numbers", main = "Raw smoothed sunspot numbers series")
par(mfrow=c(1,1))
```

Some descriptive statistics

```
summary(dataEmis)
sqrt(var(dataEmis)/72)

summary(dataInst)
sqrt(var(dataInst)/72)

summary(dataSSN)
sqrt(var(dataSSN)/72)
```

Adjustments

```
adjust_dataEmis <- dataEmis - stl(dataEmis,s.window = 13, t.window =
13)$time.series[,2]
dataEmis <- adjust_dataEmis

adjust_dataInst <- dataInst - stl(dataInst,s.window = 13, t.window =
5)$time.series[,2]
dataInst <- log(adjust_dataInst+abs(min(adjust_dataInst))+0.1)

adjust_dataSSN <- dataSSN - stl(dataSSN,s.window = 13, t.window = 7)$time.series[,2]
dataSSN <- log(adjust_dataSSN+abs(min(adjust_dataSSN))+0.1)
```

Plot adjusted and transformed series

```
par(mfrow=c(2,2))
plot(dataEmis, ylab = expression(paste("GHG emissions (Mt ", CO2-e , ")")), main =
"Adjusted GHG emissions series")
plot(dataInst, ylab = expression(paste("Installed PV capacity (", kW , ")")), main
="Adjusted average PV installations series")
plot(dataSSN, ylab = "Sunspot numbers", main = "Adjusted smoothed sunspot numbers
series")
par(mfrow=c(1,1))
```

2.3. Codes for Section 4.1. Rolling correlation analysis

Analysis of the significance of the signal. Use raw series since correlations is impacted by non-linear transformations

```
rolCorPlot(x = dataEmisRaw, y = dataAnom , width = c(5, 7, 11, 15, 17), level = 0.95,
N = 1000, main="Temperature anomalies and GHG emissions")$test
rolCorPlot(x = dataInstRaw, y = dataAnom, width = c(5, 7, 11, 15, 17), level = 0.95,
N=1000, main="Temperature anomalies and PV installations")$test
rolCorPlot(x = dataSSNRaw, y = dataAnom, width = c(5, 7, 11, 15, 17 ), level = 0.95,
N = 1000, main = "Temperature anomalies and SSN")$test
rolCorPlot(x = dataSSNRaw, y = dataEmisRaw, width = c(5, 7, 11, 15, 17 ), level =
0.90, N = 1000, main = " GHG emissions and SSN")$test
```

2.4. Codes for Section 4.2. Degree of integration

Unit-root tests

```
pp.test(x = dataAnom, alternative = "stationary",  type = "Z(t_alpha)", lshort =
TRUE)
kpss.test(x = dataAnom, null = "Trend", lshort = TRUE)
adfTest(x = dataAnom, type =  "ct", title = NULL,  description = NULL)

pp.test(x = dataEmis, alternative = "stationary",  type = "Z(t_alpha)", lshort =
TRUE)
kpss.test(x = dataEmis, null = "Trend", lshort = TRUE)
adfTest(x = dataEmis, type =  "ct", title = NULL,  description = NULL)

pp.test(x = dataInst, alternative = "stationary",  type = "Z(t_alpha)", lshort =
TRUE)
kpss.test(x = dataInst, null = "Trend", lshort = TRUE)
adfTest(x = dataInst, type =  "ct", title = NULL,  description = NULL)

pp.test(x = dataSSN, alternative = "stationary",  type = "Z(t_alpha)", lshort = TRUE)
kpss.test(x = dataSSN, null = "Trend", lshort = TRUE)
adfTest(x = dataSSN, type =  "ct", title = NULL,  description = NULL)
```

2.5. Codes for Section 4.3. Granger causality analysis

```
dataTR <- data.frame(dataAnom, dataEmis, dataInst, dataSSN)
colnames(dataTR) <- c("TempAnom", "Emis","Inst", "SSN")
head(dataTR)

M <- c(2:22)
Mm <- length(M)
grgPvalueEmis <- array(NA, Mm)
AICEmis <- array(NA, Mm)
BICEmis <- array(NA, Mm)
grgPvalueInst <- array(NA, Mm)
AICInst <- array(NA, Mm)
BICInst <- array(NA, Mm)
grgPvalueSSN <- array(NA, Mm)
AICSSN <- array(NA, Mm)
BICSSN <- array(NA, Mm)
grgPvalueEmisAnom <- array(NA, Mm)
```

```r
AICEmisAnom <- array(NA, Mm)
BICEmisAnom <- array(NA, Mm)
grgPvalueInstAnom <- array(NA, Mm)
AICInstAnom <- array(NA, Mm)
BICInstAnom <- array(NA, Mm)
grgPvalueInstEmis <- array(NA, Mm)
AICInstEmis <- array(NA, Mm)
BICInstEmis <- array(NA, Mm)
grgPvalueEmisInst <- array(NA, Mm)
AICEmisInst <- array(NA, Mm)
BICEmisInst <- array(NA, Mm)
grgPvalueSSNAnom <- array(NA, Mm)
AICSSNAnom <- array(NA, Mm)
BICSSNAnom <- array(NA, Mm)
grgPvalueEmisSSN <- array(NA, Mm)
AICEmisSSN <- array(NA, Mm)
BICEmisSSN <- array(NA, Mm)
for ( i in M){
  grgPvalueEmis[i] <- grangertest(dataAnom ~ dataEmis, order = i)$`Pr(>F)`[2]
  AICEmis[i] <- AIC(ardlDlm(TempAnom ~ Emis, data = dataTR, p = i, q = i))
  BICEmis[i] <- BIC(ardlDlm(TempAnom ~ Emis, data = dataTR, p = i, q = i))
  grgPvalueInst[i] <- grangertest(dataAnom ~ dataInst, order = i)$`Pr(>F)`[2]
  AICInst[i] <- AIC(ardlDlm(TempAnom ~ Inst, data = dataTR, p = i, q = i))
  BICInst[i] <- BIC(ardlDlm(TempAnom ~ Inst, data = dataTR, p = i, q = i))
  grgPvalueSSN[i] <- grangertest(dataAnom ~ dataSSN, order = i)$`Pr(>F)`[2]
  AICSSN[i] <- AIC(ardlDlm(TempAnom ~ SSN, data = dataTR, p = i, q = i))
  BICSSN[i] <- BIC(ardlDlm(TempAnom ~ SSN, data = dataTR, p = i, q = i))
  grgPvalueEmisAnom[i] <- grangertest(dataEmis ~ dataAnom, order = i)$`Pr(>F)`[2]
  AICEmisAnom[i] <- AIC(ardlDlm(Emis ~ TempAnom, data = dataTR, p = i, q = i))
  BICEmisAnom[i] <- BIC(ardlDlm(Emis ~ TempAnom, data = dataTR, p = i, q = i))
  grgPvalueInstAnom[i] <- grangertest(dataInst ~ dataAnom, order = i)$`Pr(>F)`[2]
  AICInstAnom[i] <- AIC(ardlDlm(Inst ~ TempAnom, data = dataTR, p = i, q = i))
  BICInstAnom[i] <- BIC(ardlDlm(Inst ~ TempAnom, data = dataTR, p = i, q = i))
  grgPvalueInstEmis[i] <- grangertest(dataEmis ~ dataInst, order = i)$`Pr(>F)`[2]
  AICInstEmis[i] <- AIC(ardlDlm(Emis ~ Inst, data = dataTR, p = i, q = i))
  BICInstEmis[i] <- BIC(ardlDlm(Emis ~ Inst, data = dataTR, p = i, q = i))
  grgPvalueSSNAnom[i] <- grangertest(dataSSN ~ dataAnom, order = i)$`Pr(>F)`[2]
  AICSSNAnom[i] <- AIC(ardlDlm(SSN ~ TempAnom, data = dataTR, p = i, q = i))
  BICSSNAnom[i] <- BIC(ardlDlm(SSN ~ TempAnom, data = dataTR, p = i, q = i))
  grgPvalueEmisSSN[i] <- grangertest(dataEmis ~ dataSSN, order = i)$`Pr(>F)`[2]
  AICEmisSSN[i] <- AIC(ardlDlm(Emis ~ SSN, data = dataTR, p = i, q = i))
  BICEmisSSN[i] <- BIC(ardlDlm(Emis ~ SSN, data = dataTR, p = i, q = i))
  grgPvalueEmisInst[i] <- grangertest(dataInst ~ dataEmis, order = i)$`Pr(>F)`[2]
  AICEmisInst[i] <- AIC(ardlDlm(Inst ~ Emis, data = dataTR, p = i, q = i))
  BICEmisInst[i] <- BIC(ardlDlm(Inst ~ Emis, data = dataTR, p = i, q = i))
}
# par(mfrow=c(4,2))
plot(grgPvalueEmis[2:22], type = "b", main = "Temperature Anomalies ~ GHG Emissions",
ylim = c(0,1), xlab = "Order", ylab = "p-value")
lines(AICEmis[2:22]/max(AICEmis[2:22]), col="green", lty = 1, type="b")
lines(BICEmis[2:22]/max(BICEmis[2:22]), col="purple3", lty = 1, type="b")
abline(h = 0.1, col = "blue", lty = 2)
abline(h = 0.05, col = "red", lty = 2)

plot(grgPvalueEmisAnom[2:22], type = "b", main = "GHG Emissions ~ Temperature
Anomalies", ylim = c(0,1), xlab = "Order", ylab = "p-value")
lines(AICEmisAnom[2:22]/max(AICEmisAnom[2:22]), col="green", lty = 1, type="b")
lines(BICEmisAnom[2:22]/max(BICEmisAnom[2:22]), col="purple3", lty = 1, type="b")
abline(h = 0.1, col = "blue", lty = 2)
abline(h = 0.05, col = "red", lty = 2)

plot(grgPvalueInst[2:22], type = "b", main = "Temperature Anomalies ~ PV
Installations", ylim = c(0,1), xlab = "Order", ylab = "p-value")
lines(AICInst[2:22]/max(AICInst[2:22]), col = "green", lty=1, type="b")
lines(BICInst[2:22]/max(BICInst[2:22]), col = "purple3", lty=1, type="b")
abline(h = 0.1, col = "blue", lty = 2)
abline(h = 0.05, col = "red", lty = 2)

plot(grgPvalueInstAnom[2:22], type = "b", main = "PV Installations ~ Temperature
Anomalies", ylim = c(0,1), xlab = "Order", ylab = "p-value")
lines(AICInstAnom[2:22]/max(AICInstAnom[2:22]), col = "green", lty=1, type="b")
lines(BICInstAnom[2:22]/max(BICInstAnom[2:22]), col = "purple3", lty=1, type="b")
abline(h = 0.1, col = "blue", lty = 2)
abline(h = 0.05, col = "red", lty = 2)

plot(grgPvalueSSN[2:22], type = "b", main = "Temperature Anomalies ~ SSN",
     ylim = c(0,1), xlab = "Order", ylab = "p-value")
lines(AICSSN[2:22]/max(AICSSN[2:22]), col = "green", lty = 1, type="b")
lines(BICSSN[2:22]/max(BICSSN[2:22]), col = "purple3", lty = 1, type="b")
abline(h = 0.1, col = "blue", lty = 2)
abline(h = 0.05, col = "red", lty = 2)

plot(grgPvalueEmisSSN[2:22], type = "b", main = "GHG Emissions ~ SSN",
     ylim = c(0,1), xlab = "Order", ylab = "p-value")
lines(AICEmisSSN[2:22]/max(AICEmisSSN[2:22]), col = "green", lty = 1, type = "b")
lines(BICEmisSSN[2:22]/max(BICEmisSSN[2:22]), col = "purple3", lty = 1, type="b")
abline(h = 0.1, col = "blue", lty = 2)
abline(h = 0.05, col = "red", lty = 2)

plot(grgPvalueInstEmis[2:22], type = "b", main = "GHG Emissions ~ PV Installations",
ylim = c(0,1), xlab = "Order", ylab = "p-value")
lines(AICInstEmis[2:22]/max(AICInstEmis[2:22]), col = "green", lty=1, type="b")
lines(BICInstEmis[2:22]/max(BICInstEmis[2:22]), col = "purple3", lty=1, type="b")
abline(h = 0.1, col = "blue", lty = 2)
abline(h = 0.05, col = "red", lty = 2)

plot(grgPvalueEmisInst[2:22], type = "b", main = "PV Installations ~ GHG Emissions",
ylim = c(0,1), xlab = "Order", ylab = "p-value")
lines(AICEmisInst[2:22]/max(AICEmisInst[2:22]), col = "green", lty=1, type="b")
lines(BICEmisInst[2:22]/max(BICEmisInst[2:22]), col = "purple3", lty=1, type="b")
abline(h = 0.1, col = "blue", lty = 2)
abline(h = 0.05, col = "red", lty = 2)

legend("bottomleft",inset=c(0,0),legend=c("5% limit for significance", "10% limit for
significance", "AIC", "BIC"), col=c("red", "blue", 'green', 'purple3'), lty = 1,
horiz=F, cex=0.8)
```

2.6. Codes for Section 4.4. Short and long run relationships

2.6.1. Codes for Section 4.4.1. Temperature anomalies

Orders for ARDL bounds test. Output can be written to a csv file to investigate top models in terms of AIC and BIC to decide the lag orders.

```
formulaTR <- TempAnom ~ Emis + Inst + SSN

orders <- ardlBoundOrders(data = dataTR , formula = formulaTR, ic = "AIC", max.p = 5,
max.q = 5 , FullSearch = TRUE)
orders$Stat.table[order(orders$Stat.table['AIC']),]
orders$p
orders$q
orders$min.Stat

orders <- ardlBoundOrders(data = dataTR , formula = formulaTR, ic = "BIC", max.p = 5,
max.q = 5 , FullSearch = TRUE)
orders$Stat.table[order(orders$Stat.table['BIC']),]
orders$p
orders$q
orders$min.Stat
dataTR <- data.frame(dataAnom, dataEmis, dataInst, dataSSN)
colnames(dataTR) <- c("TempAnom", "Emis","Inst", "SSN")
head(dataTR)
formulaTR <- TempAnom ~ Emis + Inst + SSN
p <- data.frame(1,1,6,1)
colnames(p) <- c("orders.q", "Emis", "Inst", "SSN")
```

Implement ARDL bounds test for each of the cases

```
model1 <- ardlBound(data = dataTR , formula = formulaTR, case = 1, p=p, ECM=TRUE)
summary(model1$model$modelFull$model)
aic <- AIC(model1$model$modelFull$model)
bic <- BIC(model1$model$modelFull$model)
aic
bic

# Coefficients of cointegration equation
LRcoeff <- model1$model$modelFull$model$coefficients[1:4]
LRcoeff/LRcoeff[1]
model2 <- ardlBound(data = dataTR , formula = formulaTR, case = 2, p=p, ECM=TRUE)
aic <- AIC(model2$model$modelFull$model)
bic <- BIC(model2$model$modelFull$model)
aic
bic

model3 <- ardlBound(data = dataTR , formula = formulaTR, case = 3, p=p, ECM=TRUE)
aic <- AIC(model3$model$modelFull$model)
bic <- BIC(model3$model$modelFull$model)
aic
bic

model4 <- ardlBound(data = dataTR , formula = formulaTR, case = 4, p=p, ECM=TRUE)
aic <- AIC(model4$model$modelFull$model)
bic <- BIC(model4$model$modelFull$model)
aic
bic
```

```
model5 <- ardlBound(data = dataTR , formula = formulaTR, case = 5, p=p, ECM=TRUE)
aic <- AIC(model5$model$modelFull$model)
bic <- BIC(model5$model$modelFull$model)
aic
bic
```

Step by step explanation of the calculations given in Section 4 are given in Figure S1.

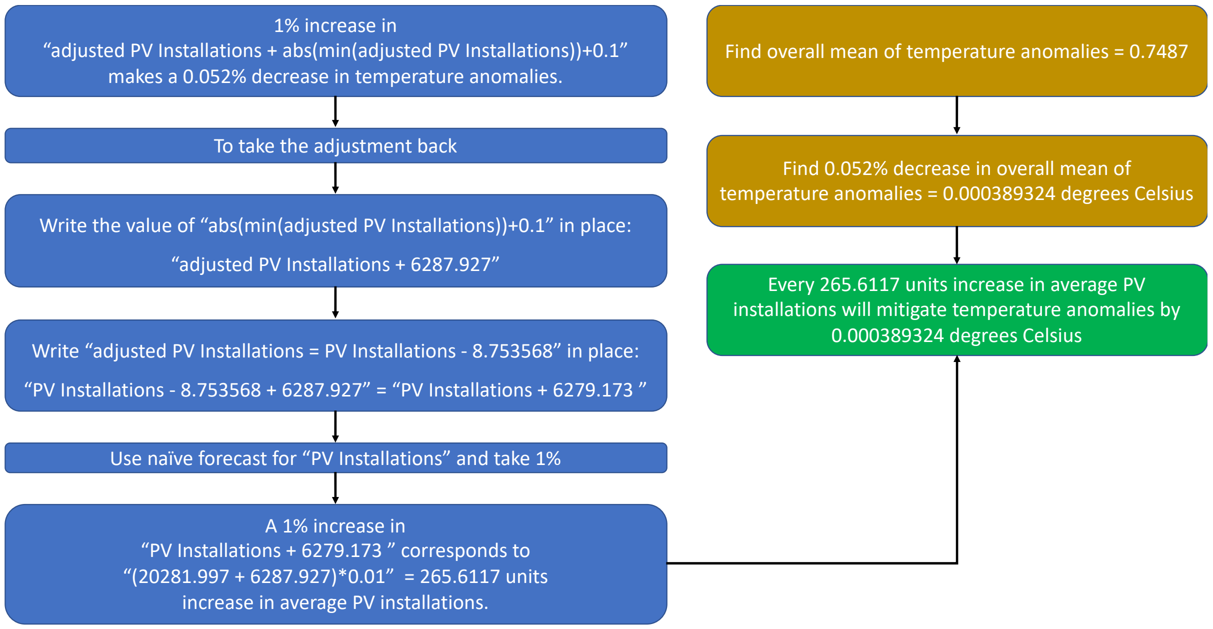
1% increase in
"adjusted PV Installations + abs(min(adjusted PV Installations))+0.1"
makes a 0.052% decrease in temperature anomalies.
To take the adjustment back
Write the value of "abs(min(adjusted PV Installations))+0.1" in place:
"adjusted PV Installations + 6287.927"
Write "adjusted PV Installations = PV Installations - 8.753568" in place:
"PV Installations - 8.753568 + 6287.927" = "PV Installations + 6279.173 "
Use naïve forecast for "PV Installations" and take 1%
A 1% increase in
"PV Installations + 6279.173 " corresponds to
"(20281.997 + 6287.927)*0.01" = 265.6117 units
increase in average PV installations.
Find overall mean of temperature anomalies = 0.7487
Find 0.052% decrease in overall mean of temperature anomalies = 0.000389324 degrees Celsius
Every 265.6117 units increase in average PV installations will mitigate temperature anomalies by 0.000389324 degrees Celsius

Figure S1. Calculation of the mitigation in temperature anomalies for installed PV capacity.

```
PVincrease <- (dataInstRaw[72] + 6279.173) * 0.01
PVincrease
OverallMeanAnom <- mean(dataAnom) # overall mean of raw temperature anomalies
OverallMeanAnom * 0.00052 # = 0.000389324 degrees Celsius mitigation
```

Every 265.6117 units increase in average PV installations will mitigate temperature anomalies by 0.000389324.
Then, the amount of mitigation is found as follows:

```
amountofMitigation <- 0.000389324*20830.67/PVincrease
amountofMitigation
```

Tempanomaly vs sunspot numbers

Prepare data and model formula.

```
dataTM<- data.frame(dataAnom, dataSSN)
colnames(dataTM) <- c("TempAnom", "SSN")
head(dataTM)

formulaTM <- TempAnom ~ SSN
```

Find the orders of the model.

```
orders <- ardlBoundOrders(data = dataTM , formula = formulaTM, ic = "AIC",
                          max.p = 5,  max.q = 5 , FullSearch = TRUE)
orders$p
orders$q
orders$min.Stat

orders <- ardlBoundOrders(data = dataTM , formula = formulaTM, ic = "BIC",
                          max.p = 5,  max.q = 5 , FullSearch = TRUE)
orders$p
orders$q
orders$min.Stat

p <- data.frame(2,6)
colnames(p) <- c("orders.q", "SSN")
```

Implement the ARDL bounds test for all cases.

```
model1 <- ardlBound(data = dataTM , formula = formulaTM, case = 1, p=p, ECM=TRUE)
summary(model1$model$modelFull$model)
aic <- AIC(model1$model$modelFull$model)
bic <- BIC(model1$model$modelFull$model)
aic
bic
model2 <- ardlBound(data = dataTM , formula = formulaTM, case = 2, p=p, ECM=TRUE)
summary(model2$model$modelFull$model)
aic <- AIC(model2$model$modelFull$model)
bic <- BIC(model2$model$modelFull$model)
aic
bic

model3 <- ardlBound(data = dataTM , formula = formulaTM, case = 3, p=p, ECM=TRUE)
summary(model3$model$modelFull$model)
aic <- AIC(model3$model$modelFull$model)
bic <- BIC(model3$model$modelFull$model)
aic
bic
model4 <- ardlBound(data = dataTM , formula = formulaTM, case = 4, p=p, ECM=TRUE)
summary(model4$model$modelFull$model)
aic <- AIC(model4$model$modelFull$model)
bic <- BIC(model4$model$modelFull$model)
aic
bic
model5 <- ardlBound(data = dataTM , formula = formulaTM, case = 5, p=p, ECM=TRUE)
summary(model5$model$modelFull$model)
aic <- AIC(model5$model$modelFull$model)
bic <- BIC(model5$model$modelFull$model)
aic
bic
```

2.6.2. Codes for Section 4.4.2 GHG emissions

Prepare data and model formula.

```
dataES <- data.frame(dataEmis,dataSSN)
colnames(dataES) <- c("Emis", "SSN")
head(dataES)
formulaES <- Emis ~ SSN
```

Find the orders of the model.

```
orders <- ardlBoundOrders(data = dataES , formula = formulaES, ic = "AIC",
                          max.p = 5,  max.q = 5 , FullSearch = TRUE)
orders$p
orders$q
orders$min.Stat

orders <- ardlBoundOrders(data = dataES , formula = formulaES, ic = "BIC",
                          max.p = 5,  max.q = 5 , FullSearch = TRUE)
orders$p
orders$q
orders$min.Stat
```

Following the parsimony principle, the orders (3,5) are selected.

```
p <- data.frame(4,6)
colnames(p) <- c("orders.q", "SSN")
```

Fit the model for Cases 1 to 5 and find AIC and BIC values of each case. The smallest AIC and BIC corresponds to Case 1. So, coefficients of cointegration equation are calculated for Case 1.

```
model3 <- ardlBound(data = dataES , formula = formulaES, case = 3, p=p, ECM=TRUE)
summary(model3$model$modelFull$model)
aic <- AIC(model3$model$modelFull$model)
bic <- BIC(model3$model$modelFull$model)
aic
bic

# Coefficients of cointegration equation
LRcoeff <- model3$model$modelFull$model$coefficients[2:3]
LRcoeff/LRcoeff[1]
model1 <- ardlBound(data = dataES , formula = formulaES, case = 1, p=p, ECM=TRUE)
summary(model1$model$modelFull$model)
aic <- AIC(model1$model$modelFull$model)
bic <- BIC(model1$model$modelFull$model)
aic
bic

model2 <- ardlBound(data = dataES , formula = formulaES, case = 2, p=p, ECM=TRUE)
summary(model2$model$modelFull$model)
aic <- AIC(model2$model$modelFull$model)
bic <- BIC(model2$model$modelFull$model)
aic
bic

model4 <- ardlBound(data = dataES , formula = formulaES, case = 4, p=p, ECM=TRUE)
summary(model4$model$modelFull$model)
aic <- AIC(model4$model$modelFull$model)
bic <- BIC(model4$model$modelFull$model)
aic
bic

model5 <- ardlBound(data = dataES , formula = formulaES, case = 5, p=p, ECM=TRUE)
summary(model5$model$modelFull$model)
aic <- AIC(model5$model$modelFull$model)
bic <- BIC(model5$model$modelFull$model)
```

```
aic
bic
```

3. Additional Figures

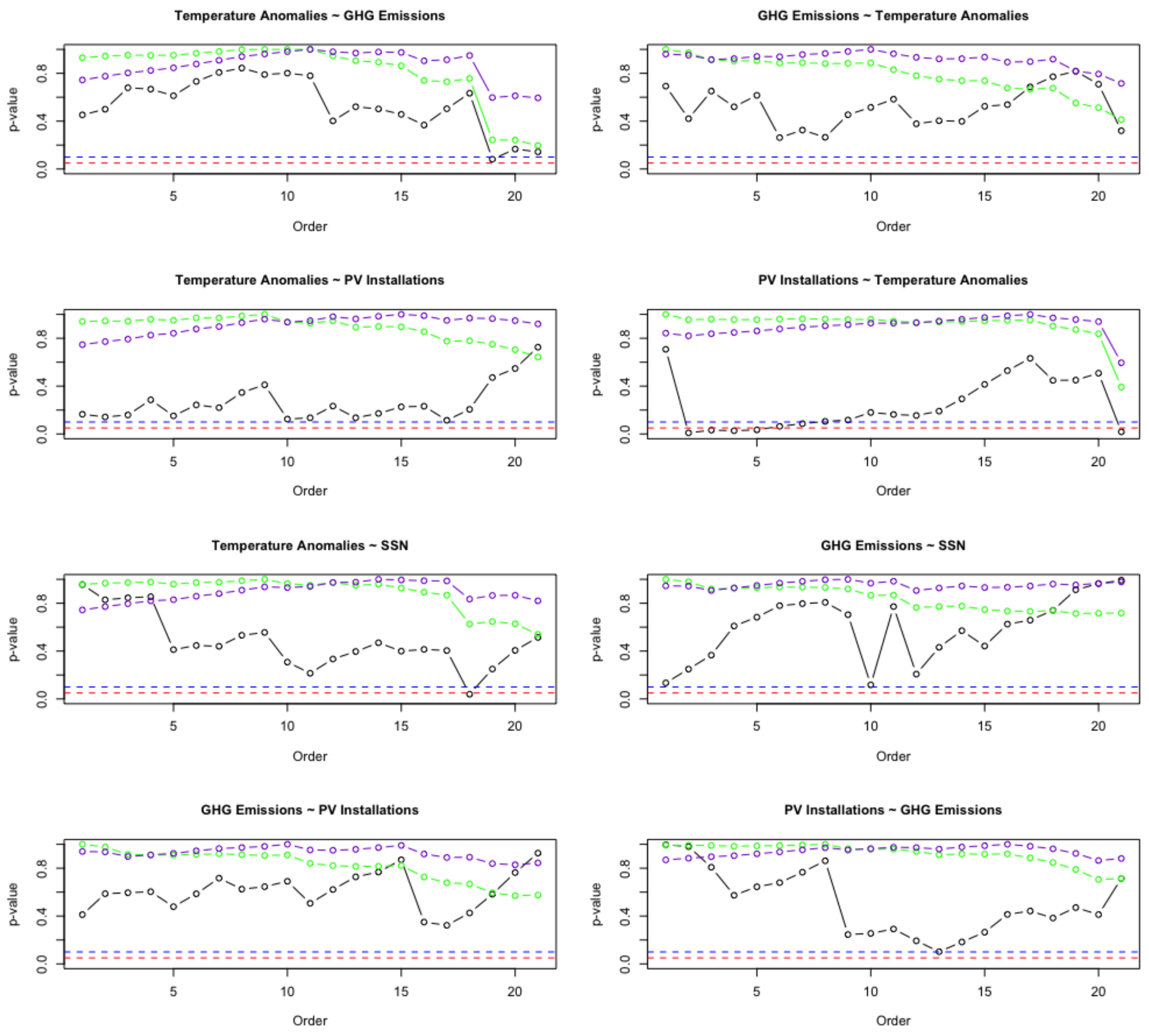

Figure S2. The *p*-values of the Granger causality test (black), rescaled AIC (green) and rescaled BIC (purple) values, and the significance levels of 0.05 (red) and 0.10 (blue).

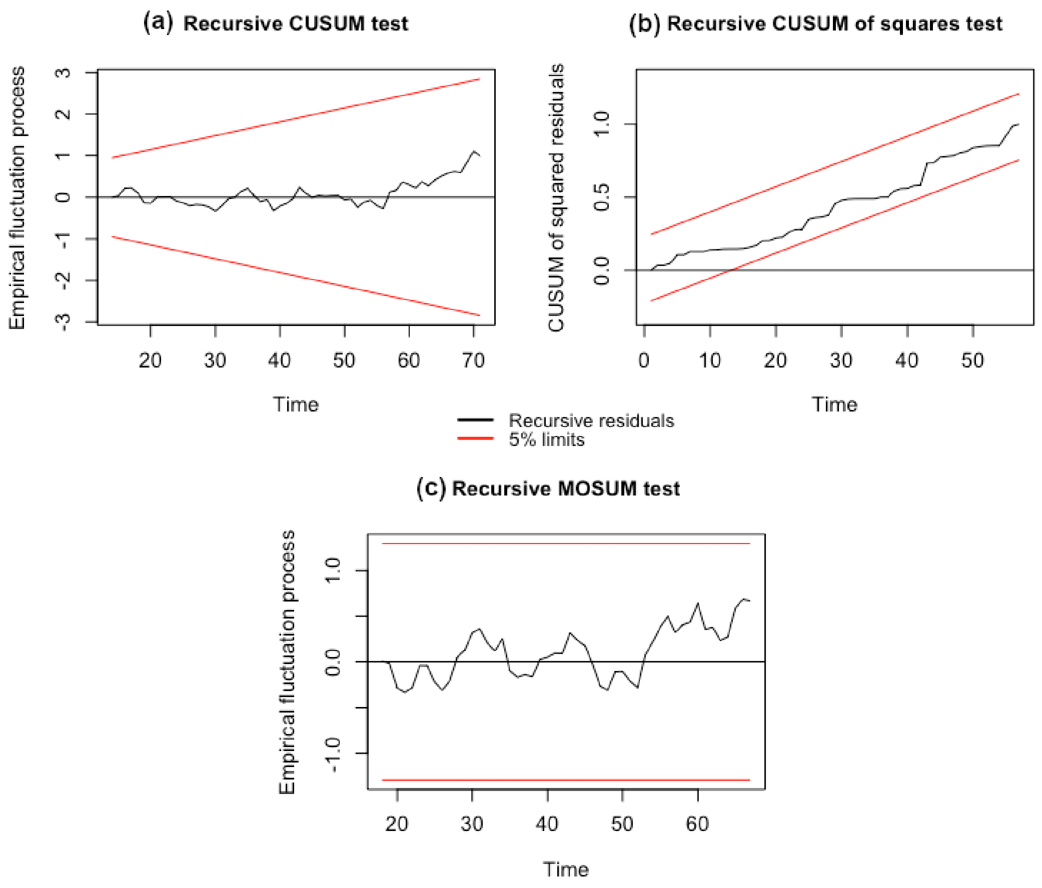

Figure S3. The CUSUM, CUSUMSQ, and MOSUM plots of recursive residuals for Case 1. Flat black lines in the plots show the zero level.

References

Asteriou, D., Hall, S.G. (2016). Applied Econometrics (3rd ed.). New York: Palgrave Macmillan. pp. 159–61.

Demirhan, H. (2020). dLagM: An R package for distributed lag models and ARDL bounds testing. PLoS ONE. 15(2), e0228812. doi: 10.1371/journal.pone.0228812.

Pesaran, M.H., Shin, Y. and Smith, R.J. (2001). Bounds testing approaches to the analysis of level relationships. Journal of Applied Econometrics. 16(3), 289-326.

Zhai, S., Song, G., Qin, Y., Ye, X. and Lee, J. (2017). Modeling the impacts of climate change and technical progress on the wheat yield in inland China: An autoregressive distributed lag approach. PloS ONE. 12(9), e0184474.