Tra Vecchie e Nuove Fonti: Modellazione Econometrica del Mercato Energetico Italiano

Relazione per il corso di Modelli Statistici per Dati Economici

Giorgia Caicchiolo, Gabriele Paganelli

19 gennaio 2026

Indice

1 Introduzione

Il mercato elettrico italiano ha attraversato, nel quinquennio 2021-2025, una fase di straordinaria turbolenza. La crisi energetica globale, unita alla crescente importanza delle fonti rinnovabili, ha reso fondamentale comprendere i meccanismi di formazione del prezzo dell'energia elettrica.

Questo studio analizza le relazioni dinamiche tra tre variabili chiave estratte dai portali istituzionali GME e Terna. Il **Prezzo Unico Nazionale (PUN)** (https://www.mercatoelettrico.org/it-it/Home/Esiti/Elettricita/MGP/Statistiche/DatiStorici?.com)rappresenta il prezzo di riferimento dell'energia elettrica alla borsa italiana. Il **prezzo del Gas Naturale**(https://www.mercatoelettrico.org/en-us/Home/Results/Gas/MGS/StatisticalData/HistoricalData?.com) costituisce il principale driver del PUN, dato che l'Italia produce ancora una quota significativa di elettricità bruciando gas. Infine, l'**Energia da Fonti Rinnovabili** (https://dati.terna.it/en/download-center#/load/total-load) immessa in rete rappresenta la potenza media prodotta da fonti pulite che, avendo costi variabili quasi nulli, tendono ad abbassare il prezzo di borsa quando la loro disponibilità è elevata.

L'obiettivo è indagare come queste serie si influenzano reciprocamente, verificando l'esistenza di un legame stabile nel lungo periodo e misurando la velocità di reazione del sistema a variazioni improvvise della produzione rinnovabile o del costo del gas. L'analisi procede attraverso modelli univariati SARIMA per isolare le componenti stocastiche, test di cointegrazione per verificare i legami di lungo periodo, transfer function models per studiare le risposte dinamiche, e modelli VAR per catturare le interdipendenze multivariate del sistema.

2 Setup e Caricamento Dati

2.1 Librerie

```r
library(here)
library(forecast)
library(dplyr)
library(readxl)
library(lubridate)
library(tidyverse)
library(vars)
library(dynlm)
library(urca)
library(lmtest)
library(MTS)
library(knitr)
library(kableExtra)

knitr::opts_chunk$set(
  dpi = 300,
  fig.align = "center",
  out.width = "100%"
)

options(digits = 4)
options(knitr.table.format = "latex")
```

2.2 Funzioni Diagnostiche

```r
acf_plot = function(mod) {
  par(mfcol = c(1, 2))
  acf(residuals(mod), main = "ACF Residui", col = "steelblue", lwd = 2)
  pacf(residuals(mod), main = "PACF Residui", col = "coral", lwd = 2)
  par(mfcol = c(1, 1))
}

ljung_box_plot = function(res, max_lag, fitdf = 0) {
  pvals = sapply(1:max_lag, function(l)
    Box.test(res, lag = l, type = "Ljung", fitdf = fitdf)$p.value)
  plot(pvals, type = "b", ylim = c(0, 1), ylab = "p-value",
       xlab = "Lag", main = "Ljung-Box Test", pch = 19,
       col = "steelblue", lwd = 2)
  abline(h = 0.05, col = "red", lty = 2, lwd = 2)
  grid(col = "gray90")
}
```

2.3 Importazione e Preprocessing dei Dati

I dati grezzi vengono standardizzati attraverso allineamento temporale, conversione delle unità in €/MWh e MW, trattamento di outlier e valori mancanti. Le serie PUN e Rinnovabili sono

originariamente a frequenza di 15 minuti, mentre il Gas ha frequenza giornaliera, rendendo necessario l'allineamento temporale mediante aggregazione giornaliera delle prime due serie.

```
prezzo_gas = bind_rows(
  read_excel(here("Annotermico_2020-2021_09.xlsx"),
             sheet = "MGP-GAS - Negoziazione continua", skip = 5)[-(1:92), 1:2] %>%
    set_names(c("data", "prezzo")),
  read_excel(here("Annotermico_2021-2022_09.xlsx"),
             sheet = "MGP-GAS - Negoziazione continua", skip = 5)[, 1:2] %>%
    set_names(c("data", "prezzo")),
  read_excel(here("Annotermico_2022-2023_09.xlsx"),
             sheet = "MGP-GAS - Negoziazione continua", skip = 5)[, 1:2] %>%
    set_names(c("data", "prezzo")),
  read_excel(here("Annotermico_2023-2024_09.xlsx"),
             sheet = "MGP-GAS - Negoziazione continua", skip = 5)[, 1:2] %>%
    set_names(c("data", "prezzo")),
  read_excel(here("Annotermico_2024-2025_09.xlsx"),
             sheet = "MGP-GAS - Negoziazione continua", skip = 5)[, 1:2] %>%
    set_names(c("data", "prezzo")),
  read_excel(here("Annotermico_2025-2026_12.xlsx"),
             sheet = "MGP-GAS - Negoziazione continua", skip = 5)[, 1:2] %>%
    set_names(c("data", "prezzo"))) %>%
  mutate(data = as.Date(data), prezzo = as.numeric(prezzo))
```

```
files_pun = c("Anno 2021_12.xlsx", "Anno 2022_12.xlsx", "Anno 2023_12.xlsx",
              "Anno 2024_12.xlsx", "Anno 2025_12.xlsx")

all_pun = bind_rows(lapply(files_pun, function(file) {
  df = read_excel(here(file), sheet = "Prezzi-Prices")

  if(is.character(df[[1]])) {
    df[[1]] = ymd(df[[1]])
  } else if(is.numeric(df[[1]])) {
    if(max(df[[1]], na.rm = TRUE) > 1000000) {
      df[[1]] = ymd(as.character(df[[1]]))
    } else {
      df[[1]] = as.Date(df[[1]], origin = "1899-12-30")
    }
  } else {
    df[[1]] = as.Date(df[[1]])
  }

  df[[2]] = as.numeric(df[[2]])
  df[[3]] = as.numeric(df[[3]])

  return(df[, c(1, 2, 3)])
}))
```

```r
names(all_pun) = c("data", "ora", "PUN")

pun_series = all_pun %>%
  group_by(data) %>%
  summarise(prezzo = mean(PUN, na.rm = TRUE)) %>%
  arrange(data)
```

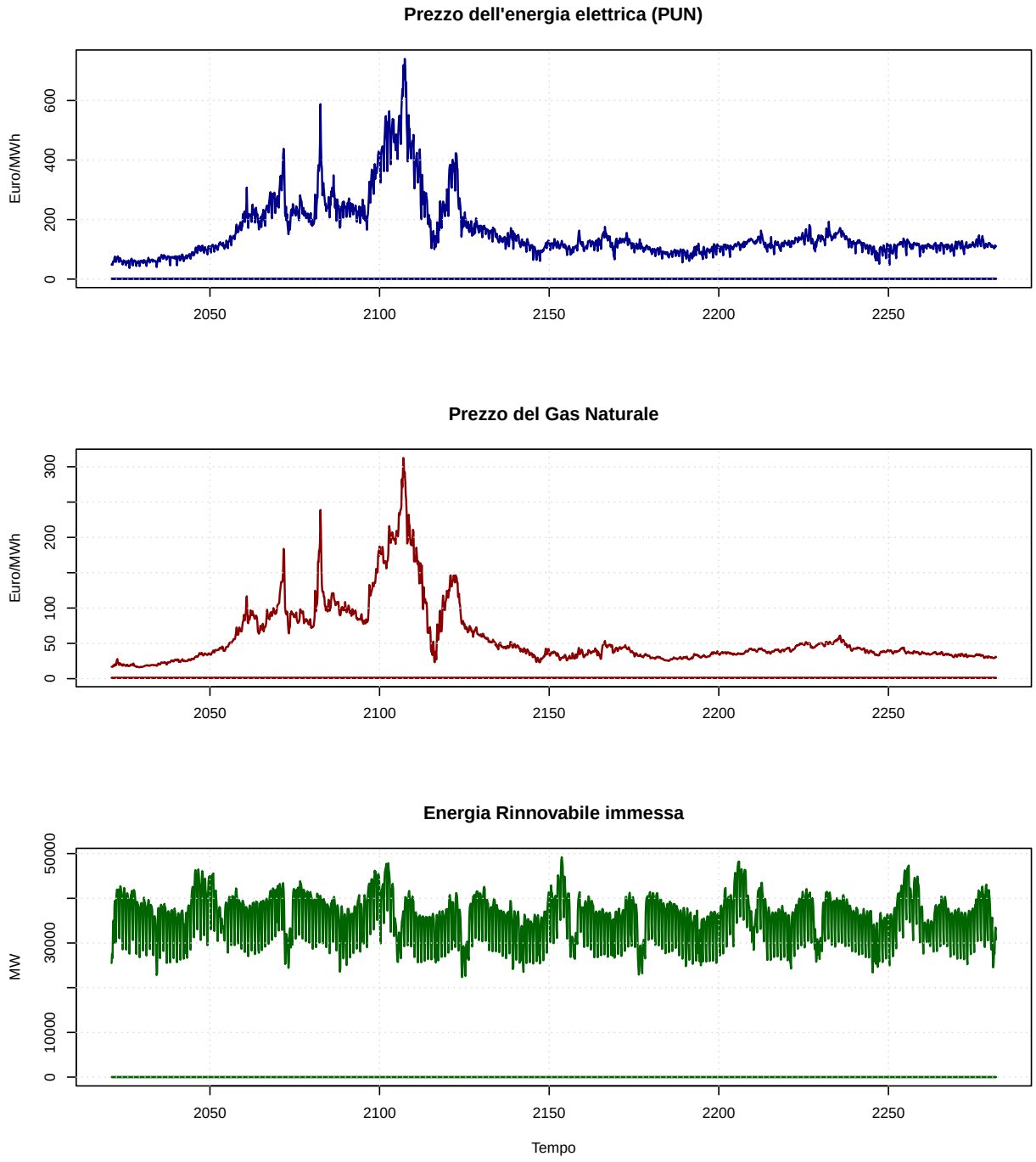
```r
files_renew = c("Rinnovabili 2021.xlsx", "Rinnovabili 2022.xlsx",
                "Rinnovabili 2023.xlsx", "Rinnovabili 2024.xlsx",
                "Rinnovabili 2025.xlsx")

all_renew = lapply(files_renew, function(f) {
  df = read_excel(here(f))
  if(!all(c("Date", "Total Load [MW]") %in% names(df))) {
    stop("Colonne mancanti in ", f)
  }
  df %>%
    dplyr::select(Date, `Total Load [MW]`) %>%
    mutate(Date = as.POSIXct(Date))
}) %>% bind_rows()

renew_series = all_renew %>%
  filter(!is.na(Date)) %>%
  mutate(data = as.Date(Date)) %>%
  group_by(data) %>%
  summarise(prezzo = mean(`Total Load [MW]`, na.rm = TRUE)) %>%
  arrange(data)
```

2.4 Creazione delle Serie Temporali

```r
gas_ts = ts(prezzo_gas$prezzo, start = c(2021, 1), frequency = 7)
pun_ts = ts(pun_series$prezzo, start = c(2021, 1), frequency = 7)
renew_ts = ts(renew_series$prezzo, start = c(2021, 1), frequency = 7)
```

```r
par(mfrow = c(3, 1))

ts.plot(pun_ts, main = "Prezzo dell'energia elettrica (PUN)",
        ylab = "Euro/MWh", xlab = "", col = "darkblue", lwd = 1.5,
        cex.main = 1.1, cex.lab = 1.05)
grid(col = "gray85", lty = "dotted")

ts.plot(gas_ts, main = "Prezzo del Gas Naturale",
        ylab = "Euro/MWh", xlab = "", col = "darkred", lwd = 1.5,
        cex.main = 1.1, cex.lab = 1.05)
grid(col = "gray85", lty = "dotted")

ts.plot(renew_ts, main = "Energia Rinnovabile immessa",
```

```
        ylab = "MW", xlab = "Tempo", col = "darkgreen", lwd = 1.5,
        cex.main = 1.1, cex.lab = 1.05)
grid(col = "gray85", lty = "dotted")
```




```
par(mfrow = c(1, 1))
```

L'analisi visiva delle serie storiche rivela le dinamiche che hanno caratterizzato il mercato energetico italiano nel periodo considerato.

La crisi energetica del 2022 emerge chiaramente nei grafici del PUN e del Gas: le tensioni geopolitiche internazionali hanno provocato un'impennata senza precedenti del prezzo del gas, trascinando con sé il prezzo dell'elettricità. Il PUN ha toccato punte superiori ai 700 €/MWh, livelli quasi dieci volte superiori alle medie storiche precedenti, con una volatilità estrema visibile nei continui salti dei prezzi.

Dal 2023 si osserva un graduale rientro verso una nuova stabilità. La diversificazione degli approvvigionamenti di gas e una domanda più contenuta hanno favorito un calo drastico rispetto ai picchi della crisi, pur senza tornare ai livelli pre-2021. Le serie si sono assestate su una fascia di prezzo più elevata, ancora sensibile a eventuali shock di mercato.

La serie dell'energia rinnovabile mostra un diverso andamento, scandito dai ritmi meteorologici. Le oscillazioni più ampie riflettono la stagionalità climatica, con maggiore produzione solare nei mesi estivi. Le fluttuazioni più ravvicinate derivano principalmente dalla variabilità delle condizioni meteo in giorni diversi che influenzano la disponibilità delle fonti rinnovabili.

Infine, il movimento quasi sincronizzato delle prime due serie testimonia il legame strutturale tra gas ed elettricità: il gas rimane la fonte principale per garantire la continuità elettrica in Italia, e quando il suo costo aumenta, il prezzo dell'elettricità segue immediatamente, suggerendo una possibile relazione di cointegrazione.

3 Modelli Univariati SARIMA

3.1 Trasformazione Logaritmica

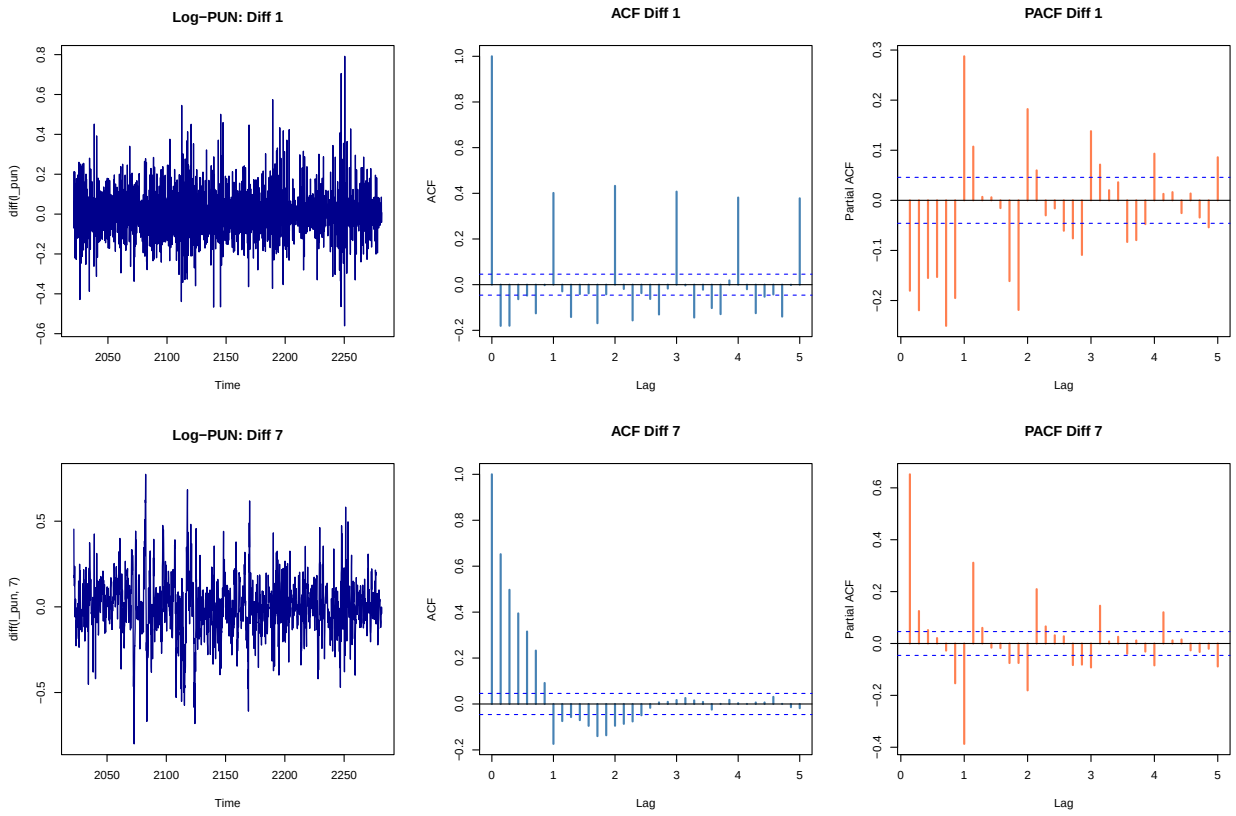
Per stabilizzare la varianza e rendere le serie più adatte alla modellazione, applichiamo una trasformazione logaritmica a tutte le serie. Questa operazione mitiga l'impatto dei valori estremi registrati durante lo shock del 2022, rendendo la variabilità più omogenea nel tempo e le assunzioni dei modelli ARIMA più plausibili. Inoltre, consente di ragionare in termini di variazioni percentuali anziché assolute, interpretazione più significativa dal punto di vista economico.

```
l_pun = log(pun_ts)
l_gas = log(gas_ts)
l_renew = log(renew_ts)
```

3.2 Analisi della Stazionarietà

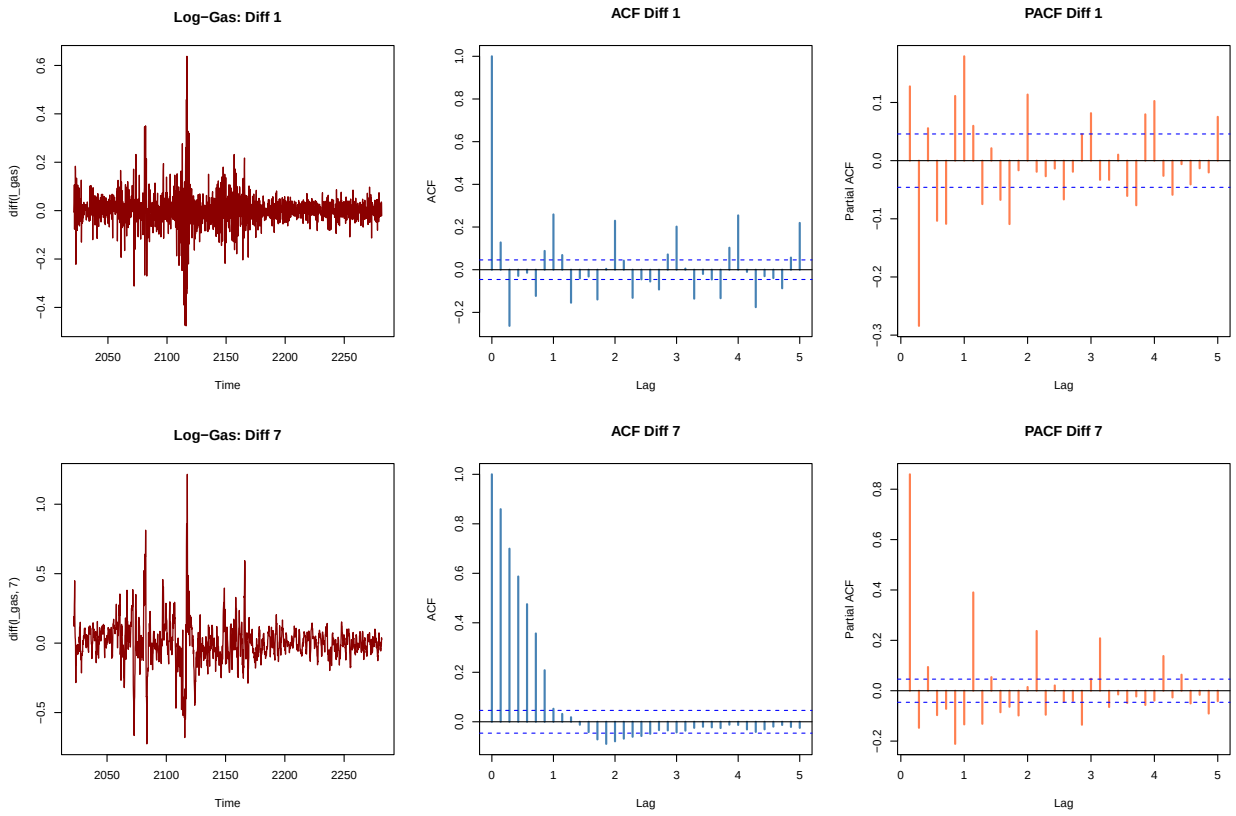
Analizziamo le funzioni di autocorrelazione (ACF) e autocorrelazione parziale (PACF) dopo differenziazione ordinaria e stagionale per identificare la struttura ARIMA appropriata.

```
par(mfrow = c(2, 3))
plot.ts(diff(l_pun), main = "Log-PUN: Diff 1", col = "darkblue", lwd = 1.2)
acf(diff(l_pun), lag.max = 35, main = "ACF Diff 1", col = "steelblue", lwd = 2)
pacf(diff(l_pun), lag.max = 35, main = "PACF Diff 1", col = "coral", lwd = 2)
plot.ts(diff(l_pun, 7), main = "Log-PUN: Diff 7", col = "darkblue", lwd = 1.2)
acf(diff(l_pun, 7), lag.max = 35, main = "ACF Diff 7", col = "steelblue", lwd = 2)
pacf(diff(l_pun, 7), lag.max = 35, main = "PACF Diff 7", col = "coral", lwd = 2)
```



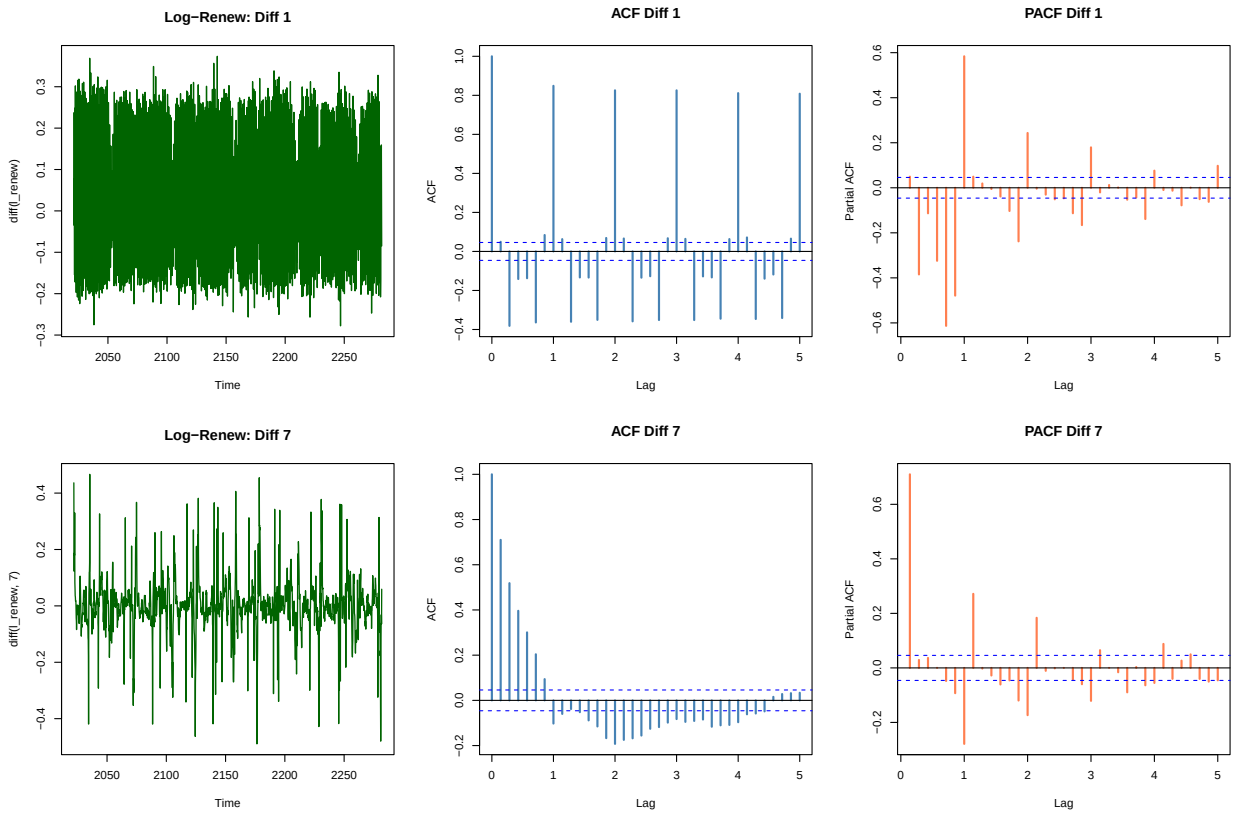

```r
par(mfrow = c(1, 1))
```

```r
par(mfrow = c(2, 3))
plot.ts(diff(l_gas), main = "Log-Gas: Diff 1", col = "darkred", lwd = 1.2)
acf(diff(l_gas), lag.max = 35, main = "ACF Diff 1", col = "steelblue", lwd = 2)
pacf(diff(l_gas), lag.max = 35, main = "PACF Diff 1", col = "coral", lwd = 2)
plot.ts(diff(l_gas, 7), main = "Log-Gas: Diff 7", col = "darkred", lwd = 1.2)
acf(diff(l_gas, 7), lag.max = 35, main = "ACF Diff 7", col = "steelblue", lwd = 2)
pacf(diff(l_gas, 7), lag.max = 35, main = "PACF Diff 7", col = "coral", lwd = 2)
```




```
par(mfrow = c(1, 1))
```

```
par(mfrow = c(2, 3))
plot.ts(diff(l_renew), main = "Log-Renew: Diff 1", col = "darkgreen", lwd = 1.2)
acf(diff(l_renew), lag.max = 35, main = "ACF Diff 1", col = "steelblue", lwd = 2)
pacf(diff(l_renew), lag.max = 35, main = "PACF Diff 1", col = "coral", lwd = 2)
plot.ts(diff(l_renew, 7), main = "Log-Renew: Diff 7", col = "darkgreen", lwd = 1.2)
acf(diff(l_renew, 7), lag.max = 35, main = "ACF Diff 7", col = "steelblue", lwd = 2)
pacf(diff(l_renew, 7), lag.max = 35, main = "PACF Diff 7", col = "coral", lwd = 2)
```




```
par(mfrow = c(1, 1))
```

L'analisi dei grafici ACF e PACF per le serie trasformate in logaritmi determina gli ordini di integrazione ordinaria (d) e stagionale (D) necessari per la stazionarietà.

Per il **Prezzo dell'energia (PUN)**, la differenza ordinaria ($d = 1$) elimina il trend di fondo centrando la serie sullo zero, ma l'ACF presenta picchi significativi e persistenti al lag 7 e suoi multipli, evidenza di stagionalità settimanale non risolta. La differenza stagionale a lag 7 riduce questa persistenza, suggerendo che la componente stagionale richieda parametri autoregressivi e di media mobile stagionali.

Il **Prezzo del Gas** mostra dinamiche simili. La differenza prima è necessaria ma non sufficiente per stabilizzare la media della serie. La struttura stagionale settimanale, sebbene meno marcata rispetto al PUN, risulta comunque presente e richiede trattamento attraverso differenziazione stagionale.

L'**Energia Rinnovabile** presenta caratteristiche diverse. La serie originale è dominata da un ciclo settimanale quasi deterministico, e l'applicazione della differenza stagionale ($D = 1$) stabilizza visibilmente i dati. A differenza dei prezzi, la produzione rinnovabile non mostra un trend stocastico di lungo periodo ma oscilla attorno a livelli medi relativamente stabili una volta rimossa la componente settimanale, suggerendo $d = 0$.

3.3 Stima dei Modelli SARIMA

3.3.1 Modello PUN

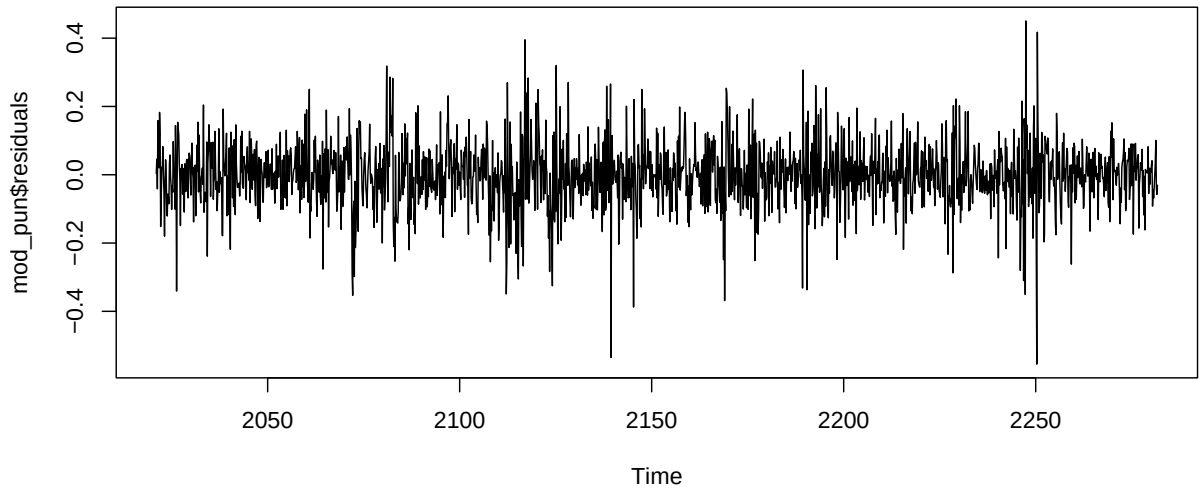
Per il PUN stimiamo un modello SARIMA(0,1,2)(1,0,1)[7]:

$$(1-\Phi_1 B^7)(1-B)\log(\text{PUN}_t) = (1+\theta_1 B+\theta_2 B^2)(1+\Theta_1 B^7)\varepsilon_t$$

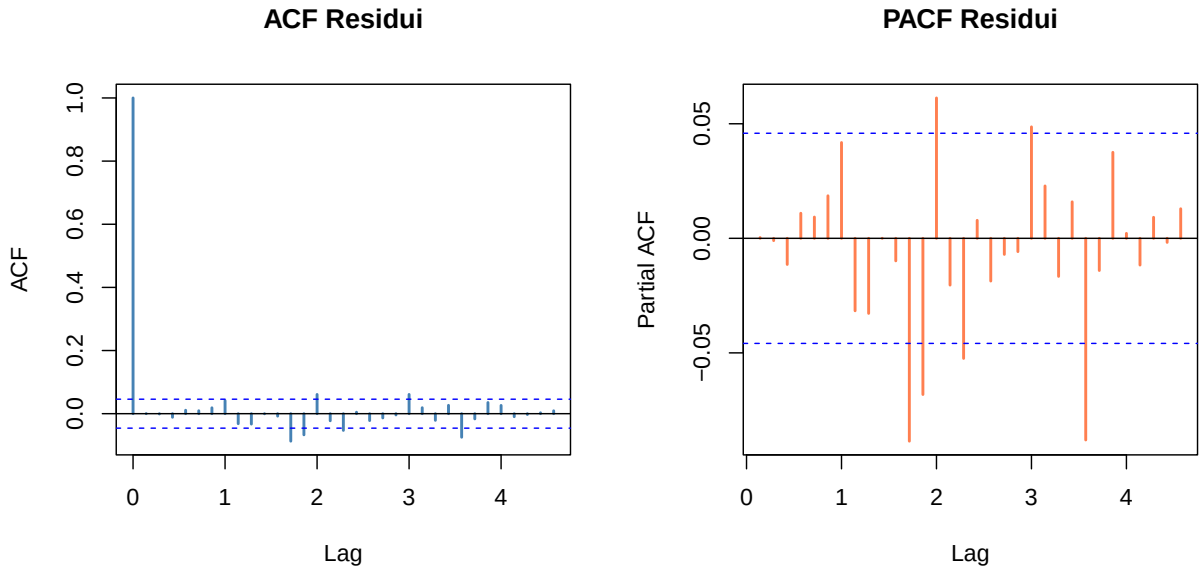
```
mod_pun = Arima(l_pun, order = c(0, 1, 2), seasonal = list(order = c(1, 0, 1), period = 7))
summary(mod_pun)
```

```
## Series: l_pun
## ARIMA(0,1,2)(1,0,1)[7]
##
## Coefficients:
##          ma1     ma2  sar1    sma1
##       -0.351  -0.100     1  -0.982
## s.e.   0.023   0.023     0   0.007
##
## sigma^2 = 0.0089:  log likelihood = 1711
## AIC=-3412   AICc=-3412   BIC=-3385
##
## Training set error measures:
##                       ME   RMSE     MAE      MPE  MAPE   MASE      ACF1
## Training set -0.001355 0.0942 0.06921 -0.05031 1.425 0.5798 0.0003403
```

```
plot.ts(mod_pun$residuals)
```

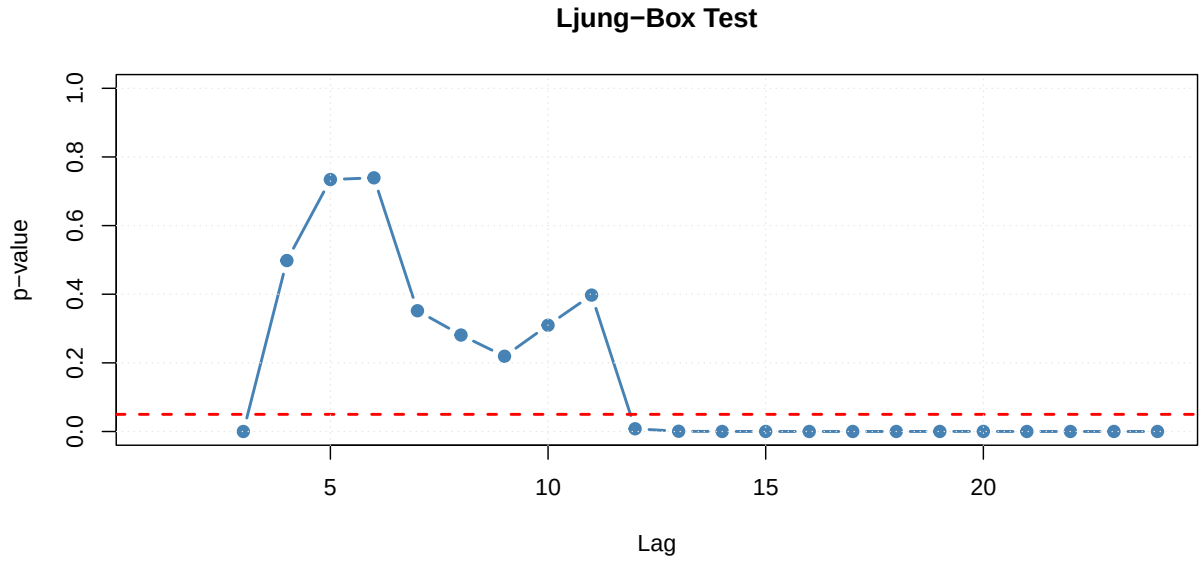



```
acf_plot(mod_pun)
```




```
ljung_box_plot(mod_pun$residuals, 24, fitdf = 3)
```

```
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
```



Il modello SARIMA(0,1,2)(1,0,1)[7] è stato selezionato come migliore specificazione secondo la procedura iterativa Box-Jenkins. Tuttavia, i test diagnostici sui residui (ACF, PACF e Ljung-Box) evidenziano persistenza di struttura non catturata. Il parametro della componente media mobile stagionale risulta vicino a 1 in valore assoluto, segnalando possibile instabilità numerica o una stagionalità settimanale particolarmente rigida.

Le performance non ottimali derivano principalmente dalla forte eteroschedasticità residua e dagli shock esogeni del 2021-2022, che introducono variabilità difficilmente assorbibile da modelli lineari

SARIMA. Nonostante questi limiti, il modello rappresenta una prima approssimazione utile del comportamento univariato della serie e fornisce una base per le analisi successive più sofisticate.

3.3.2 Modello Gas

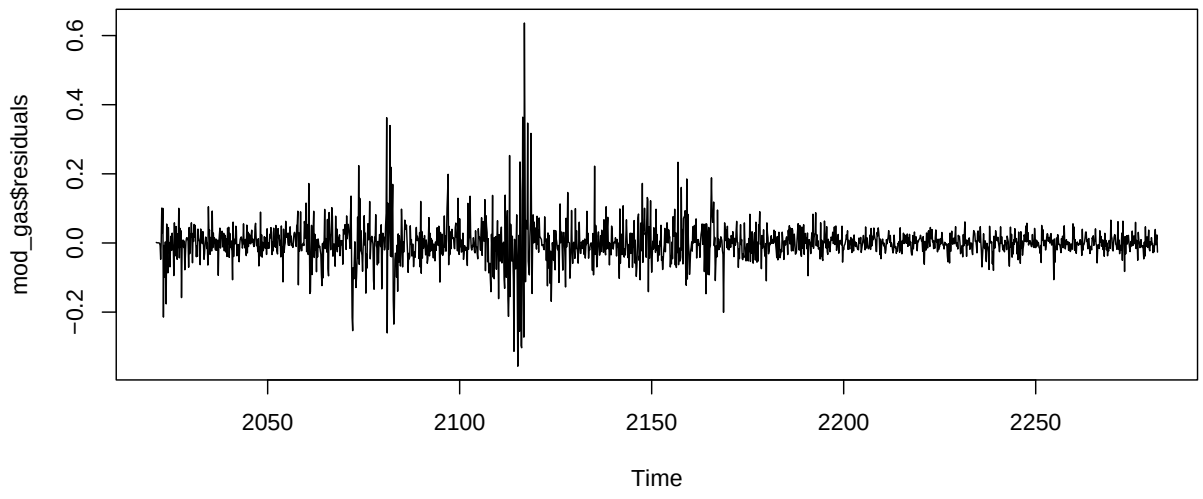
Per il Gas stimiamo un modello SARIMA(0,1,2)(0,1,1)[7]:

$$(1-B)(1-B^7)\log(\text{Gas}_t) = (1+\theta_1 B+\theta_2 B^2)(1+\Theta_1 B^7)\varepsilon_t$$

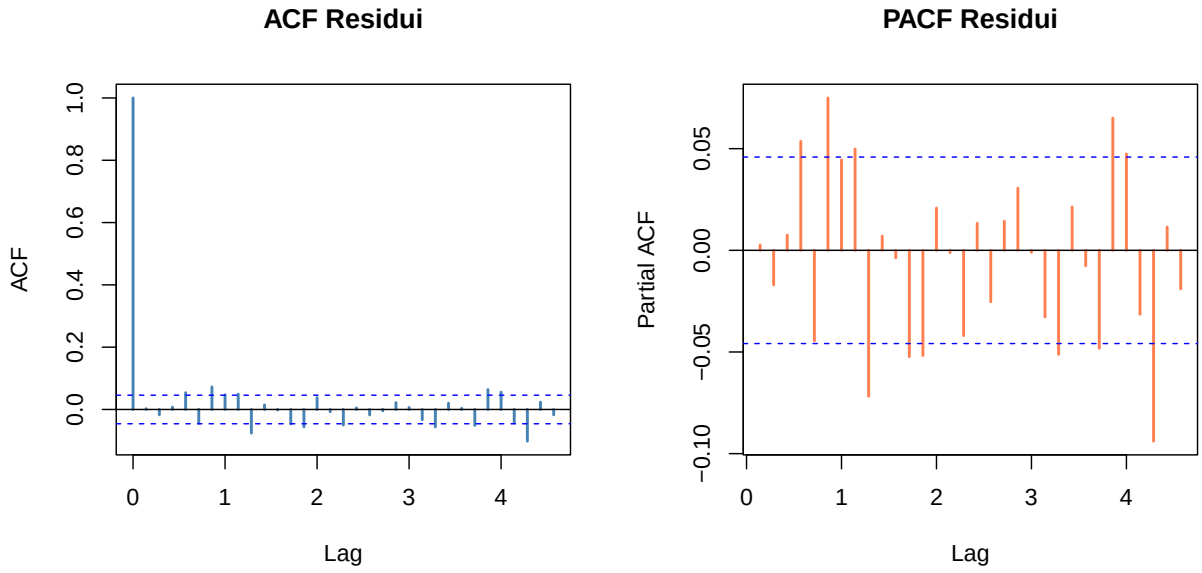
```
mod_gas = Arima(l_gas, order = c(0, 1, 2), seasonal = list(order = c(0, 1, 1), period = 7))
summary(mod_gas)
```

```
## Series: l_gas
## ARIMA(0,1,2)(0,1,1)[7]
##
## Coefficients:
##          ma1     ma2    sma1
##       0.140  -0.200  -0.954
## s.e.  0.023   0.021   0.011
##
## sigma^2 = 0.00315:  log likelihood = 2649
## AIC=-5289   AICc=-5289   BIC=-5267
##
## Training set error measures:
##                       ME    RMSE    MAE      MPE  MAPE   MASE    ACF1
## Training set -0.0005468 0.05598 0.0338 -0.01921 0.866 0.3553 0.00264
```

```
plot.ts(mod_gas$residuals)
```

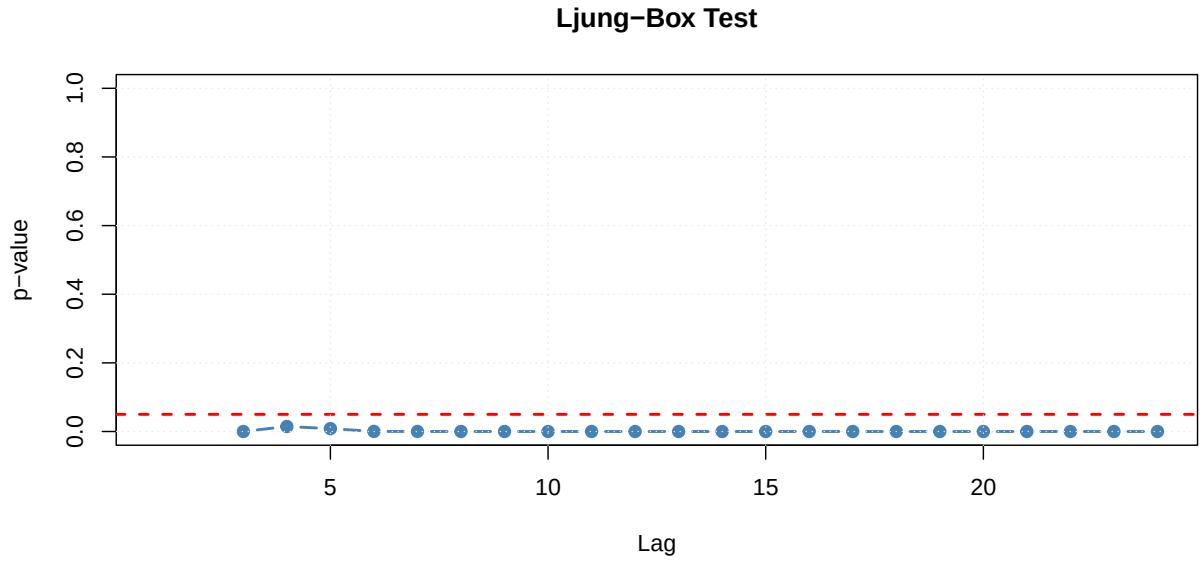



```
acf_plot(mod_gas)
```




```
ljung_box_plot(mod_gas$residuals, 24, fitdf = 3)
```

```
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
```



Anche il modello del Gas presenta diagnostiche non pienamente soddisfacenti, con residui che mostrano struttura autocorrelata residua. Il parametro della componente media mobile stagionale assume un valore prossimo a 1 in modulo, come nel modello SARIMA per PUN. L'estrema eteroschedasticità del periodo e i cambiamenti strutturali post-2022 limitano l'efficacia della modellazione lineare, ma, anche qui, il SARIMA$(0,1,2)(0,1,1)[7]$ fornisce comunque la migliore approssimazione disponibile per questa fase preliminare.

3.3.3 Modello Rinnovabili

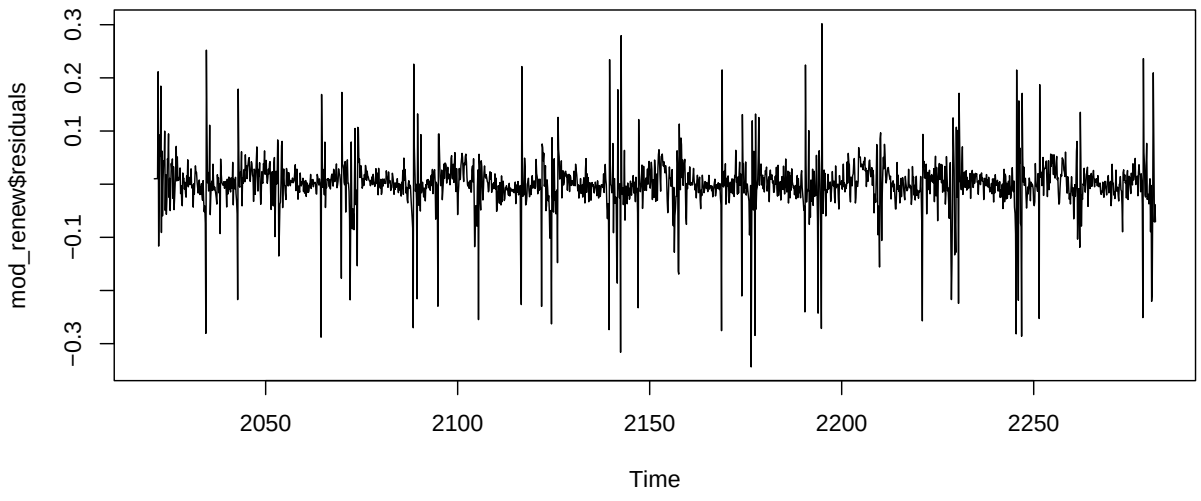
Per le Rinnovabili stimiamo un modello SARIMA(1,0,0)(1,1,1)[7]:

$$(1-\phi_1 B)(1-\Phi_1 B^7)(1-B^7)\log(\text{Renew}_t) = (1+\Theta_1 B^7)\varepsilon_t$$

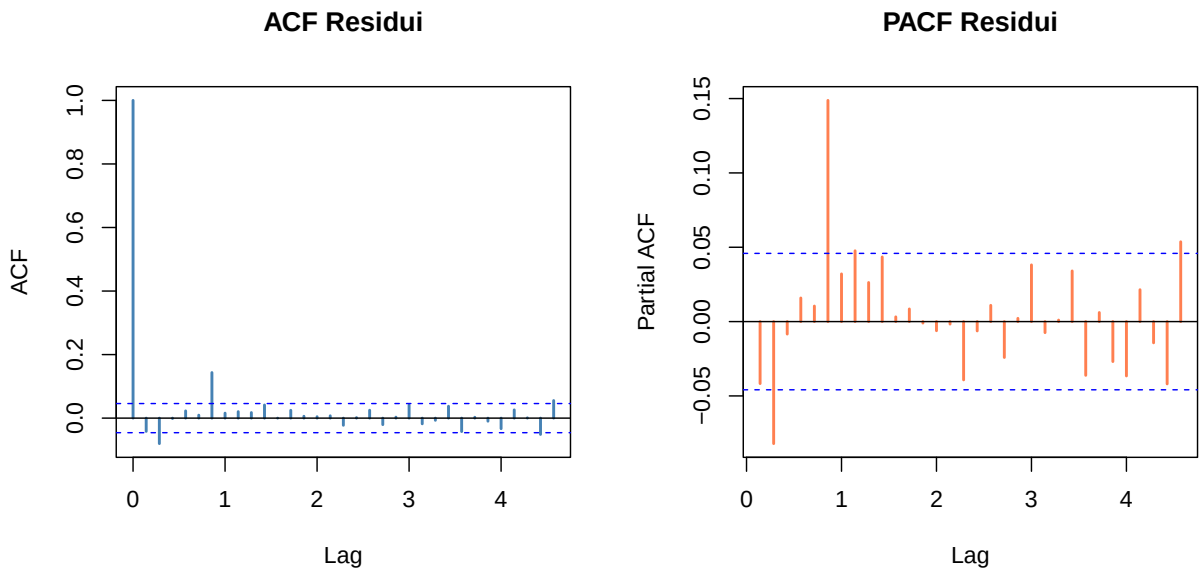
```
mod_renew = Arima(l_renew, order = c(1, 0, 0), seasonal = list(order = c(1, 1, 1), period = 7))
summary(mod_renew)
```

```
## Series: l_renew
## ARIMA(1,0,0)(1,1,1)[7]
##
## Coefficients:
##         ar1   sar1    sma1
##       0.837  0.168  -0.993
## s.e.  0.014  0.025   0.006
##
## sigma^2 = 0.00277:  log likelihood = 2763
## AIC=-5517   AICc=-5517   BIC=-5495
##
## Training set error measures:
##                       ME    RMSE     MAE       MPE   MAPE   MASE     ACF1
## Training set -0.0003786 0.05248 0.02938 -0.005904 0.2823 0.4892 -0.04169
```

```
plot.ts(mod_renew$residuals)
```

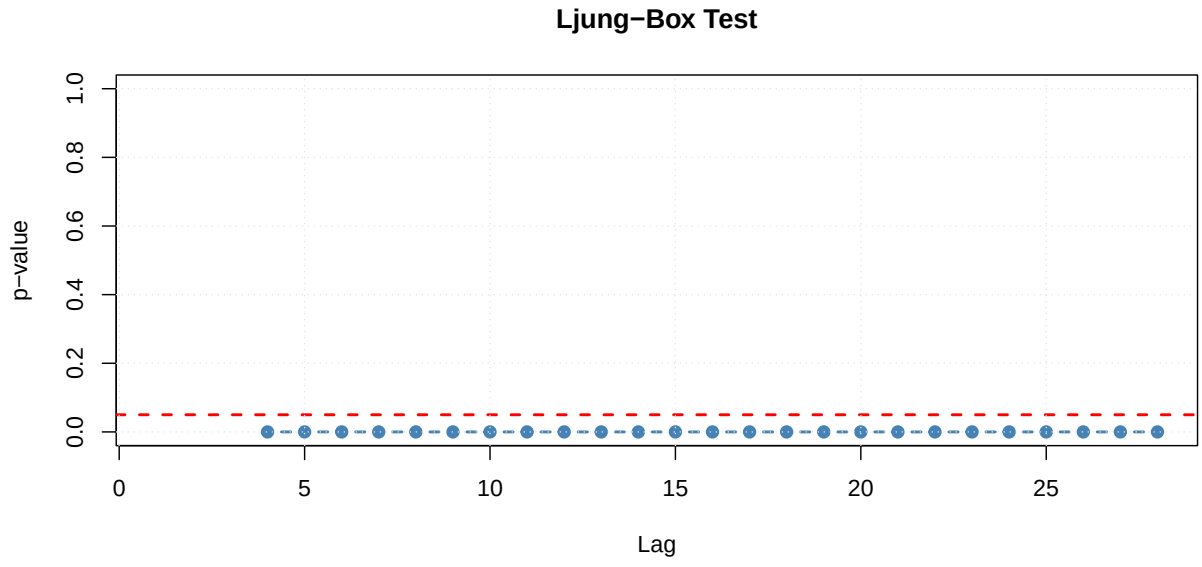


```
acf_plot(mod_renew)
```




```
ljung_box_plot(mod_renew$residuals, 28, fitdf = 4)
```

```
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
```



Il modello per le rinnovabili presenta il termine a media mobile stagionale prossimo a 1 in modulo, e autocorrelazione residua persistente. Il limite principale risiede probabilmente nella presenza di una stagionalità climatica annuale (inverno-estate) non catturabile con la frequenza settimanale adottata. Questa scelta di modellazione, pur impedendo di cogliere variazioni stagionali di lungo periodo,

è funzionale all'analisi di breve termine e alla successiva integrazione con le altre serie. I residui eteroschedastici riflettono inoltre cambiamenti strutturali nel mix rinnovabile degli anni 2021-2025, confermando come i modelli lineari forniscano una base interpretativa utile ma non esaustiva della complessità dei dati reali.

4 Analisi di Cointegrazione

L'analisi condotta finora ha evidenziato come le serie dei prezzi (PUN e Gas) condividano trend simili, suggerendo un legame di lungo periodo. In questa sezione estendiamo l'indagine per verificare se sia possibile includere anche la serie delle Rinnovabili in un'analisi di cointegrazione a tre variabili, o se la relazione di equilibrio sia limitata alla sola coppia dei prezzi.

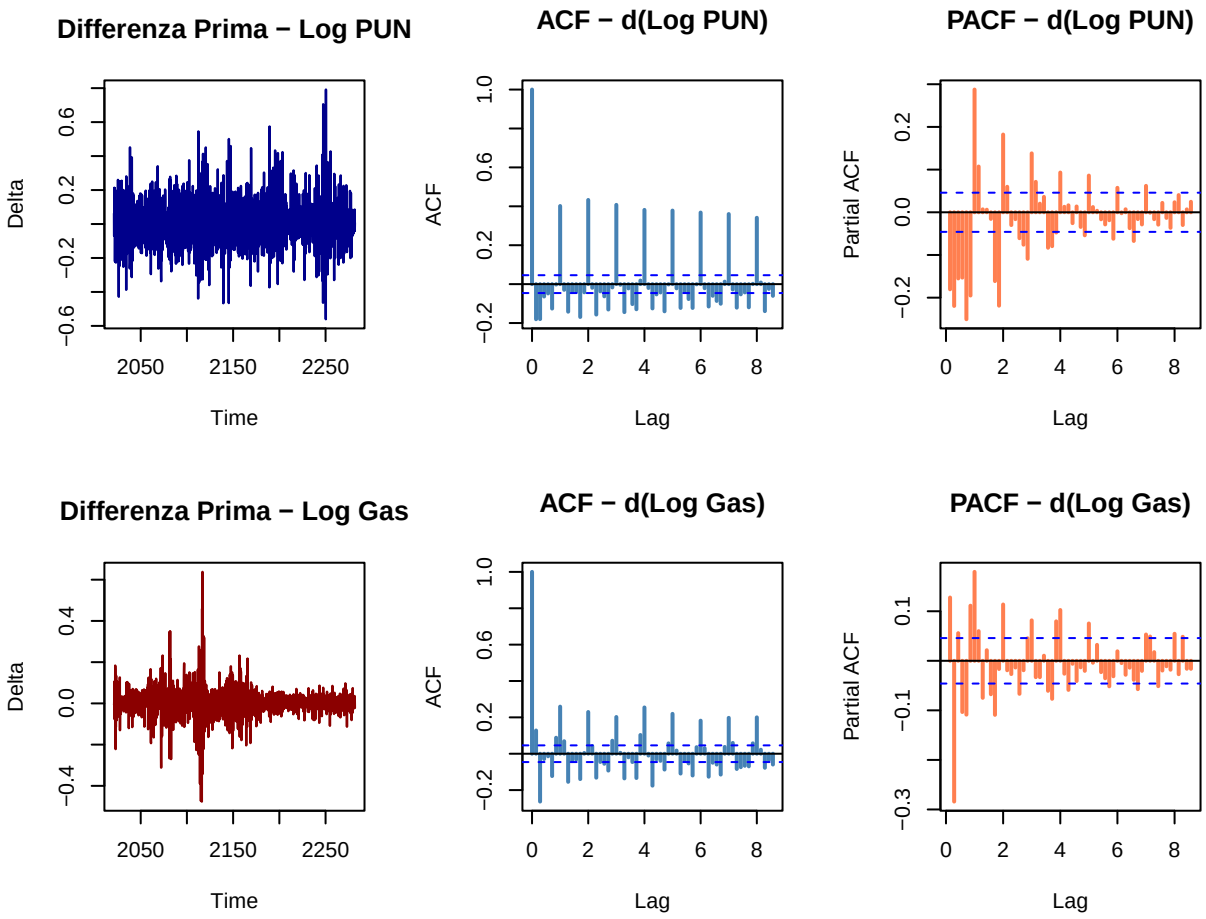
L'obiettivo è determinare se esista un equilibrio stabile che leghi il costo delle materie prime fossili e l'immissione di energia pulita alla formazione del prezzo elettrico nazionale. Il test di Johansen permetterà di identificare il rango di cointegrazione e di stabilire con rigore statistico se il sistema converga verso un equilibrio comune. Questo approccio multivariato supera i limiti dei modelli univariati, rivelando se l'incremento delle rinnovabili eserciti una pressione strutturale sui prezzi o se il legame Gas-PUN rimanga la dinamica dominante.

4.1 Preparazione dei Dati

Costruiamo le matrici dei dati in livelli e in differenze prime per PUN e Gas.

```
data_levels = cbind(l_pun, l_gas)
colnames(data_levels) = c("l_PUN", "l_Gas")

data_diff = diff(data_levels)
colnames(data_diff) = c("d_PUN", "d_Gas")
```

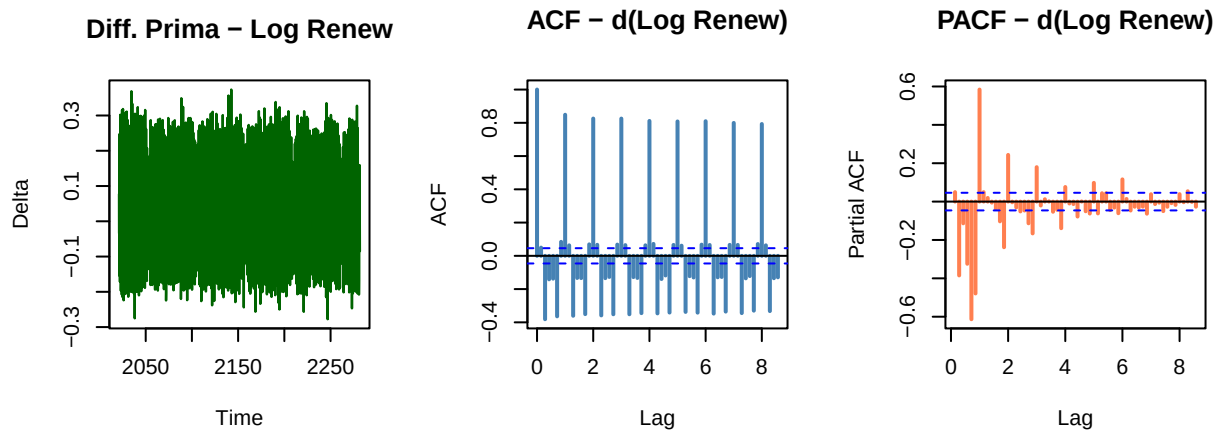
```
par(mfrow = c(2, 3))
plot(data_diff[, 1], main = "Differenza Prima - Log PUN", ylab = "Delta",
     col = "darkblue", lwd = 1.2)
acf(data_diff[, 1], lag.max = 60, main = "ACF - d(Log PUN)",
    col = "steelblue", lwd = 2)
pacf(data_diff[, 1], lag.max = 60, main = "PACF - d(Log PUN)",
     col = "coral", lwd = 2)

plot(data_diff[, 2], main = "Differenza Prima - Log Gas", ylab = "Delta",
     col = "darkred", lwd = 1.2)
acf(data_diff[, 2], lag.max = 60, main = "ACF - d(Log Gas)",
    col = "steelblue", lwd = 2)
pacf(data_diff[, 2], lag.max = 60, main = "PACF - d(Log Gas)",
     col = "coral", lwd = 2)
```




```
par(mfrow = c(1, 1))
```

L'analisi dei grafici conferma la stazionarietà delle serie differenziate, che oscillano attorno allo zero eliminando il trend. Tuttavia, i picchi ricorrenti nell'ACF al lag 7 segnalano una stagionalità settimanale ancora presente e strutturale. La volatilità sincrona nel 2022 suggerisce una risposta comune a shock esogeni, giustificando l'approccio multivariato per cercare un equilibrio di lungo periodo tra PUN e Gas.

```
data_diff_renew = diff(l_renew)

par(mfrow = c(1, 3))
plot(data_diff_renew, main = "Diff. Prima - Log Renew", ylab = "Delta",
     col = "darkgreen", lwd = 1.2)
acf(data_diff_renew, lag.max = 60, main = "ACF - d(Log Renew)",
    col = "steelblue", lwd = 2)
pacf(data_diff_renew, lag.max = 60, main = "PACF - d(Log Renew)",
     col = "coral", lwd = 2)
```




```
par(mfrow = c(1, 1))
```

La serie delle Rinnovabili presenta dinamiche radicalmente diverse dai prezzi energetici. A differenza di PUN e Gas, questa serie non mostra un trend stocastico di lungo periodo ma oscilla attorno a livelli fisicamente determinati. L'applicazione della differenza prima rischia di introdurre rumore superfluo, sovra-differenziando un processo intrinsecamente stazionario nei suoi livelli una volta depurato dalla stagionalità. Questa natura ciclica suggerisce un possibile rifiuto dell'ipotesi nulla di radice unitaria nei test ADF.

4.2 Test di Radice Unitaria (Augmented Dickey-Fuller)

4.2.1 Test ADF sul PUN

Verifichiamo la presenza di radici unitarie nel logaritmo del PUN attraverso la sequenza di test ADF.

```
test_pun_trend = ur.df(l_pun, type = "trend", lags = 7, selectlags = "BIC")
summary(test_pun_trend)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5193 -0.0570  0.0030  0.0597  0.4538
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.29e-02   2.69e-02    2.71  0.00674 **
```

```
## z.lag.1     -1.36e-02   5.20e-03   -2.62  0.00876 **
## tt          -5.84e-06   4.87e-06   -1.20  0.23053
## z.diff.lag1 -3.01e-01   2.28e-02  -13.20  < 2e-16 ***
## z.diff.lag2 -2.89e-01   2.37e-02  -12.17  < 2e-16 ***
## z.diff.lag3 -2.34e-01   2.42e-02   -9.66  < 2e-16 ***
## z.diff.lag4 -1.95e-01   2.43e-02   -8.02  1.9e-15 ***
## z.diff.lag5 -2.01e-01   2.41e-02   -8.34  < 2e-16 ***
## z.diff.lag6 -9.03e-02   2.35e-02   -3.84  0.00013 ***
## z.diff.lag7  2.87e-01   2.24e-02   12.79  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.107 on 1808 degrees of freedom
## Multiple R-squared:  0.279,  Adjusted R-squared:  0.276
## F-statistic: 77.9 on 9 and 1808 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -2.624 2.478 3.695
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

Il test tau3 non rifiuta l'ipotesi nulla di radice unitaria. I test phi2 e phi3 suggeriscono l'assenza di trend deterministico.

```
test_pun_drift = ur.df(l_pun, type = "drift", lags = 7, selectlags = "BIC")
summary(test_pun_drift)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5230 -0.0575  0.0025  0.0603  0.4555
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0616     0.0252    2.45  0.01446 *
```

```
## z.lag.1      -0.0124     0.0051   -2.44  0.01481 *
## z.diff.lag1  -0.3016     0.0228  -13.21  < 2e-16 ***
## z.diff.lag2  -0.2888     0.0237  -12.17  < 2e-16 ***
## z.diff.lag3  -0.2333     0.0242   -9.64  < 2e-16 ***
## z.diff.lag4  -0.1947     0.0243   -8.00  2.2e-15 ***
## z.diff.lag5  -0.2000     0.0241   -8.31  < 2e-16 ***
## z.diff.lag6  -0.0897     0.0235   -3.82  0.00014 ***
## z.diff.lag7   0.2876     0.0224   12.82  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.107 on 1809 degrees of freedom
## Multiple R-squared:  0.279,  Adjusted R-squared:  0.276
## F-statistic: 87.5 on 8 and 1809 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -2.439 2.997
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

Anche con drift, il test tau2 non rifiuta la radice unitaria e suggerisce l'assenza di drift.

```
test_pun_none = ur.df(l_pun, type = "none", lags = 7, selectlags = "BIC")
summary(test_pun_none)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5181 -0.0558  0.0039  0.0599  0.4570
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1     -1.88e-05   5.08e-04   -0.04     0.97
## z.diff.lag1 -3.11e-01   2.25e-02  -13.83  < 2e-16 ***
## z.diff.lag2 -2.97e-01   2.35e-02  -12.63  < 2e-16 ***
## z.diff.lag3 -2.40e-01   2.41e-02   -9.98  < 2e-16 ***
```

```
## z.diff.lag4 -2.00e-01   2.43e-02   -8.25  3.1e-16 ***
## z.diff.lag5 -2.04e-01   2.40e-02   -8.49  < 2e-16 ***
## z.diff.lag6 -9.23e-02   2.35e-02   -3.93  8.9e-05 ***
## z.diff.lag7  2.86e-01   2.24e-02   12.75  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.107 on 1810 degrees of freedom
## Multiple R-squared:  0.277,  Adjusted R-squared:  0.273
## F-statistic: 86.5 on 8 and 1810 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -0.0369
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
summary(ur.df(diff(l_pun), type = "drift", lags = 7, selectlags = "BIC"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5317 -0.0569  0.0011  0.0592  0.4461
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.000489   0.002496    0.20   0.8448
## z.lag.1     -1.832458   0.118013  -15.53  < 2e-16 ***
## z.diff.lag1  0.491225   0.110559    4.44  9.4e-06 ***
## z.diff.lag2  0.205511   0.097669    2.10   0.0355 *
## z.diff.lag3 -0.012011   0.083083   -0.14   0.8851
## z.diff.lag4 -0.189980   0.068240   -2.78   0.0054 **
## z.diff.lag5 -0.366987   0.052993   -6.93  6.0e-12 ***
## z.diff.lag6 -0.427170   0.037814  -11.30  < 2e-16 ***
## z.diff.lag7 -0.106965   0.023303   -4.59  4.7e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.106 on 1808 degrees of freedom
## Multiple R-squared:  0.697,  Adjusted R-squared:  0.696
## F-statistic:  520 on 8 and 1808 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -15.53 120.6
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

Il test sulla serie differenziata rifiuta l'ipotesi di radice unitaria, confermando che il PUN è I(1) senza trend né drift.

4.2.2 Test ADF sul Gas

Applichiamo la stessa procedura al logaritmo del prezzo del gas.

```
test_gas_trend = ur.df(l_gas, type = "trend", lags = 7, selectlags = "BIC")
summary(test_gas_trend)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3460 -0.0218  0.0001  0.0221  0.6244
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.96e-02   1.04e-02    2.85  0.00441 **
## z.lag.1     -6.67e-03   2.44e-03   -2.73  0.00635 **
## tt          -3.95e-06   2.67e-06   -1.48  0.13992
## z.diff.lag1  1.71e-01   2.31e-02    7.38  2.3e-13 ***
## z.diff.lag2 -2.81e-01   2.34e-02  -12.02  < 2e-16 ***
## z.diff.lag3  4.50e-02   2.42e-02    1.86  0.06324 .
## z.diff.lag4 -5.29e-02   2.42e-02   -2.19  0.02886 *
## z.diff.lag5 -7.15e-02   2.42e-02   -2.96  0.00314 **
## z.diff.lag6  7.87e-02   2.33e-02    3.38  0.00075 ***
```

```
## z.diff.lag7  1.81e-01   2.31e-02    7.83  8.0e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0579 on 1808 degrees of freedom
## Multiple R-squared:  0.162,  Adjusted R-squared:  0.158
## F-statistic:   39 on 9 and 1808 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -2.732 2.738 4.096
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

```
test_gas_drift = ur.df(l_gas, type = "drift", lags = 7, selectlags = "BIC")
summary(test_gas_drift)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3450 -0.0226 -0.0002  0.0215  0.6261
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02263    0.00925    2.45  0.01452 *
## z.lag.1     -0.00581    0.00237   -2.45  0.01435 *
## z.diff.lag1  0.17111    0.02312    7.40  2.1e-13 ***
## z.diff.lag2 -0.28063    0.02338  -12.00  < 2e-16 ***
## z.diff.lag3  0.04563    0.02422    1.88  0.05974 .
## z.diff.lag4 -0.05222    0.02417   -2.16  0.03086 *
## z.diff.lag5 -0.07080    0.02418   -2.93  0.00346 **
## z.diff.lag6  0.07938    0.02330    3.41  0.00067 ***
## z.diff.lag7  0.18125    0.02305    7.86  6.4e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.058 on 1809 degrees of freedom
## Multiple R-squared:  0.161,  Adjusted R-squared:  0.158
## F-statistic: 43.5 on 8 and 1809 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -2.451 3.015
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

```
test_gas_none = ur.df(l_gas, type = "none", lags = 7, selectlags = "BIC")
summary(test_gas_none)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3471 -0.0213  0.0010  0.0222  0.6290
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1     -7.31e-05   3.49e-04   -0.21  0.83420
## z.diff.lag1  1.68e-01   2.31e-02    7.28  5.1e-13 ***
## z.diff.lag2 -2.84e-01   2.34e-02  -12.16  < 2e-16 ***
## z.diff.lag3  4.31e-02   2.42e-02    1.78  0.07537 .
## z.diff.lag4 -5.46e-02   2.42e-02   -2.26  0.02409 *
## z.diff.lag5 -7.31e-02   2.42e-02   -3.02  0.00254 **
## z.diff.lag6  7.81e-02   2.33e-02    3.35  0.00083 ***
## z.diff.lag7  1.79e-01   2.31e-02    7.77  1.3e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.058 on 1810 degrees of freedom
## Multiple R-squared:  0.159,  Adjusted R-squared:  0.155
## F-statistic: 42.7 on 8 and 1810 DF,  p-value: <2e-16
##
```

```
##
## Value of test-statistic is: -0.2093
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
summary(ur.df(diff(l_gas), type = "drift", lags = 7, selectlags = "BIC"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3477 -0.0219  0.0005  0.0216  0.6290
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.000237   0.001362    0.17   0.8619
## z.lag.1     -0.941925   0.065857  -14.30  < 2e-16 ***
## z.diff.lag1  0.110426   0.060353    1.83   0.0675 .
## z.diff.lag2 -0.172946   0.053426   -3.24   0.0012 **
## z.diff.lag3 -0.130722   0.046601   -2.81   0.0051 **
## z.diff.lag4 -0.184122   0.039308   -4.68  3.0e-06 ***
## z.diff.lag5 -0.257505   0.029632   -8.69  < 2e-16 ***
## z.diff.lag6 -0.179136   0.023071   -7.76  1.4e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.058 on 1809 degrees of freedom
## Multiple R-squared:  0.517,  Adjusted R-squared:  0.515
## F-statistic:  276 on 7 and 1809 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -14.3 102.3
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

Anche il Gas risulta essere I(1) senza trend né drift.

4.2.3 Test ADF su Renew

Applichiamo la stessa procedura al logaritmo delle Rinnovabili.

```
test_renew_trend = ur.df(l_renew, type = "trend", lags = 7, selectlags = "BIC")
summary(test_renew_trend)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3455 -0.0230 -0.0003  0.0251  0.4080
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.35e+00   1.74e-01    7.76  1.4e-14 ***
## z.lag.1     -1.29e-01   1.66e-02   -7.76  1.4e-14 ***
## tt          -3.22e-06   2.83e-06   -1.14     0.26
## z.diff.lag1 -1.16e-01   2.39e-02   -4.86  1.3e-06 ***
## z.diff.lag2 -2.09e-01   2.30e-02   -9.10  < 2e-16 ***
## z.diff.lag3 -1.86e-01   2.18e-02   -8.53  < 2e-16 ***
## z.diff.lag4 -1.87e-01   2.09e-02   -8.95  < 2e-16 ***
## z.diff.lag5 -2.52e-01   2.00e-02  -12.63  < 2e-16 ***
## z.diff.lag6 -1.63e-01   1.93e-02   -8.45  < 2e-16 ***
## z.diff.lag7  5.93e-01   1.87e-02   31.65  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0629 on 1808 degrees of freedom
## Multiple R-squared:  0.776,  Adjusted R-squared:  0.774
## F-statistic:  694 on 9 and 1808 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -7.759 20.09 30.11
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
```

```
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

I test formali confermano l'assunzione di stazionarietà. Il test di radice unitaria porta a rifiutare l'ipotesi nulla: il logaritmo delle Rinnovabili risulta stazionario già nei livelli. Poiché la cointegrazione richiede che le variabili siano integrate dello stesso ordine, la natura I(0) delle rinnovabili ne impone l'esclusione dall'analisi di Johansen.

4.3 Test di Causalità di Granger

Prima di procedere con l'analisi di cointegrazione, esploriamo la presenza di causalità di Granger tra le variabili differenziate.

```
lag_selection <- VARselect(data_diff, lag.max = 30)
tabella_lag <- as.data.frame(t(lag_selection$selection))

knitr::kable(
  tabella_lag,
  caption = "Selezione ottimale dei lag (Modello in differenze)",
  booktabs = TRUE
) %>%
  kable_styling(latex_options = "hold_position")
```

Tabella 1: Selezione ottimale dei lag (Modello in differenze)

AIC(n)	HQ(n)	SC(n)	FPE(n)
30	14	14	30

```
lag_granger <- 14
```

```
grangertest(data_diff[, "d_PUN"], data_diff[, "d_Gas"], order = lag_granger)
```

```
## Granger causality test
##
## Model 1: data_diff[, "d_Gas"] ~ Lags(data_diff[, "d_Gas"], 1:14) + Lags(data_diff[, "d_PUN"]
## Model 2: data_diff[, "d_Gas"] ~ Lags(data_diff[, "d_Gas"], 1:14)
##   Res.Df  Df   F  Pr(>F)
## 1   1782
## 2   1796 -14 5.5 1.9e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Il test verifica se il Gas causa secondo Granger il PUN.

```
grangertest(data_diff[, "d_Gas"], data_diff[, "d_PUN"], order = lag_granger)
```

```
## Granger causality test
##
## Model 1: data_diff[, "d_PUN"] ~ Lags(data_diff[, "d_PUN"], 1:14) + Lags(data_diff[, "d_Gas"]
## Model 2: data_diff[, "d_PUN"] ~ Lags(data_diff[, "d_PUN"], 1:14)
```

```
##   Res.Df  Df    F Pr(>F)
## 1   1782
## 2   1796 -14 26.9 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Il test verifica se il PUN causa secondo Granger il Gas.

I risultati evidenziano una forte interdipendenza statistica tra i due mercati, con p-value estremamente significativi in entrambe le direzioni. Dal Gas al PUN si conferma la causalità attesa: il prezzo del gas agisce come driver fondamentale per il prezzo energetico, anticipando e spiegando i movimenti del PUN. La presenza di una causalità in direzione opposta segnala un possibile effetto di feedback. Sebbene l'intensità sia inferiore rispetto alla direzione primaria, sembra esistere una causalità bidirezionale: se il Gas guida i costi di generazione del PUN, quest'ultimo esercita un effetto di ritorno sul mercato del combustibile. Questa interconnessione rafforza l'ipotesi di un equilibrio di cointegrazione nel lungo periodo.

4.4 Test di Cointegrazione di Johansen

Avendo verificato che entrambe le serie sono I(1), procediamo con il test di Johansen per identificare eventuali relazioni di cointegrazione. Sfruttiamo il BIC per la selezione dei lag ottimali.

```
lag_select_levels = VARselect(data_levels, lag.max = 30, type = "both")
knitr::kable(as.data.frame(t(lag_select_levels$selection)),
             caption = "Selezione ottimale dei lag per Test di Johansen (Livelli)",
             booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position")
```

Tabella 2: Selezione ottimale dei lag per Test di Johansen (Livelli)

AIC(n)	HQ(n)	SC(n)	FPE(n)
29	15	15	29

```
K_johansen = lag_select_levels$selection["SC(n)"]
```

4.4.1 Test della Traccia

```
vecm_trace = ca.jo(data_levels, type = "trace", ecdet = "const",
                   K = K_johansen, spec = "longrun")
summary(vecm_trace)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , without linear trend and constant in cointegration
##
```

```
## Eigenvalues (lambda):
## [1]  1.385e-02  2.786e-03 -1.178e-20
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  5.05  7.52  9.24 12.97
## r = 0  | 30.31 17.85 19.96 24.60
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##           l_PUN.l15 l_Gas.l15 constant
## l_PUN.l15     1.000     1.000   1.0000
## l_Gas.l15    -0.839    -4.584  -0.7104
## constant     -1.677    12.900  -0.8162
##
## Weights W:
## (This is the loading matrix)
##
##          l_PUN.l15 l_Gas.l15   constant
## l_PUN.d -0.1298498 0.0008344 -3.164e-16
## l_Gas.d  0.0008279 0.0013710  4.106e-18
```

Il test della traccia rifiuta l'ipotesi $r = 0$ ma non rifiuta $r \leq 1$, indicando la presenza di una relazione di cointegrazione.

4.4.2 Test del Massimo Autovalore

```
vecm_eigen = ca.jo(data_levels, type = "eigen", ecdet = "const",
                   K = K_johansen, spec = "longrun")
summary(vecm_eigen)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in
##
## Eigenvalues (lambda):
## [1]  1.385e-02  2.786e-03 -1.178e-20
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  5.05  7.52  9.24 12.97
## r = 0  | 25.26 13.75 15.67 20.20
```

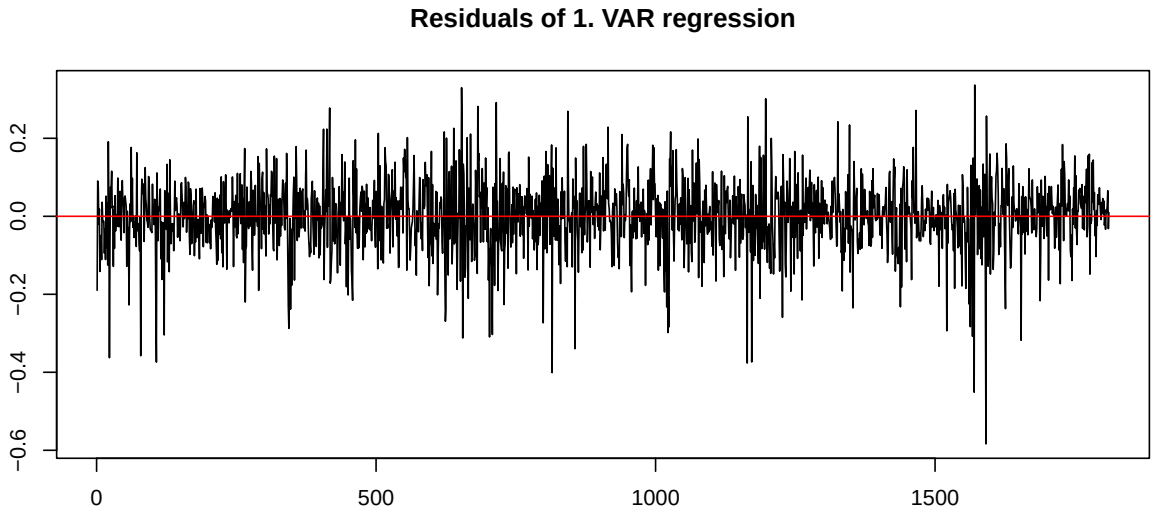
```
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##            l_PUN.l15 l_Gas.l15 constant
## l_PUN.l15      1.000     1.000   1.0000
## l_Gas.l15     -0.839    -4.584  -0.7104
## constant      -1.677    12.900  -0.8162
##
## Weights W:
## (This is the loading matrix)
##
##           l_PUN.l15 l_Gas.l15   constant
## l_PUN.d -0.1298498 0.0008344 -3.164e-16
## l_Gas.d  0.0008279 0.0013710  4.106e-18
```

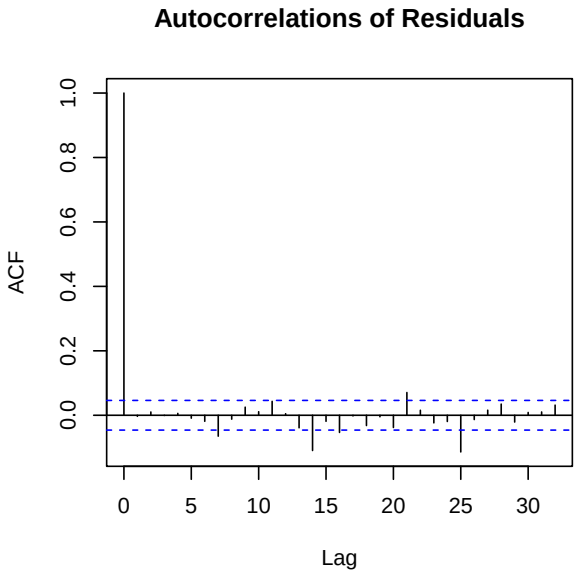
Il test del massimo autovalore conferma la presenza di $r = 1$ relazione di cointegrazione.

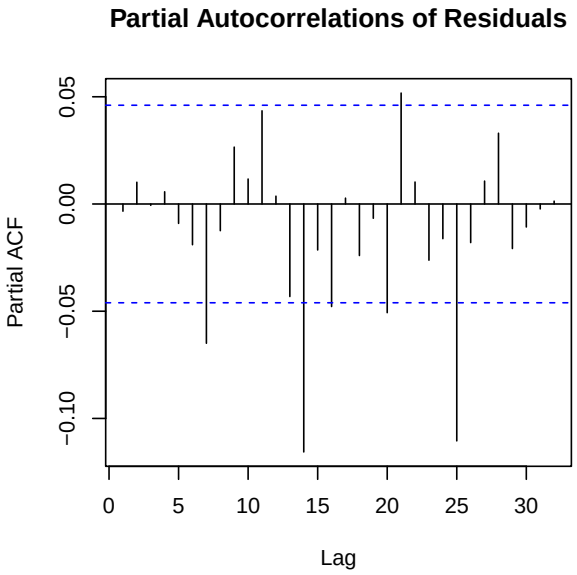
4.5 Diagnostica del VECM

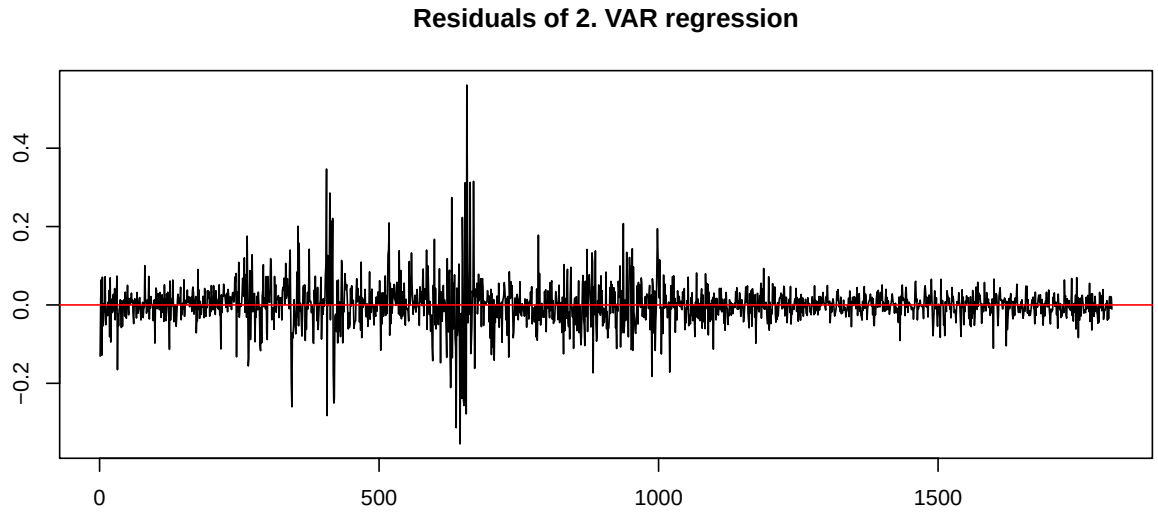
Verifichiamo la qualità dei residui del modello VECM.

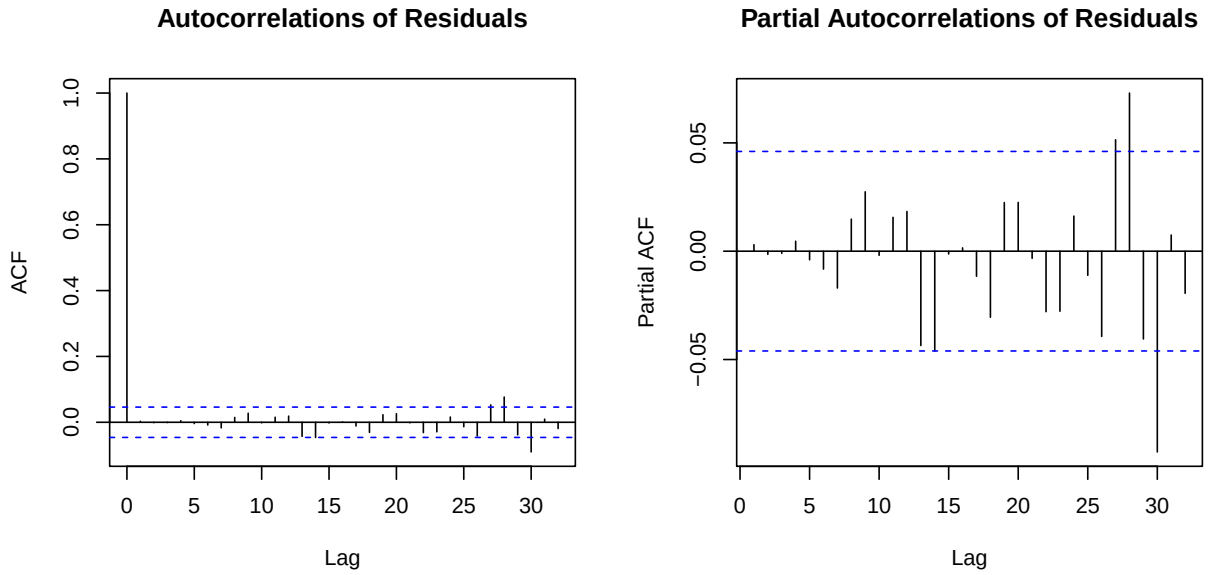
```
par(mfrow = c(2, 2))
plotres(vecm_trace)
```


Residuals of 1. VAR regression
0.2
0.0
−0.2
−0.4
−0.6
0
500
1000
1500


Autocorrelations of Residuals
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
Lag


Partial Autocorrelations of Residuals
Partial ACF
0.05
0.00
−0.05
−0.10
0
5
10
15
20
25
30
Lag






```
par(mfrow = c(1, 1))
```

I residui del modello VECM risultano generalmente soddisfacenti e coerenti con un processo white noise, indicando che il modello ha catturato correttamente la dinamica media del sistema. Tuttavia, l'analisi grafica evidenzia alcuni cluster di volatilità e fenomeni di eteroschedasticità, probabilmente legati ai picchi di prezzo occorsi durante la crisi energetica.

4.6 Stima del VECM e Vettori di Cointegrazione

```
vecm_rls = cajorls(vecm_trace, r = 1)
```

4.6.1 Vettore di Cointegrazione

```
beta_tab <- as.data.frame(vecm_rls$beta)
colnames(beta_tab) <- paste0("Beta_", seq_len(ncol(beta_tab)))

knitr::kable(
  beta_tab,
  caption = "Matrice dei vettori di cointegrazione ($\\beta$)",
  digits = 4,
  booktabs = TRUE
) %>%
  kable_styling(latex_options = "hold_position")
```

Tabella 3: Matrice dei vettori di cointegrazione (β)

	Beta_1
l_PUN.l15	1.000
l_Gas.l15	-0.839
constant	-1.677

La relazione di lungo periodo stimata è:

$$\log(\text{PUN}_t) = 0.839 \cdot \log(\text{Gas}_t) + 1.677$$

Il coefficiente 0.839 indica una trasmissione quasi totale dei costi: un incremento dell'1% nel prezzo del gas si traduce in un aumento dello 0.84% circa del PUN, confermando il ruolo del gas come driver principale. La costante (1.677) rappresenta la componente legata ai costi fissi, agli oneri di sistema e alla generazione da altre fonti non legate al gas. Il modello riflette la struttura del mercato energetico italiano, in cui il gas rappresenta la fonte di produzione più flessibile e costosa, le cui variazioni di prezzo tendono a guidare l'intero mercato elettrico.

4.6.2 Coefficienti di Aggiustamento (Alpha)

```
alpha_matrix = vecm_rls$rlm$coefficients[grep("ect", rownames(vecm_rls$rlm$coefficients)), ]
knitr::kable(alpha_matrix, caption = "Velocità di aggiustamento (Alpha)") %>%
  kable_styling(latex_options = "hold_position")
```

Tabella 4: Velocità di aggiustamento (Alpha)

	x
l_PUN.d	-0.1298
l_Gas.d	0.0008

```
alpha_summary <- summary(vecm_rls$rlm)

alpha_pun_pval <- alpha_summary$`Response l_PUN.d`$coefficients["ect1", "Pr(>|t|)"]
```

```r
alpha_gas_pval <- alpha_summary$`Response l_Gas.d`$coefficients["ect1", "Pr(>|t|)"]

cat("Alpha PUN: p-value =", format(alpha_pun_pval, scientific = TRUE), "\n")
```

```
## Alpha PUN: p-value = 3.621e-06
```

```r
cat("Alpha Gas: p-value =", format(alpha_gas_pval, scientific = TRUE), "\n")
```

```
## Alpha Gas: p-value = 9.612e-01
```

I coefficienti alpha indicano la velocità di aggiustamento verso l'equilibrio di lungo periodo. α_{PUN} risulta negativo (-0.130) e altamente significativo, confermando che il PUN reagisce agli squilibri correggendo circa il 13% dello scostamento per periodo. Al contrario, α_{Gas} è praticamente nullo e statisticamente non significativo, indicando che il Gas non reagisce agli squilibri del sistema e agisce come variabile debolmente esogena che guida il meccanismo di formazione dei prezzi.

4.7 Procedura di Engle-Granger

Per garantire la robustezza dei risultati ottenuti con la procedura di Johansen, procediamo a una verifica utilizzando gli stimatori Engle-Granger e DOLS (Dynamic Ordinary Least Squares). Questo confronto permette di controllare la coerenza delle stime attraverso approcci metodologici differenti.

4.7.1 Passo 1: Regressione OLS Statica

La procedura di Engle-Granger inizia con una regressione OLS delle variabili in livelli per stimare la relazione di lungo periodo.

```r
ols_pun = lm(l_PUN ~ l_Gas, data = data_levels)
summary(ols_pun)
```

```
##
## Call:
## lm(formula = l_PUN ~ l_Gas, data = data_levels)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8672 -0.0635  0.0088  0.0677  0.4083
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.65323    0.01901      87   <2e-16 ***
## l_Gas        0.84416    0.00488     173   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.121 on 1824 degrees of freedom
## Multiple R-squared:  0.943,  Adjusted R-squared:  0.943
## F-statistic: 3e+04 on 1 and 1824 DF,  p-value: <2e-16
```

Il coefficiente stimato è coerente con quello della procedura di Johansen.

4.7.2 Passo 2: Test di Stazionarietà sui Residui

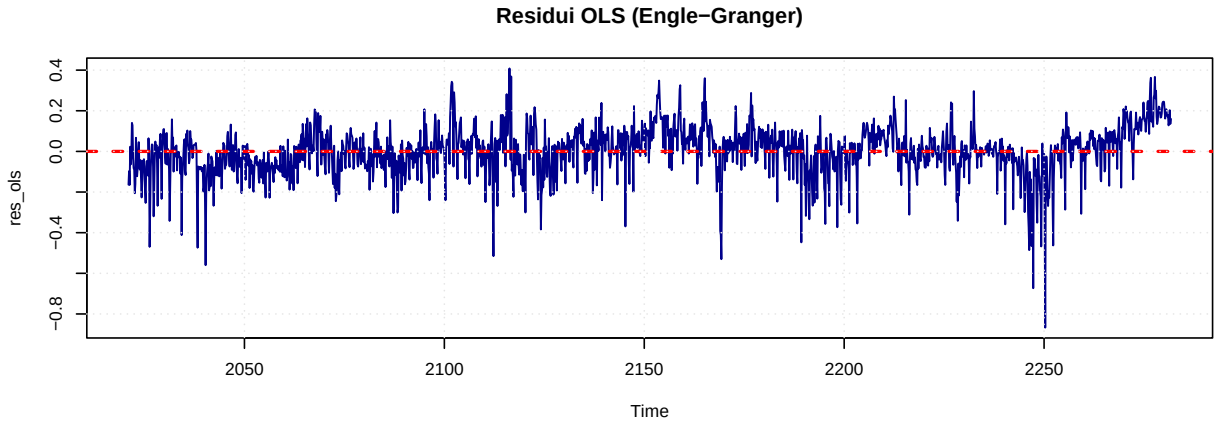
Per verificare la cointegrazione, testiamo la stazionarietà dei residui della regressione OLS.

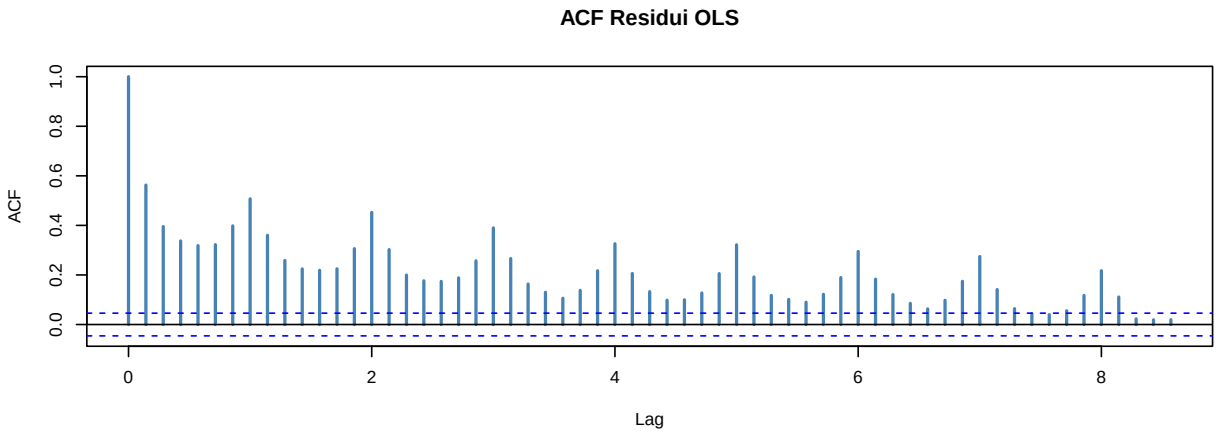
```
res_ols = ts(residuals(ols_pun), start = start(data_levels),
             frequency = frequency(data_levels))
test_res_ols = ur.df(res_ols, type = "none", lags = 7, selectlags = "BIC")
summary(test_res_ols)
```

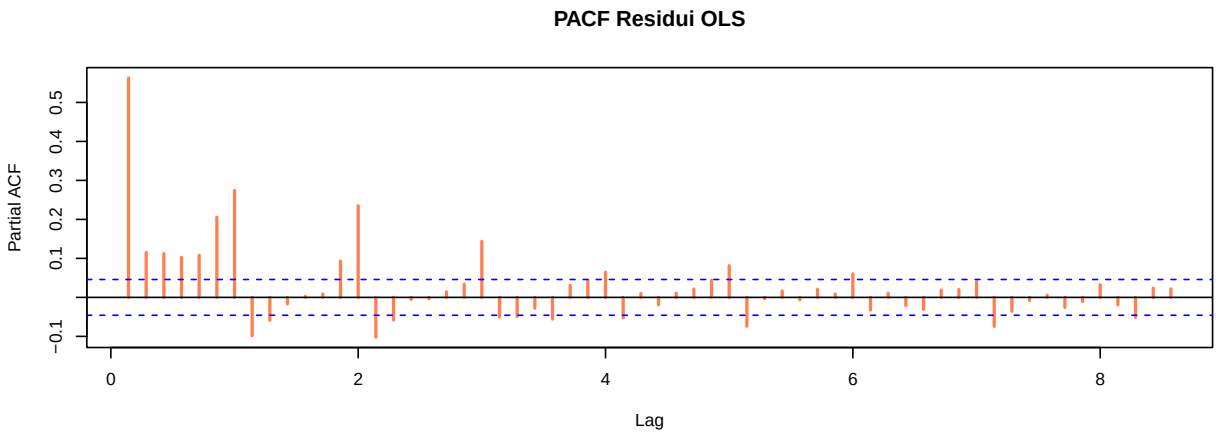
```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5930 -0.0443  0.0076  0.0539  0.3515
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1      -0.1690     0.0255   -6.63  4.3e-11 ***
## z.diff.lag1  -0.4227     0.0320  -13.19  < 2e-16 ***
## z.diff.lag2  -0.3830     0.0322  -11.88  < 2e-16 ***
## z.diff.lag3  -0.3478     0.0315  -11.05  < 2e-16 ***
## z.diff.lag4  -0.3151     0.0304  -10.35  < 2e-16 ***
## z.diff.lag5  -0.3070     0.0288  -10.66  < 2e-16 ***
## z.diff.lag6  -0.2169     0.0267   -8.12  8.6e-16 ***
## z.diff.lag7   0.0969     0.0234    4.15  3.5e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0911 on 1810 degrees of freedom
## Multiple R-squared:  0.349,  Adjusted R-squared:  0.346
## F-statistic:  121 on 8 and 1810 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -6.633
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

Il test tau1 rifiuta l'ipotesi di radice unitaria nei residui, confermando che le serie sono cointegrate.

```r
par(mfrow = c(3, 1))
plot.ts(res_ols, main = "Residui OLS (Engle-Granger)", col = "darkblue", lwd = 1.2)
abline(h = 0, col = "red", lty = 2, lwd = 2)
grid(col = "gray90")
acf(res_ols, lag.max = 60, main = "ACF Residui OLS", col = "steelblue", lwd = 2)
pacf(res_ols, lag.max = 60, main = "PACF Residui OLS", col = "coral", lwd = 2)
```







```r
par(mfrow = c(1, 1))
```

I residui confermano la stazionarietà della relazione oscillando attorno allo zero, ma i grafici evidenziano un'autocorrelazione persistente. Tale dinamica conferma che la sola regressione statica è insufficiente, giustificando l'impiego del modello ECM per catturare i corretti aggiustamenti di breve periodo.

4.7.3 Error Correction Model (ECM)

Stimiamo un modello di correzione dell'errore utilizzando i residui della regressione OLS come termine di errore di cointegrazione.

```r
data_ecm = cbind(data_levels, ect_ols = res_ols)
ecm_ols = dynlm(d(data_levels.l_PUN) ~ d(data_levels.l_Gas) + L(ect_ols),
                data = data_ecm)
summary(ecm_ols)
```

```
##
## Time series regression with "ts" data:
## Start = 2021(2), End = 2281(6)
##
## Call:
## dynlm(formula = d(data_levels.l_PUN) ~ d(data_levels.l_Gas) +
##     L(ect_ols), data = data_ecm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6910 -0.0511  0.0049  0.0597  0.3846
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           0.000146   0.002334    0.06     0.95
## d(data_levels.l_Gas)  0.823961   0.037032   22.25   <2e-16 ***
## L(ect_ols)           -0.437398   0.019391  -22.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0997 on 1822 degrees of freedom
## Multiple R-squared:  0.37,   Adjusted R-squared:  0.369
## F-statistic:  534 on 2 and 1822 DF,  p-value: <2e-16
```

Il coefficiente del termine ECT è negativo e significativo (-0.437), indicando che il sistema corregge circa il 44% dello squilibrio per periodo. Il coefficiente di $\Delta \log(Gas)$ cattura le relazioni di breve periodo, confermando una reattività immediata e quasi totale del PUN alle variazioni del prezzo del gas.

4.8 Dynamic OLS (DOLS)

4.8.1 Stima DOLS

Il metodo Dynamic OLS corregge l'endogeneità dei regressori includendo lead e lag della variabile esplicativa differenziata. In base alla lunghezza della serie, il numero ottimale di lead/lag considerati è **5**.

```
popt = ceiling((1/3)*(length(data_levels[, 1]))^(1/3))
dols_pun = dynlm(l_PUN ~ l_Gas + L(d(l_Gas), -popt:popt), data = data_levels)
summary(dols_pun)
```

```
##
## Time series regression with "ts" data:
## Start = 2021(7), End = 2281(1)
##
## Call:
## dynlm(formula = l_PUN ~ l_Gas + L(d(l_Gas), -popt:popt), data = data_levels)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8631 -0.0606  0.0073  0.0665  0.4494
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.65850    0.01925   86.16   <2e-16 ***
## l_Gas                         0.84278    0.00493  170.79   <2e-16 ***
## L(d(l_Gas), -popt:popt)-5     0.06915    0.04890    1.41   0.1575
## L(d(l_Gas), -popt:popt)-4    -0.03667    0.04943   -0.74   0.4582
## L(d(l_Gas), -popt:popt)-3    -0.06471    0.05114   -1.27   0.2060
## L(d(l_Gas), -popt:popt)-2    -0.06889    0.05104   -1.35   0.1773
## L(d(l_Gas), -popt:popt)-1    -0.05711    0.05060   -1.13   0.2593
## L(d(l_Gas), -popt:popt)0     -0.13929    0.05044   -2.76   0.0058 **
## L(d(l_Gas), -popt:popt)1      0.16794    0.05051    3.33   0.0009 ***
## L(d(l_Gas), -popt:popt)2     -0.05853    0.05089   -1.15   0.2502
## L(d(l_Gas), -popt:popt)3     -0.01673    0.05097   -0.33   0.7427
## L(d(l_Gas), -popt:popt)4      0.00711    0.04927    0.14   0.8853
## L(d(l_Gas), -popt:popt)5     -0.09196    0.04867   -1.89   0.0590 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.12 on 1802 degrees of freedom
## Multiple R-squared:  0.944,  Adjusted R-squared:  0.943
## F-statistic: 2.51e+03 on 12 and 1802 DF,  p-value: <2e-16
```

Il modello DOLS conferma la robustezza della relazione di lungo periodo attraverso un'elasticità stimata dello 0.843, valore quasi identico allo 0.839 ottenuto con la procedura di Johansen. L'elevata significatività statistica del parametro e un R-squared del 94.4% indicano un legame estremamente solido, validando definitivamente la coerenza del vettore di cointegrazione individuato.

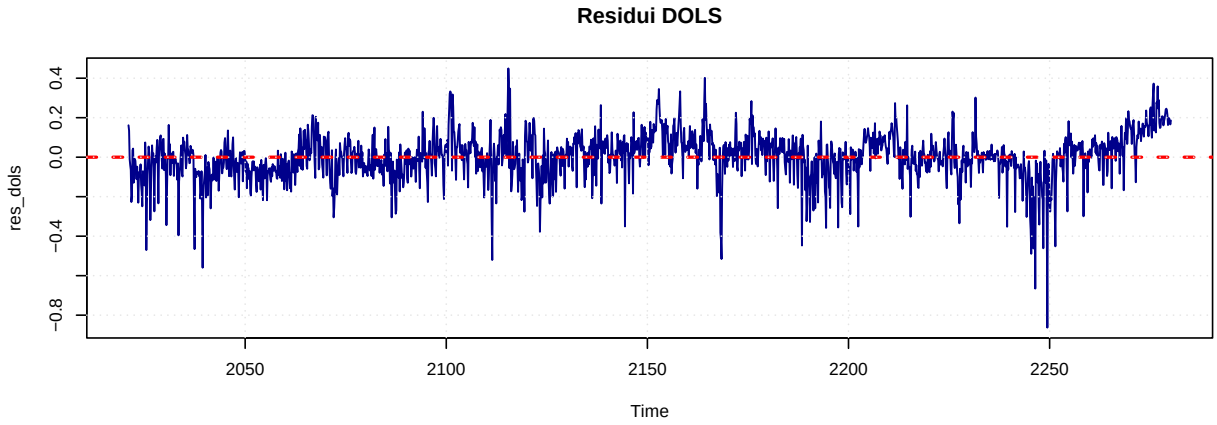
4.8.2 Test sui Residui DOLS

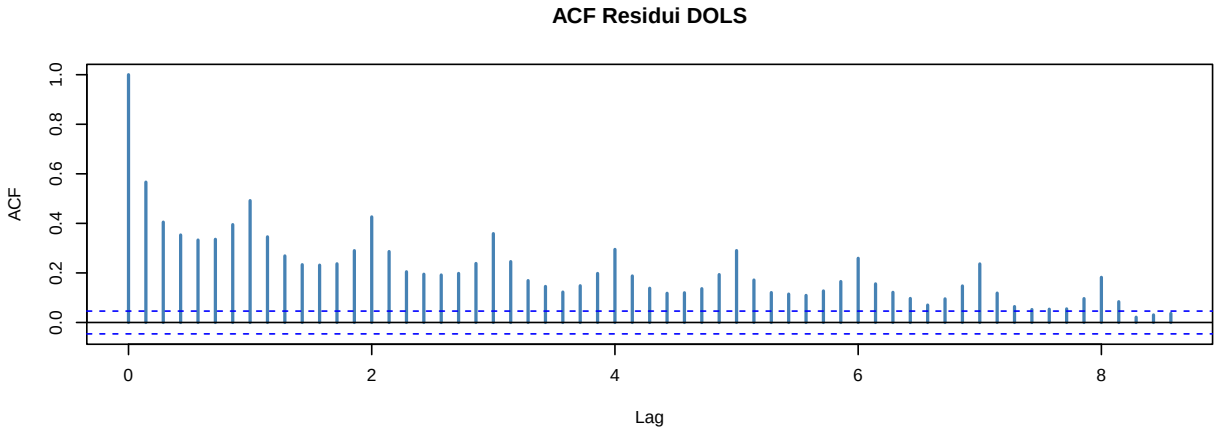
```
res_dols = ts(residuals(dols_pun), start = start(data_levels),
              frequency = frequency(data_levels))
test_res_dols = ur.df(res_dols, lags = 7, type = "none", selectlags = "BIC")
summary(test_res_dols)
```

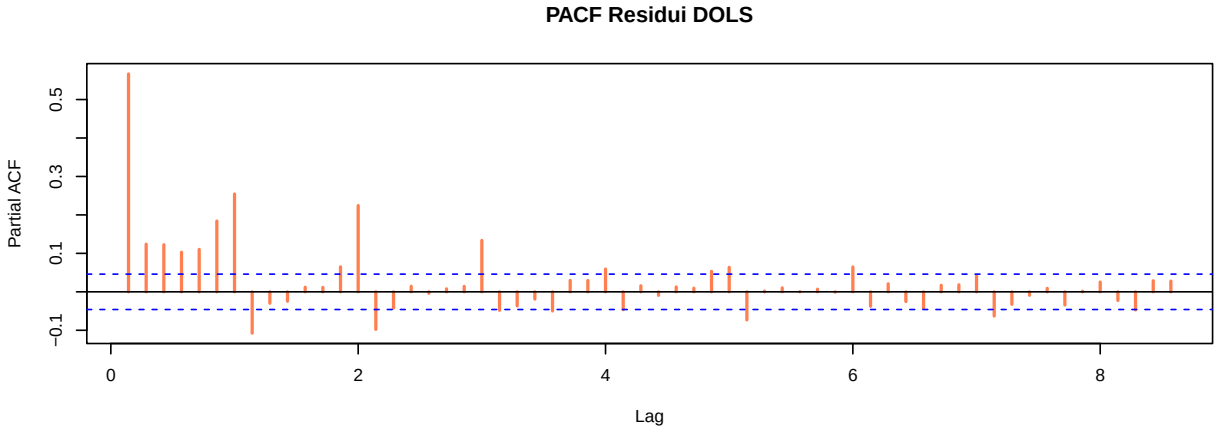
```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6035 -0.0442  0.0052  0.0532  0.3323
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1      -0.1732     0.0255   -6.80  1.4e-11 ***
## z.diff.lag1  -0.4138     0.0320  -12.92  < 2e-16 ***
## z.diff.lag2  -0.3722     0.0323  -11.52  < 2e-16 ***
## z.diff.lag3  -0.3271     0.0317  -10.31  < 2e-16 ***
## z.diff.lag4  -0.2934     0.0307   -9.56  < 2e-16 ***
## z.diff.lag5  -0.2693     0.0291   -9.25  < 2e-16 ***
## z.diff.lag6  -0.1926     0.0268   -7.18  1.0e-12 ***
## z.diff.lag7   0.1043     0.0234    4.46  8.9e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0904 on 1799 degrees of freedom
## Multiple R-squared:  0.339,  Adjusted R-squared:  0.336
## F-statistic:  115 on 8 and 1799 DF,  p-value: <2e-16
##
##
## Value of test-statistic is: -6.804
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

La stazionarietà dei residui conferma che il legame individuato non è una regressione spuria ma un equilibrio di lungo periodo.

```r
par(mfrow = c(3, 1))
plot.ts(res_dols, main = "Residui DOLS", col = "darkblue", lwd = 1.2)
abline(h = 0, col = "red", lty = 2, lwd = 2)
grid(col = "gray90")
acf(res_dols, lag.max = 60, main = "ACF Residui DOLS", col = "steelblue", lwd = 2)
pacf(res_dols, lag.max = 60, main = "PACF Residui DOLS", col = "coral", lwd = 2)
```







```r
par(mfrow = c(1, 1))
```

Tuttavia, come nel caso dell'ECM basato su Engle-Granger, i residui mostrano autocorrelazione persistente e non possono essere considerati white noise.

4.8.3 ECM con Residui DOLS

```r
data_ecm_dols = cbind(data_levels, ect_dols = res_dols)
ecm_dols = dynlm(d(data_levels.l_PUN) ~ d(data_levels.l_Gas) + L(ect_dols),
                 data = data_ecm_dols)
summary(ecm_dols)
```

```
##
## Time series regression with "ts" data:
## Start = 2021(2), End = 2280(3)
##
## Call:
## dynlm(formula = d(data_levels.l_PUN) ~ d(data_levels.l_Gas) +
##     L(ect_dols), data = data_ecm_dols)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5730 -0.0596 -0.0038  0.0550  0.7472
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.00017    0.00265    0.06   0.9489
## d(data_levels.l_Gas)   0.87466    0.04181   20.92   <2e-16 ***
## L(ect_dols)           -0.07105    0.02218   -3.20   0.0014 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.113 on 1812 degrees of freedom
##   (0 observations deleted due to missingness)
## Multiple R-squared:  0.198,  Adjusted R-squared:  0.197
## F-statistic:  224 on 2 and 1812 DF,  p-value: <2e-16
```

L'ECM basato su DOLS mostra che il PUN reagisce con estrema rapidità alle variazioni del gas nel brevissimo periodo (0.875). Tuttavia, la velocità di correzione dello squilibrio è più contenuta (7.1%), indicando che il sistema impiega più tempo per tornare stabilmente verso la sua relazione di equilibrio di lungo periodo.

Il confronto tra i tre modelli conferma la solidità della relazione di lungo periodo, con un'elasticità del PUN rispetto al Gas costantemente compresa tra 0.82 e 0.87. Emergono tuttavia differenze nette nella dinamica di aggiustamento: il modello Engle-Granger tende a sovrastimare la velocità di correzione (44%), mentre DOLS e VECM suggeriscono un rientro all'equilibrio più lento (rispettivamente 7% e 13%). Il DOLS, in quanto stimatore più efficiente in presenza di endogeneità, fornisce stime più affidabili della relazione di lungo periodo, convergendo quasi perfettamente con il VECM. Quest'ultimo si distingue come l'approccio statisticamente più robusto: i suoi residui mostrano

un comportamento superiore, avvicinandosi alle proprietà di un processo white noise e superando nettamente le autocorrelazioni persistenti presenti negli ECM basati su Engle-Granger e DOLS. Si nota infine come la direzione della relazione modellata dal VECM sia unidirezionale da Gas a PUN, in contrasto con quanto emerso dai test sulla cusalità secondo Granger.

5 Transfer Function Models

L'analisi di cointegrazione ha evidenziato come il prezzo del gas sia debolmente esogeno rispetto al PUN, suggerendo una relazione dinamica prevalentemente unidirezionale. Questo risultato motiva l'adozione di modelli a funzione di trasferimento (TFM), che consentono di isolare e quantificare l'impatto dinamico di una variabile esplicativa su una variabile risposta attraverso una struttura parsimoniosa.

Poiché l'analisi precedente si è concentrata sulla relazione Gas-PUN, e non è emersa alcuna relazione di lungo periodo tra PUN e Rinnovabili, si ritiene opportuno stimare un ulteriore TFM anche per questa coppia di variabili. L'obiettivo è indagare se, pur in assenza di cointegrazione, esistano dinamiche di breve periodo significative attraverso cui le Rinnovabili influenzino il prezzo dell'energia elettrica.

5.1 Pre-Whitening e Cross-Correlazione

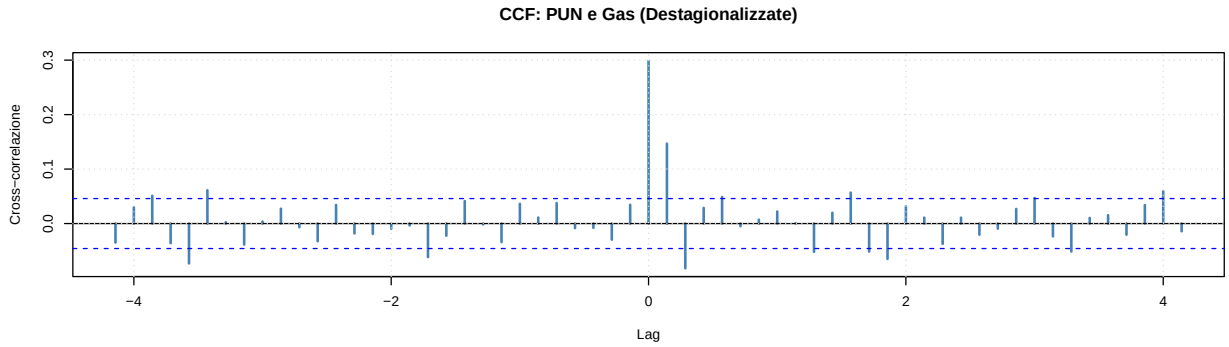
Prima di procedere con la stima dei modelli di funzione di trasferimento, è necessario destagionalizzare le serie per eliminare le componenti deterministiche che potrebbero distorcere l'identificazione della struttura dinamica. Utilizziamo i modelli SARIMA stimati in precedenza per ottenere serie pre-whitened, dalle quali la stagionalità settimanale è stata rimossa, pur tenendo a mente i limiti precedentemente discussi.

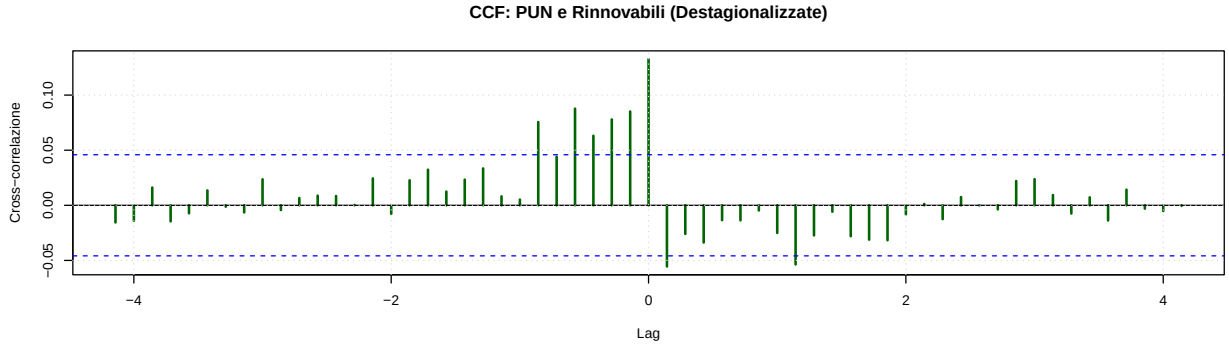
```
mod_pun_ds = Arima(l_pun, order = c(0, 1, 0),
                   seasonal = list(order = c(1, 0, 1), period = 7))
l_pun_ds = mod_pun_ds$residuals

mod_gas_ds = Arima(l_gas, order = c(0, 1, 0),
                   seasonal = list(order = c(0, 1, 1), period = 7))
l_gas_ds = mod_gas_ds$residuals

mod_renew_ds = Arima(l_renew, order = c(0, 0, 0),
                     seasonal = list(order = c(1, 1, 1), period = 7))
l_renew_ds = mod_renew_ds$residuals
```

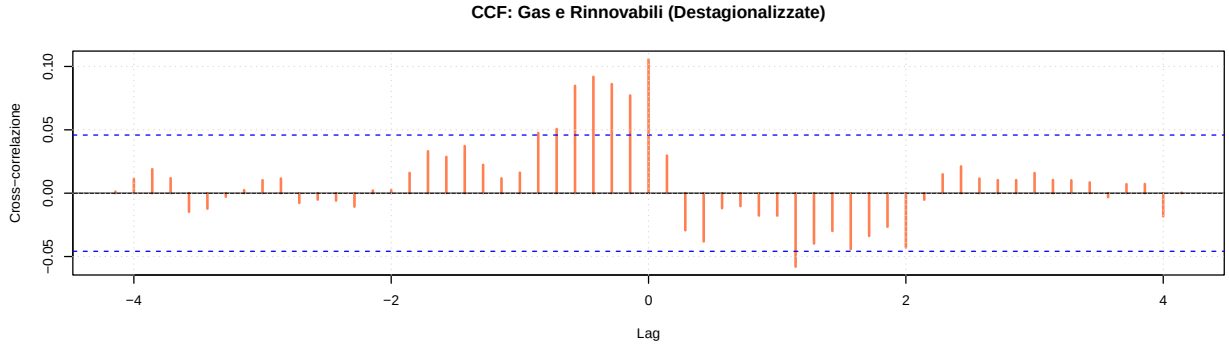
L'analisi delle funzioni di cross-correlazione (CCF) sulle serie destagionalizzate permette di identificare la presenza e la struttura temporale delle relazioni dinamiche tra le variabili.

```
par(mfrow = c(3, 1))
ccf(l_pun_ds, l_gas_ds, main = "CCF: PUN e Gas (Destagionalizzate)",
    col = "steelblue", lwd = 2, ylab = "Cross-correlazione")
grid(col = "gray85", lty = "dotted")

ccf(l_pun_ds, l_renew_ds, main = "CCF: PUN e Rinnovabili (Destagionalizzate)",
    col = "darkgreen", lwd = 2, ylab = "Cross-correlazione")
grid(col = "gray85", lty = "dotted")
```

```r
ccf(l_gas_ds, l_renew_ds, main = "CCF: Gas e Rinnovabili (Destagionalizzate)",
    col = "coral", lwd = 2, ylab = "Cross-correlazione")
grid(col = "gray85", lty = "dotted")
```








```r
par(mfrow = c(1, 1))
```

La cross-correlazione tra PUN e Gas mostra un picco netto al lag zero, confermando l'impatto immediato e dominante del costo del gas sul prezzo elettrico. La relazione PUN-Rinnovabili presenta invece una struttura più articolata, con correlazioni significative distribuite su più lag, indicando una possibile risposta ritardata del sistema. Si nota inoltre la somiglianza tra le CCF di PUN-Rinnovabili e Gas-Rinnovabili; in questa relazione l'obiettivo principale è la modellazione di PUN, pertanto si ignorerà questa ultima coppia di variabili.

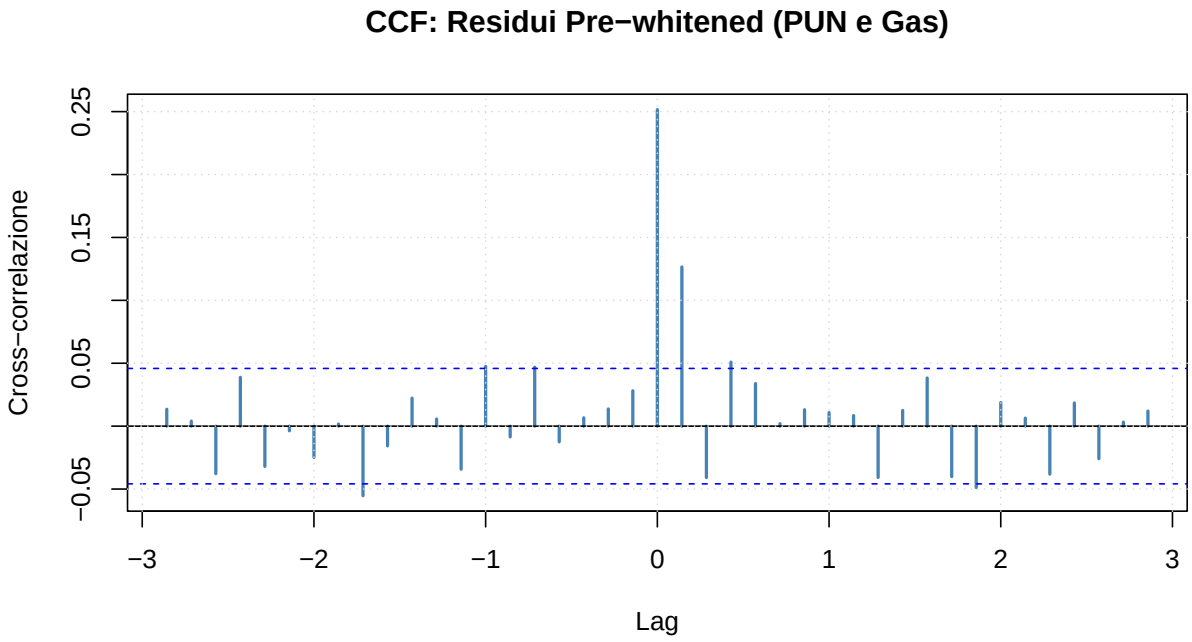
5.2 Transfer Function Model: Gas → PUN

5.2.1 Identificazione della Struttura

Il primo passo consiste nell'identificare la struttura ARMA della serie input (Gas) e nel calcolare i residui pre-whitened per entrambe le serie. Dopo destagionalizzazione e differenziazione, il modello del Gas si riduce a un processo ARMA(0,2).

```
mod_gas_arma = Arima(l_gas_ds, order = c(0, 0, 2))
alfa_gas = mod_gas_arma$residuals
mod_pun_w = Arima(l_pun_ds, order = c(0, 0, 2),
                  fixed = c(mod_gas_arma$coef[1], mod_gas_arma$coef[2],
                            mod_gas_arma$coef[3]))
beta_gas = mod_pun_w$residuals
```

```
ccf(beta_gas, alfa_gas, lag.max = 20,
    main = "CCF: Residui Pre-whitened (PUN e Gas)",
    col = "steelblue", lwd = 2, ylab = "Cross-correlazione")
grid(col = "gray85", lty = "dotted")
```



L'analisi della cross-correlazione dei residui pre-whitened permette di identificare i parametri della funzione di trasferimento secondo la notazione standard: b (ritardo iniziale), r (ordine del denominatore), s (ordine del numeratore). Dal grafico si identificano $b = 0$, $r = 0$, $s = 1$, indicando una risposta immediata del PUN con persistenza limitata a un periodo.

5.2.2 Stima del Modello

```
tfm_gas = MTS::tfm1(l_pun_ds, l_gas_ds, orderN = c(0, 0, 2),
                    orderX = c(0, 1, 0))
```

```
## ARMA coefficients & s.e.:
##                ma1     ma2
## coef.arma -0.5710 -0.1849
## se.arma    0.0224  0.0236
## Transfer function coefficients & s.e.:
##      intercept     X
## v    -0.000539 0.544 0.2400
## se.v  0.000473 0.034 0.0341
```

```
coef_gas = tfm_gas$estimate
se_gas = sqrt(diag(tfm_gas$varcoef))
t_stat_gas = coef_gas / se_gas
p_value_gas = 2 * (1 - pnorm(abs(t_stat_gas)))

results_gas = data.frame(
  Parametro = names(coef_gas),
  Stima = coef_gas,
  Std_Error = se_gas,
  t_stat = t_stat_gas,
  p_value = p_value_gas,
  row.names = NULL
)
knitr::kable(results_gas, digits = 4,
             caption = "Coefficienti stimati - TFM Gas $\\rightarrow$ PUN",
             booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position")
```

Tabella 5: Coefficienti stimati - TFM Gas → PUN

Parametro	Stima	Std_Error	t_stat	p_value
ma1	-0.5710	0.0224	-25.500	0.0000
ma2	-0.1849	0.0236	-7.846	0.0000
intercept	-0.0005	0.0005	-1.140	0.2544
X	0.5445	0.0340	16.027	0.0000
	0.2400	0.0341	7.043	0.0000

Il modello evidenzia una relazione dinamica altamente significativa tra il prezzo del Gas e il PUN. Un incremento dell'1% nel prezzo del gas si traduce in un aumento contemporaneo del 0.54% del PUN, seguito da un effetto aggiuntivo di 0.24% nel periodo successivo. La componente stocastica del PUN è descritta da un processo MA(2), coerente con la persistenza di breve periodo già osservata nei modelli univariati.

5.2.3 Diagnostica dei Residui

```
par(mfrow = c(3, 1))
ccf(tfm_gas$residuals, alfa_gas[2:length(alfa_gas)], lag.max = 20,
    main = "CCF: Residui TFM e Input Pre-whitened",
```

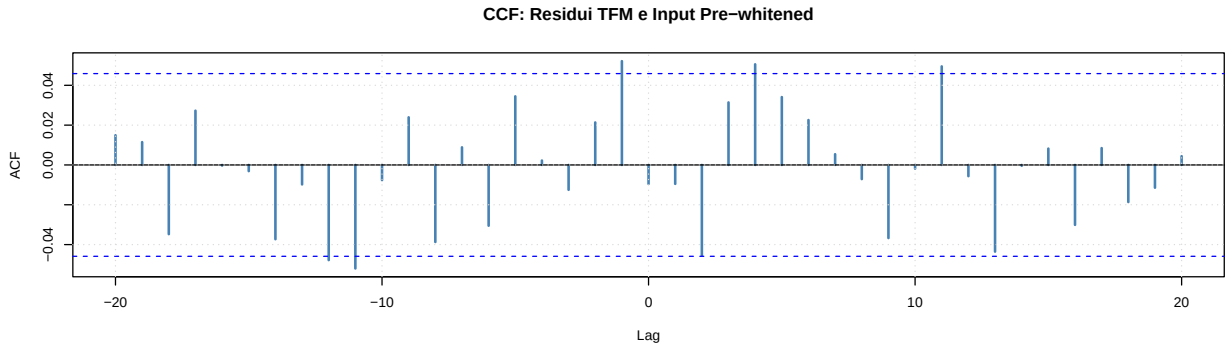
```
    col = "steelblue", lwd = 2)
grid(col = "gray85", lty = "dotted")

acf(tfm_gas$residuals, lag.max = 35, main = "ACF Residui TFM Gas$\rightarrow$PUN",
    col = "steelblue", lwd = 2)
```

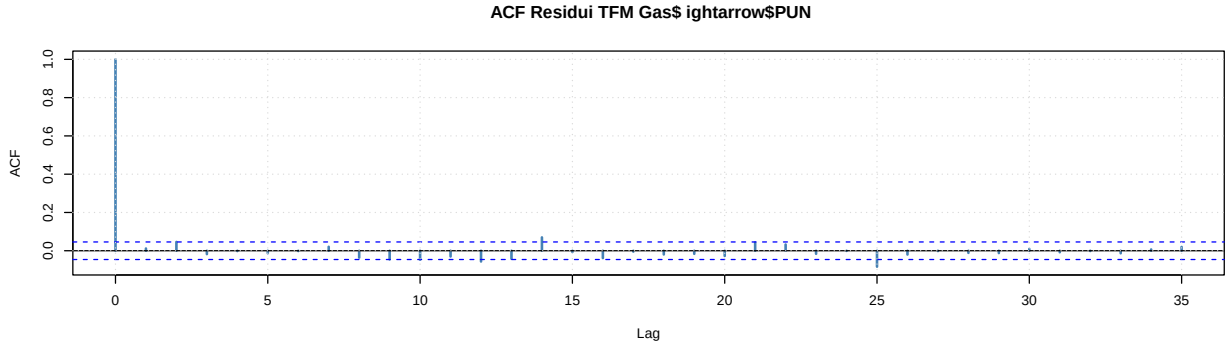
```
## Warning in title(main %||% if (i == j) snames[i] else paste(sn.abbr[i], : font
## width unknown for character 0x0d in encoding latin1
```

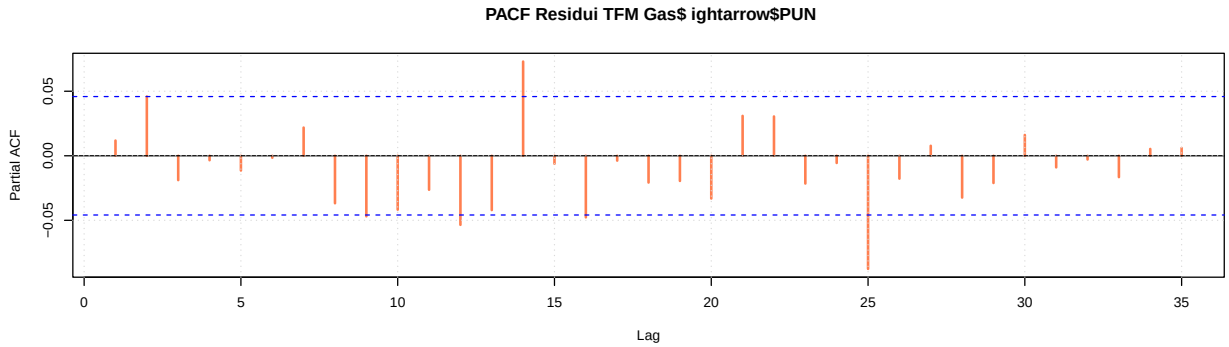
```
grid(col = "gray85", lty = "dotted")

pacf(tfm_gas$residuals, lag.max = 35, main = "PACF Residui TFM Gas$\rightarrow$PUN",
     col = "coral", lwd = 2)
```

```
## Warning in title(main %||% if (i == j) snames[i] else paste(sn.abbr[i], : font
## width unknown for character 0x0d in encoding latin1
```

```
grid(col = "gray85", lty = "dotted")
```



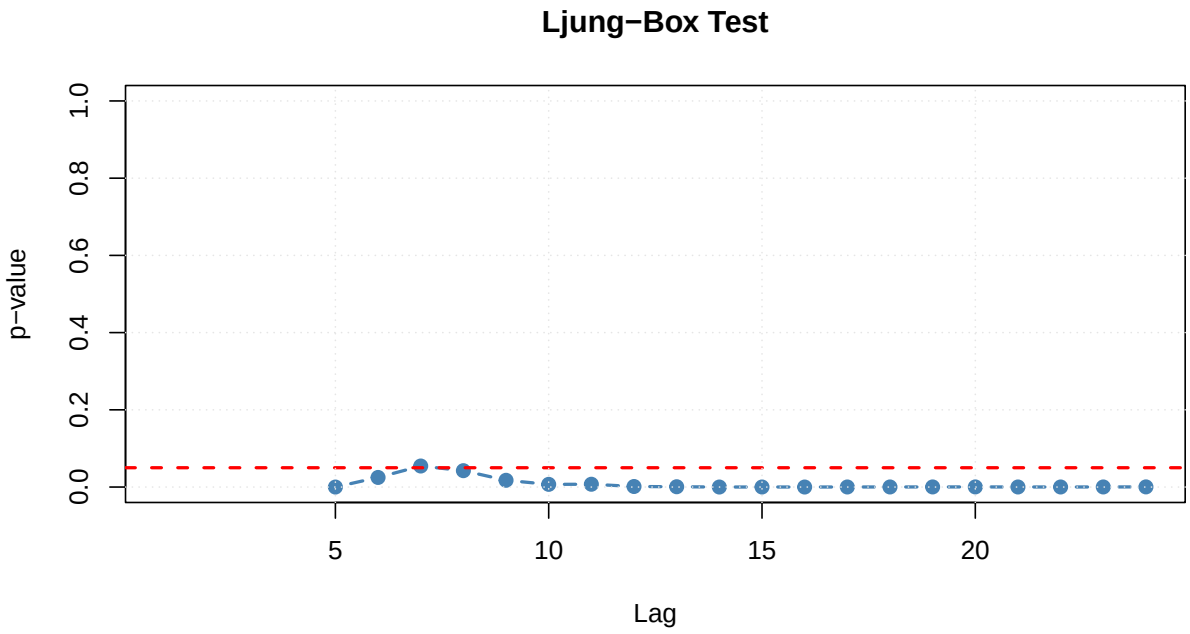





```
par(mfrow = c(1, 1))
```

```
ljung_box_plot(tfm_gas$residuals, max_lag = 24, fitdf = length(coef_gas))
```

```
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
```



I residui del modello mostrano proprietà generalmente coerenti con un processo White Noise. La CCF tra residui e input pre-whitened non presenta autocorrelazioni significative, confermando che la struttura dinamica è stata catturata adeguatamente. L'ACF e la PACF dei residui mostrano valori contenuti entro le bande di confidenza, con eventuali superamenti di ampiezza trascurabile. Nonostante ciò, il test di Ljung-Box segnala la presenza di autocorrelazione residua significativa.

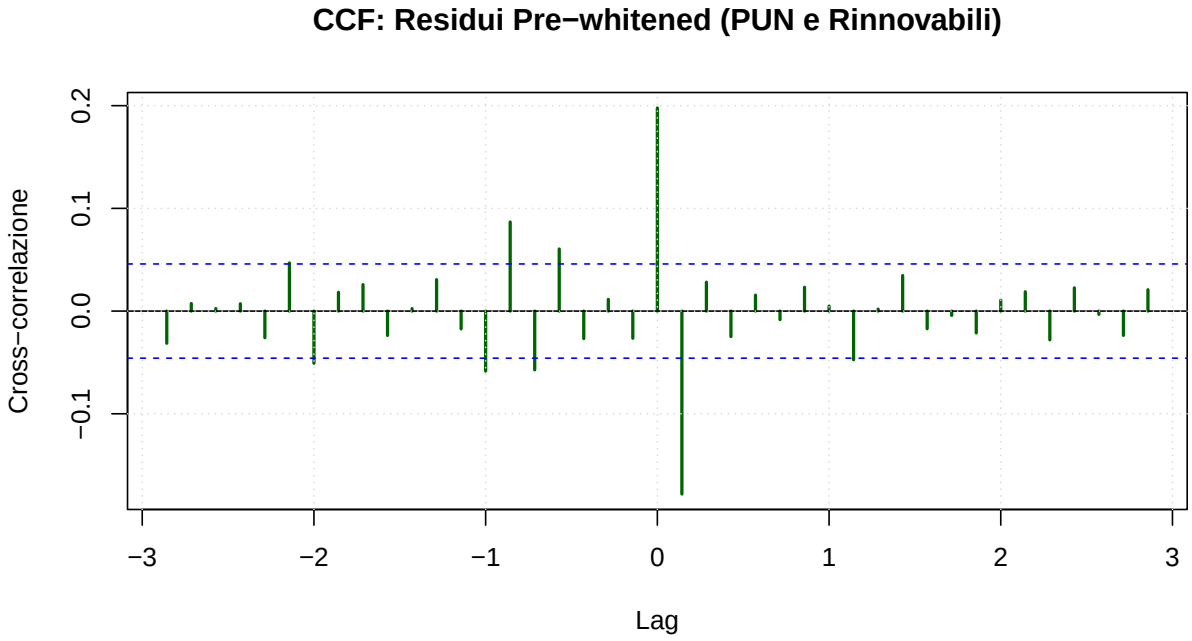
5.3 Transfer Function Model: Rinnovabili → PUN

5.3.1 Identificazione della Struttura

Analogamente, procediamo con l'identificazione del modello per l'impatto delle rinnovabili sul PUN. Il modello delle rinnovabili destagionalizzato si riduce a un processo AR(1).

```
mod_renew_arma = Arima(l_renew_ds, order = c(1, 0, 0))
alfa_renew = mod_renew_arma$residuals
mod_pun_w2 = Arima(l_pun_ds, order = c(1, 0, 0),
                   fixed = mod_renew_arma$coef)
beta_renew = mod_pun_w2$residuals
```

```
ccf(beta_renew, alfa_renew, lag.max = 20,
    main = "CCF: Residui Pre-whitened (PUN e Rinnovabili)",
    col = "darkgreen", lwd = 2, ylab = "Cross-correlazione")
grid(col = "gray85", lty = "dotted")
```



La struttura identificata rimane $b = 0$, $r = 0$, $s = 1$, coerente con il modello precedente.

5.3.2 Stima del Modello

```
tfm_renew = MTS::tfm1(l_pun_ds, l_renew_ds, orderN = c(0, 0, 2),
                      orderX = c(0, 1, 0))
```

```
## ARMA coefficients & s.e.:
##                ma1    ma2
## coef.arma -0.3676 -0.117
## se.arma    0.0233  0.023
## Transfer function coefficients & s.e.:
##      intercept      X
## v    -0.000681 0.4804 -0.4260
## se.v  0.001089 0.0366  0.0366
```

```
coef_renew = tfm_renew$estimate
se_renew = sqrt(diag(tfm_renew$varcoef))
t_stat_renew = coef_renew / se_renew
p_value_renew = 2 * (1 - pnorm(abs(t_stat_renew)))

results_renew = data.frame(
  Parametro = names(coef_renew),
  Stima = coef_renew,
  Std_Error = se_renew,
  t_stat = t_stat_renew,
  p_value = p_value_renew,
  row.names = NULL
```

```
)
knitr::kable(results_renew, digits = 4,
             caption = "Coefficienti stimati - TFM Rinnovabili $\\rightarrow$ PUN",
             booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position")
```

Tabella 6: Coefficienti stimati - TFM Rinnovabili → PUN

Parametro	Stima	Std_Error	t_stat	p_value
ma1	-0.3676	0.0233	-15.7943	0.0000
ma2	-0.1173	0.0230	-5.1015	0.0000
intercept	-0.0007	0.0011	-0.6254	0.5317
X	0.4804	0.0366	13.1218	0.0000
	-0.4260	0.0366	-11.6388	0.0000

Il modello evidenzia una relazione dinamica statisticamente significativa, ma con una struttura controintuitiva rispetto alle attese economiche. Un incremento dell'1% nella quantità di rinnovabili si traduce in un aumento contemporaneo del 0.48% del PUN, seguito da un effetto negativo di 0.42% nel periodo successivo.

Questo risultato appare in contrasto con la teoria economica standard, secondo cui un aumento di produzione da fonti rinnovabili (caratterizzate da costi variabili quasi nulli) dovrebbe esercitare una pressione al ribasso sul prezzo di mercato. Una possibile spiegazione risiede in una dinamica di compensazione del sistema elettrico. L'effetto negativo ritardato potrebbe poi riflettere il consolidamento dell'abbassamento dei prezzi una volta assorbito lo shock iniziale. Tuttavia, questa interpretazione richiede ulteriori verifiche attraverso analisi più sofisticate e dati ad alta frequenza sulla struttura del dispacciamento elettrico.

5.3.3 Diagnostica dei Residui

```
par(mfrow = c(3, 1))
ccf(tfm_renew$residuals, alfa_renew[2:length(alfa_renew)], lag.max = 20,
    main = "CCF: Residui TFM e Input Pre-whitened",
    col = "darkgreen", lwd = 2)
grid(col = "gray85", lty = "dotted")

acf(tfm_renew$residuals, lag.max = 35, main = "ACF Residui TFM Renew$\rightarrow$PUN",
    col = "steelblue", lwd = 2)
```

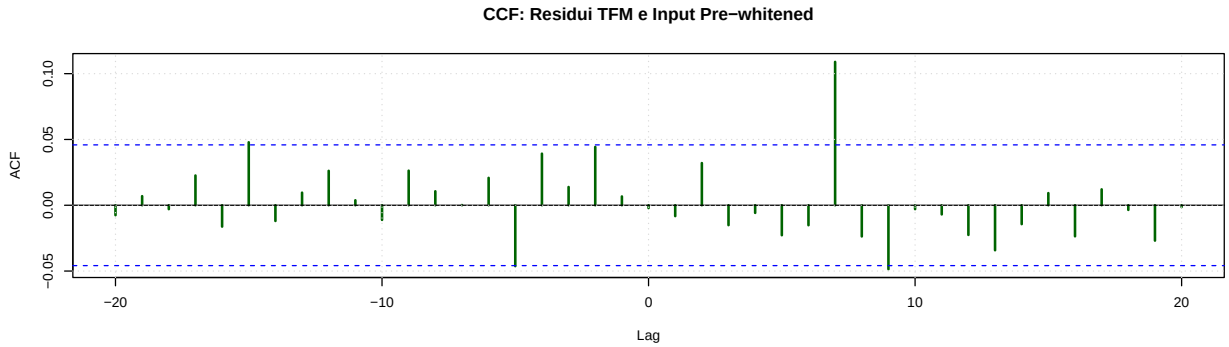
```
## Warning in title(main %||% if (i == j) snames[i] else paste(sn.abbr[i], : font
## width unknown for character 0x0d in encoding latin1
```

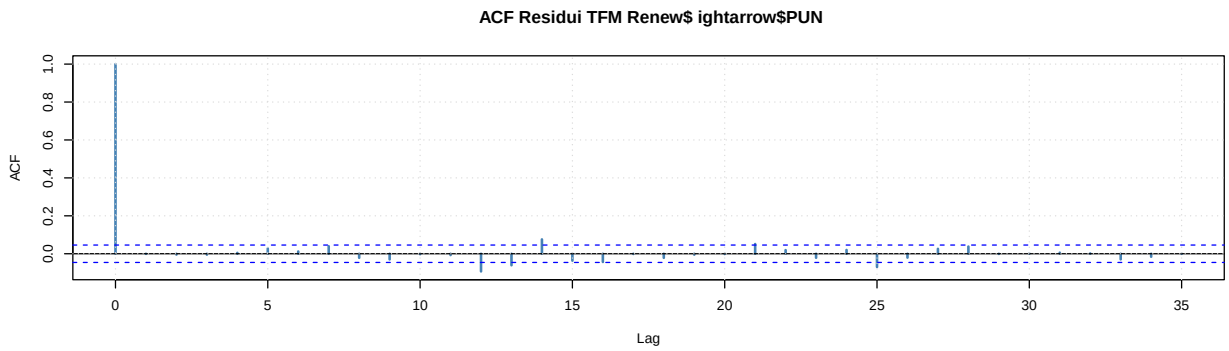
```
grid(col = "gray85", lty = "dotted")

pacf(tfm_renew$residuals, lag.max = 35, main = "PACF Residui TFM Renew$\rightarrow$PUN",
     col = "coral", lwd = 2)
```

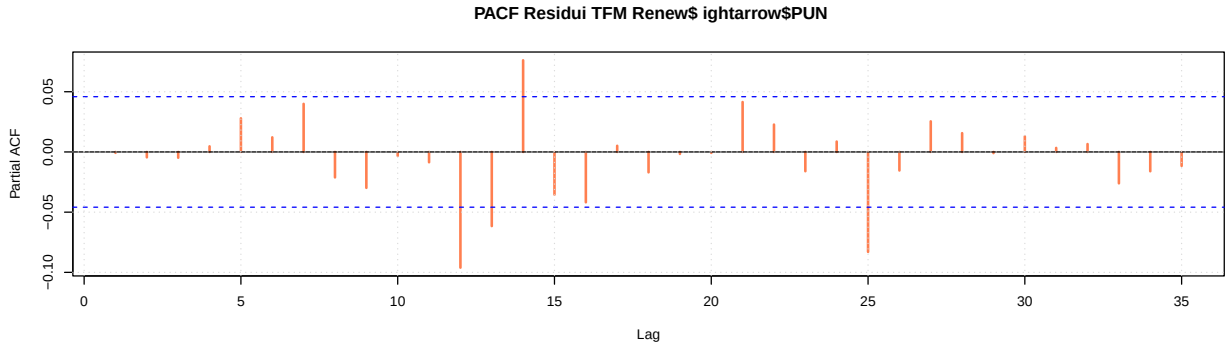
```
## Warning in title(main %||% if (i == j) snames[i] else paste(sn.abbr[i], : font
```

```
## width unknown for character 0x0d in encoding latin1
```

```
grid(col = "gray85", lty = "dotted")
```



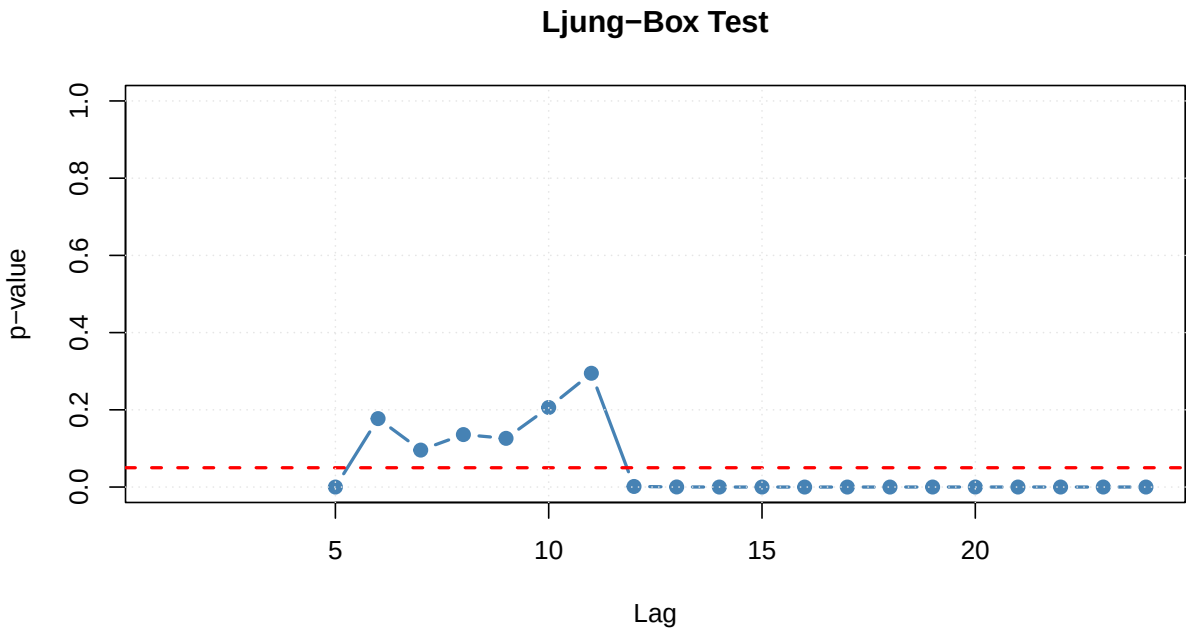





```
par(mfrow = c(1, 1))
```

```
ljung_box_plot(tfm_renew$residuals, max_lag = 24, fitdf = length(coef_renew))
```

```
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
## Warning in pchisq(STATISTIC, lag - fitdf): NaNs produced
```



I residui mostrano comportamento sostanzialmente coerente con un processo white noise, confermando che il modello ha catturato adeguatamente la struttura dinamica della relazione. La CCF con l'input pre-whitened non presenta autocorrelazioni significative, mentre ACF e PACF rimangono entro le bande di confidenza. Di nuovo, però, il test di Ljung-Box sostanzialmente rifiuta l'ipotesi nulla di assenza di autocorrelazione.

Entrambi i modelli confermano relazioni causali unidirezionali, in cui i valori presenti e passati delle variabili esplicative influenzano il valore attuale del PUN, senza evidenza di feedback inverso nel breve periodo. Questo risultato è coerente con la struttura del mercato elettrico italiano, dove il PUN si forma come variabile endogena in risposta ai costi di generazione (Gas) e alla disponibilità di fonti a costo marginale nullo (Rinnovabili).

6 Modelli VAR

In questa sezione estendiamo l'analisi multivariata utilizzando modelli Autoregressivi Vettoriali (VAR) sulle serie destagionalizzate. I modelli VAR permettono di catturare le interdipendenze dinamiche tra le tre variabili del sistema (prezzi del gas, prezzi dell'energia elettrica e produzione da rinnovabili) senza imporre a priori una struttura causale.

Per svolgere successivamente le analisi IRF e FEVD è necessario ordinare le variabili. L'ordinamento scelto riflette la struttura economica del sistema: il prezzo del Gas è considerato la variabile più esogena, determinata da mercati internazionali e non influenzata nel breve periodo dal mercato elettrico nazionale. La produzione da rinnovabili, dipendente principalmente da fattori meteo e di capacità installata, non reagisce istantaneamente ai prezzi dell'energia ma può essere influenzata dal prezzo del Gas per quanto riguarda la quantità effettivamente immessa nel mercato. Il prezzo dell'energia risulta la variabile più endogena, reagendo immediatamente a shock su Gas e Rinnovabili.

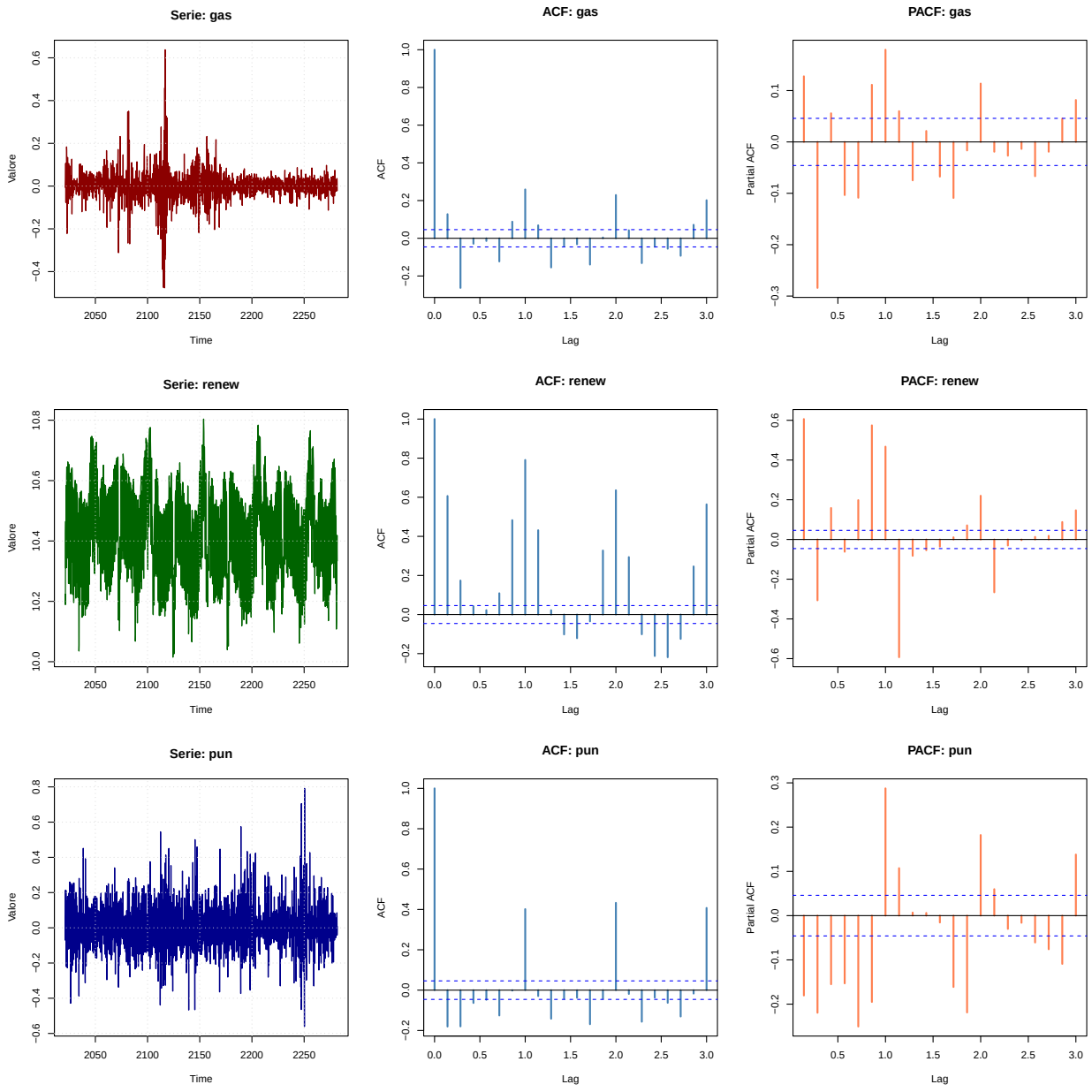
6.1 Preparazione dei Dati

Per il gas e il PUN, non stazionarie, applichiamo la differenza prima, mentre per le rinnovabili utilizziamo direttamente il logaritmo.

```
# Trasformazioni
l_gas_diff <- diff(log(gas_ts))
l_pun_diff <- diff(log(pun_ts))
l_renew <- log(renew_ts)

# Allineamento serie (diff rimuove la prima osservazione)
l_renew_aligned <- window(l_renew, start = start(l_gas_diff))

# Dataset per il VAR
df_var <- ts.union(l_gas_diff, l_renew_aligned, l_pun_diff)
colnames(df_var) <- c("gas", "renew", "pun")
```

Prima di procedere con la destagionalizzazione, esaminiamo le proprietà delle serie grezze.

```
par(mfrow = c(3, 3))
for(i in 1:NCOL(df_var)){
  plot.ts(df_var[,i], main = paste("Serie:", colnames(df_var)[i]),
          ylab = "Valore", col = c("darkred", "darkgreen", "darkblue")[i], lwd = 1.2)
  grid(col = "gray85", lty = "dotted")
  acf(df_var[,i], lag.max = 21, main = paste("ACF:", colnames(df_var)[i]),
      col = "steelblue", lwd = 2)
  pacf(df_var[,i], lag.max = 21, main = paste("PACF:", colnames(df_var)[i]),
       col = "coral", lwd = 2)
}
```




```
par(mfrow = c(1, 1))
```

Coerentemente con le analisi precedenti, le serie presentano una evidente componente stagionale con frequenza settimanale.

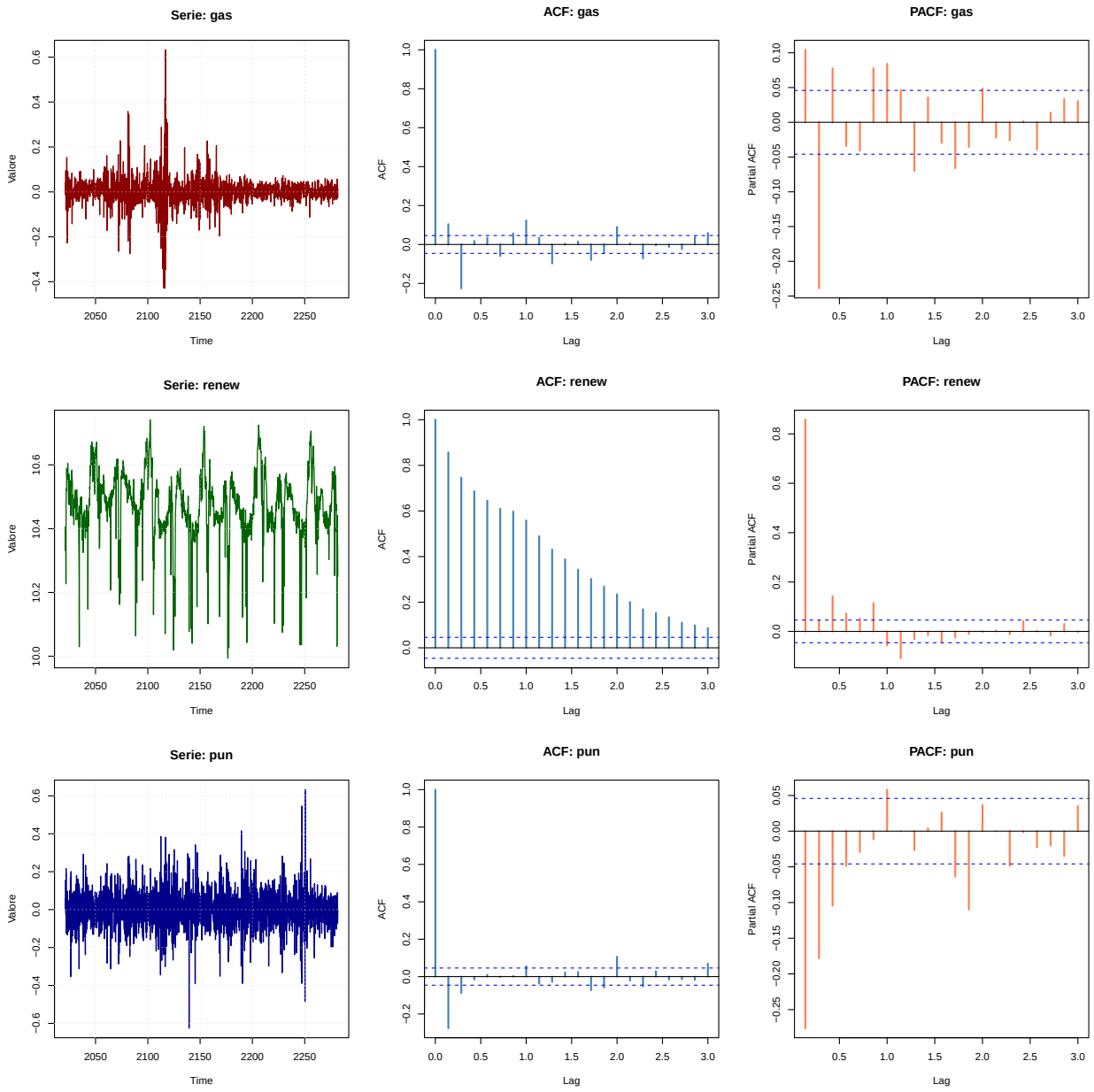
6.2 Destagionalizzazione con STL

Per rimuovere la componente stagionale dalle serie, utilizziamo la decomposizione STL (Seasonal and Trend decomposition using Loess) con finestra stagionale periodica.

```
gas_ds <- seasadj(stl(df_var[, "gas"], s.window = "periodic"))
renew_ds <- seasadj(stl(df_var[, "renew"], s.window = "periodic"))
pun_ds <- seasadj(stl(df_var[, "pun"], s.window = "periodic"))
```

```r
# Ricrea dataset con serie destagionalizzate
df_var <- ts.union(gas_ds, renew_ds, pun_ds)
colnames(df_var) <- c("gas", "renew", "pun")
```

```r
par(mfrow = c(3, 3))
for(i in 1:NCOL(df_var)){
  plot.ts(df_var[,i], main = paste("Serie:", colnames(df_var)[i]),
          ylab = "Valore", col = c("darkred", "darkgreen", "darkblue")[i], lwd = 1.2)
  grid(col = "gray85", lty = "dotted")
  acf(df_var[,i], lag.max = 21, main = paste("ACF:", colnames(df_var)[i]),
      col = "steelblue", lwd = 2)
  pacf(df_var[,i], lag.max = 21, main = paste("PACF:", colnames(df_var)[i]),
       col = "coral", lwd = 2)
}
```




```
par(mfrow = c(1, 1))
```

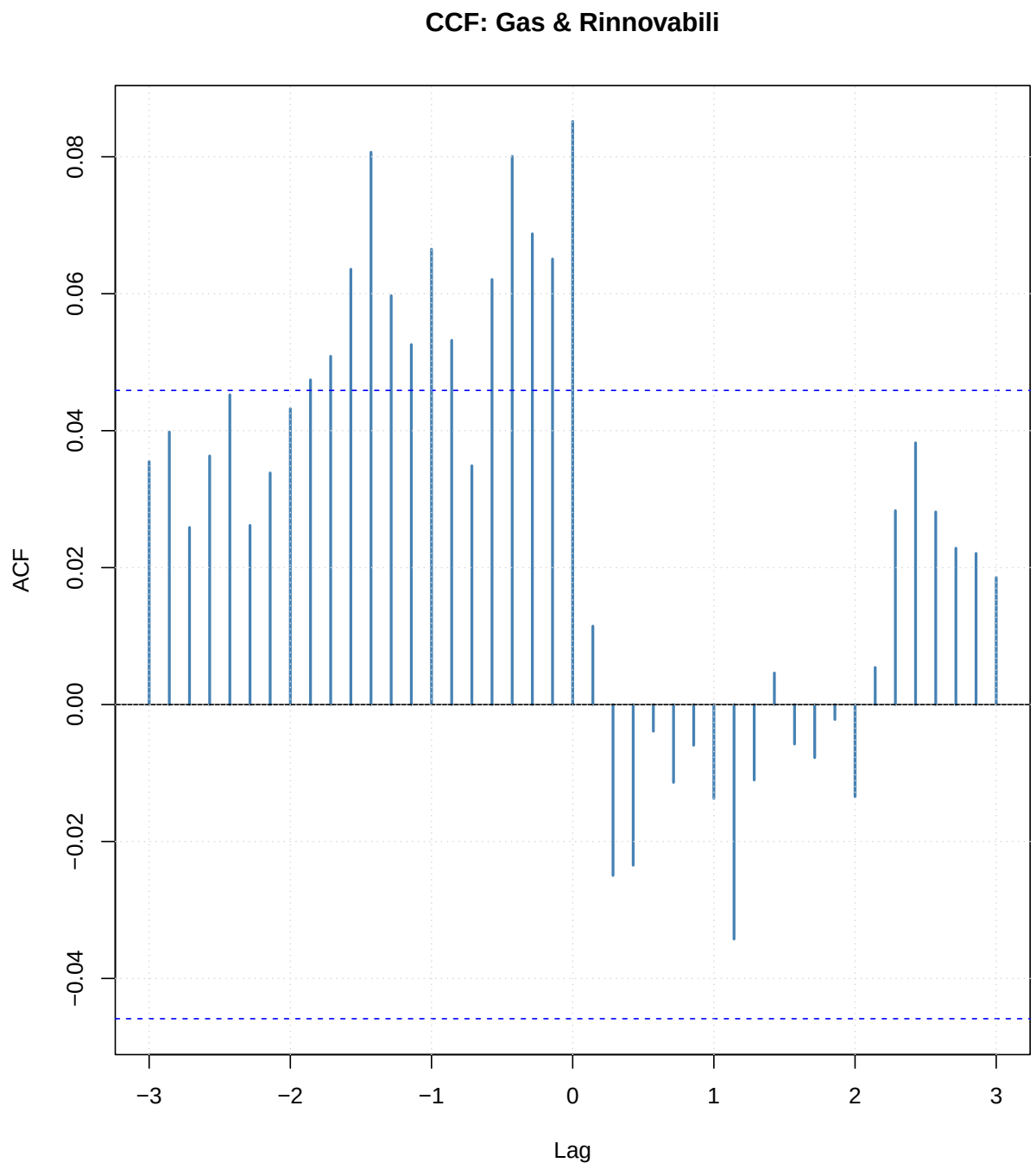
La rimozione della stagionalità ha effetti evidenti: le serie mostrano ora chiaramente le rispettive dinamiche autoregressive (AR) e di media mobile (MA), depurate dalle oscillazioni settimanali che dominavano i grafici precedenti.

6.3 Analisi delle Cross-Correlazioni

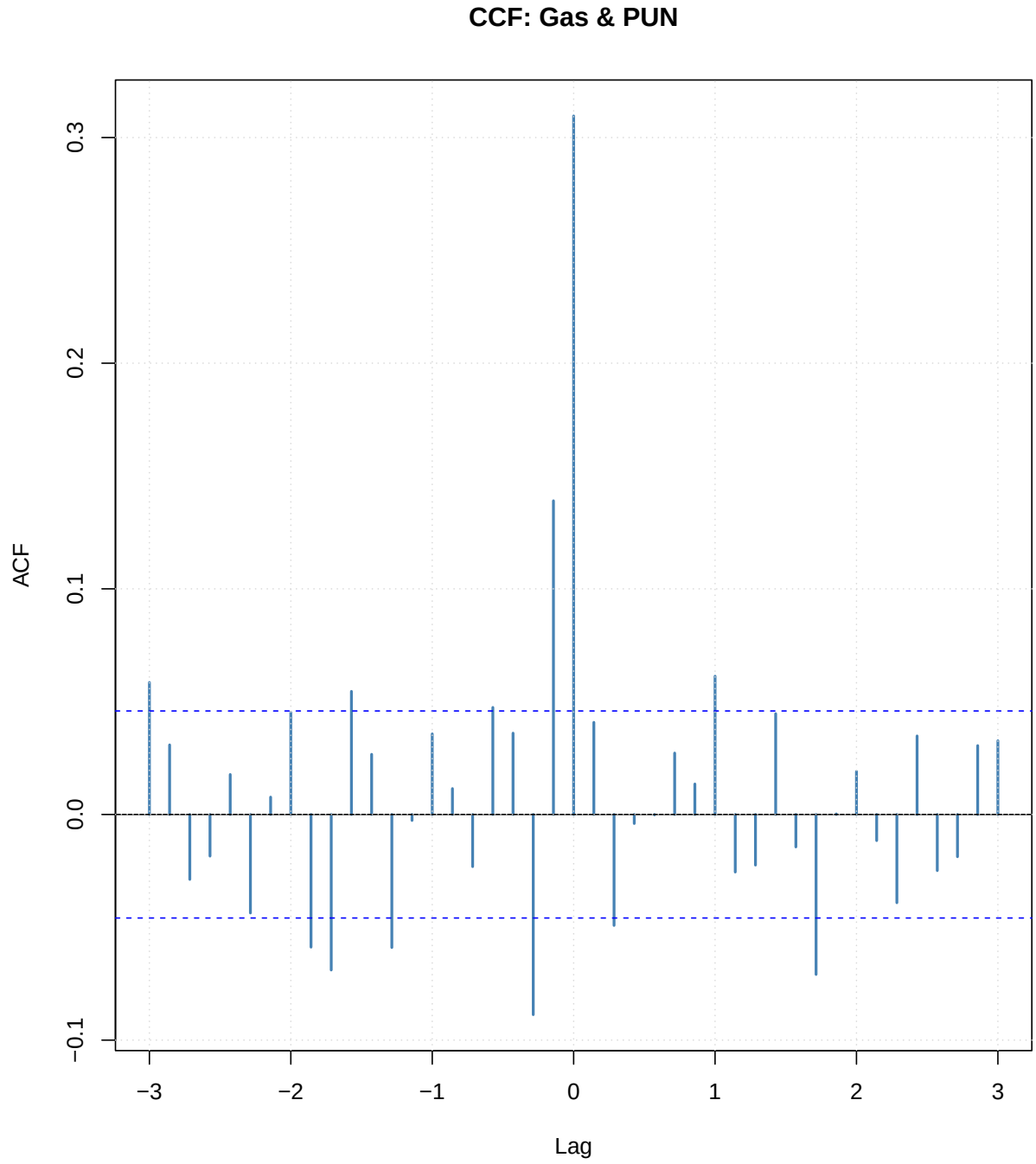
Le funzioni di cross-correlazione (CCF) identificano potenziali relazioni di lead-lag tra le variabili, fornendo indicazioni preliminari sulle direzioni causali.

```
ccf(df_var[, "gas"], df_var[, "renew"], lag.max = 21,
    main = "CCF: Gas & Rinnovabili", col = "steelblue", lwd = 2)
```

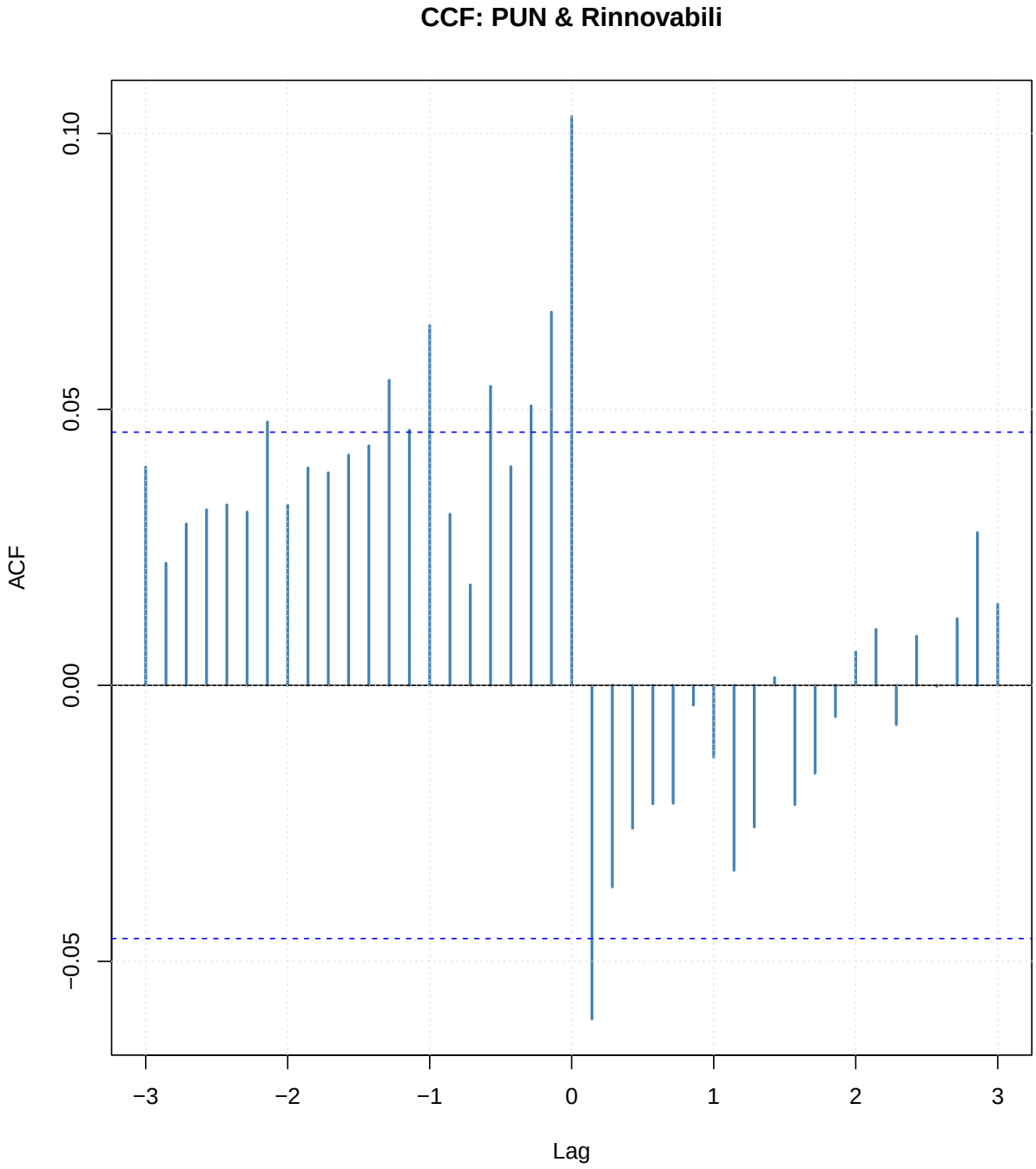
```
grid(col = "gray85", lty = "dotted")
```




```
ccf(df_var[, "gas"], df_var[, "pun"], lag.max = 21,
    main = "CCF: Gas & PUN", col = "steelblue", lwd = 2)
grid(col = "gray85", lty = "dotted")
```




```
ccf(df_var[, "pun"], df_var[, "renew"], lag.max = 21,
    main = "CCF: PUN & Rinnovabili", col = "steelblue", lwd = 2)
grid(col = "gray85", lty = "dotted")
```



Tutte le coppie di variabili mostrano una correlazione contemporanea significativa. Il PUN risulta influenzato sia dal prezzo del Gas (relazione prevalentemente unilaterale) sia dalle Rinnovabili. Si osserva inoltre un'influenza unidirezionale dalle Rinnovabili al Gas.

6.4 Selezione dell'Ordine di Ritardo

Utilizziamo criteri di informazione multipli (AIC, HQ, SC/BIC, FPE) per determinare il numero ottimale di ritardi da includere nel modello VAR.

```r
IC_selection <- VARselect(df_var, lag.max = 21, type = "const")
print(IC_selection$selection)
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     13      6      3     13
```

I criteri informativi suggeriscono ordini differenti. Si adotta la specificazione più parsimoniosa indicata dal BIC, ma data la presenza di una marcata stagionalità settimanale, si procede alla stima, oltre che del modello VAR(3), anche dei modelli VAR(7) e VAR(14).

6.5 Stima dei Modelli VAR

Per ciascun ordine di ritardo, si stimano modelli VAR completi e ridotti. Il modello ridotto applica restrizioni sui coefficienti non significativi mediante il metodo della significatività sequenziale (SER). Il test del rapporto di verosimiglianza determina quale specificazione sia preferibile.

6.5.1 VAR(3)

```r
mod_var3 <- vars::VAR(df_var, p = 3, type = "const")
mod_var3_res <- restrict(mod_var3, method = "ser", thresh = 1.96)
```

```r
lrtest(mod_var3, mod_var3_res)
```

```
## Likelihood ratio test
##
## Model 1: vars::VAR(y = df_var, p = 3, type = "const")
## Model 2: vars::VAR(y = df_var, p = 3, type = "const")
##   #Df LogLik  Df Chisq Pr(>Chisq)
## 1  30   7515
## 2  16   7506 -14  18.4       0.19
```

```r
summary(mod_var3_res)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: gas, renew, pun
## Deterministic variables: const
## Sample size: 1822
## Log Likelihood: 7506.07
## Roots of the characteristic polynomial:
## 0.908 0.575 0.575 0.536 0.536 0.516 0.399 0.399 0.27
## Call:
## vars::VAR(y = df_var, p = 3, type = "const")
##
##
## Estimation results for equation gas:
## ====================================
## gas = gas.l1 + renew.l1 + gas.l2 + gas.l3 + renew.l3
##
```

```
##          Estimate Std. Error t value Pr(>|t|)
## gas.l1     0.1405     0.0236    5.95  3.2e-09 ***
## renew.l1   0.0386     0.0182    2.12  0.03443 *
## gas.l2    -0.2519     0.0229  -10.99  < 2e-16 ***
## gas.l3     0.0774     0.0234    3.31  0.00095 ***
## renew.l3  -0.0386     0.0182   -2.11  0.03457 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0559 on 1817 degrees of freedom
## Multiple R-Squared: 0.0753,  Adjusted R-squared: 0.0727
## F-statistic: 29.6 on 5 and 1817 DF,  p-value: <2e-16
##
##
## Estimation results for equation renew:
## ======================================
## renew = renew.l1 + renew.l2 + renew.l3 + const
##
##          Estimate Std. Error t value Pr(>|t|)
## renew.l1   0.8194     0.0232   35.27  < 2e-16 ***
## renew.l2  -0.0787     0.0301   -2.62    0.009 **
## renew.l3   0.1443     0.0232    6.21  6.4e-10 ***
## const      1.2041     0.1334    9.03  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0521 on 1818 degrees of freedom
## Multiple R-Squared:    1,    Adjusted R-squared:    1
## F-statistic: 1.84e+07 on 4 and 1818 DF,  p-value: <2e-16
##
##
## Estimation results for equation pun:
## ====================================
## pun = gas.l1 + pun.l1 + pun.l2 + gas.l3 + renew.l3 + pun.l3 + const
##
##          Estimate Std. Error t value Pr(>|t|)
## gas.l1     0.5544     0.0394   14.07  < 2e-16 ***
## pun.l1    -0.4621     0.0235  -19.63  < 2e-16 ***
## pun.l2    -0.3033     0.0241  -12.60  < 2e-16 ***
## gas.l3     0.2874     0.0404    7.12  1.5e-12 ***
## renew.l3  -0.0486     0.0205   -2.37    0.018 *
## pun.l3    -0.1708     0.0239   -7.16  1.2e-12 ***
## const      0.5097     0.2147    2.37    0.018 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
##
## Residual standard error: 0.089 on 1815 degrees of freedom
## Multiple R-Squared: 0.21,    Adjusted R-squared: 0.207
## F-statistic: 68.7 on 7 and 1815 DF,  p-value: <2e-16
##
##
##
## Covariance matrix of residuals:
##            gas    renew     pun
## gas   0.003137 0.000489 0.00163
## renew 0.000489 0.002725 0.00151
## pun   0.001635 0.001512 0.00793
##
## Correlation matrix of residuals:
##         gas renew   pun
## gas   1.000 0.167 0.328
## renew 0.167 1.000 0.325
## pun   0.328 0.325 1.000
```

6.5.2 VAR(7)

```
mod_var7 <- vars::VAR(df_var, p = 7, type = "const")
mod_var7_res <- restrict(mod_var7, method = "ser", thresh = 1.96)
```

```
lrtest(mod_var7, mod_var7_res)
```

```
## Likelihood ratio test
##
## Model 1: vars::VAR(y = df_var, p = 7, type = "const")
## Model 2: vars::VAR(y = df_var, p = 7, type = "const")
##   #Df LogLik  Df Chisq Pr(>Chisq)
## 1  66   7622
## 2  31   7602 -35  39.7       0.27
```

```
summary(mod_var7_res)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: gas, renew, pun
## Deterministic variables: const
## Sample size: 1818
## Log Likelihood: 7602.26
## Roots of the characteristic polynomial:
## 0.94 0.76 0.76 0.756 0.756 0.751 0.751 0.751 0.701 0.701 0.695 0.695 0.682 0.682 0.666 0.657
## Call:
## vars::VAR(y = df_var, p = 7, type = "const")
##
##
```

```
## Estimation results for equation gas:
## ====================================
## gas = gas.l1 + renew.l1 + gas.l2 + gas.l3 + renew.l3 + gas.l6 + gas.l7
##
##          Estimate Std. Error t value Pr(>|t|)
## gas.l1     0.1393     0.0236    5.91  4.1e-09 ***
## renew.l1   0.0436     0.0182    2.40   0.0166 *
## gas.l2    -0.2498     0.0229  -10.92  < 2e-16 ***
## gas.l3     0.0715     0.0232    3.08   0.0021 **
## renew.l3  -0.0435     0.0182   -2.40   0.0166 *
## gas.l6     0.0668     0.0226    2.96   0.0032 **
## gas.l7     0.0918     0.0226    4.05  5.2e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0554 on 1811 degrees of freedom
## Multiple R-Squared: 0.0903,  Adjusted R-squared: 0.0868
## F-statistic: 25.7 on 7 and 1811 DF,  p-value: <2e-16
##
##
## Estimation results for equation renew:
## ======================================
## renew = renew.l1 + renew.l2 + gas.l3 + renew.l3 + pun.l4 + renew.l6 + renew.l7 + pun.l7 + co
##
##          Estimate Std. Error t value Pr(>|t|)
## renew.l1   0.8069     0.0233   34.63  < 2e-16 ***
## renew.l2  -0.0817     0.0297   -2.75   0.0061 **
## gas.l3     0.0455     0.0208    2.19   0.0286 *
## renew.l3   0.0794     0.0245    3.25   0.0012 **
## pun.l4     0.0257     0.0121    2.13   0.0332 *
## renew.l6   0.1585     0.0245    6.47  1.2e-10 ***
## renew.l7  -0.0584     0.0235   -2.49   0.0130 *
## pun.l7     0.0344     0.0120    2.85   0.0044 **
## const      0.9984     0.1403    7.12  1.6e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0509 on 1809 degrees of freedom
## Multiple R-Squared:    1,    Adjusted R-squared:    1
## F-statistic: 8.54e+06 on 9 and 1809 DF,  p-value: <2e-16
##
##
## Estimation results for equation pun:
## ====================================
## pun = gas.l1 + renew.l1 + pun.l1 + gas.l2 + pun.l2 + gas.l3 + pun.l3 + gas.l4 + pun.l4 + gas
##
```

```
##          Estimate Std. Error t value Pr(>|t|)
## gas.l1     0.5918     0.0391   15.13  < 2e-16 ***
## renew.l1   0.0707     0.0254    2.78  0.00550 **
## pun.l1    -0.5484     0.0255  -21.49  < 2e-16 ***
## gas.l2     0.1517     0.0412    3.68  0.00024 ***
## pun.l2    -0.4347     0.0288  -15.08  < 2e-16 ***
## gas.l3     0.3825     0.0427    8.96  < 2e-16 ***
## pun.l3    -0.3535     0.0301  -11.74  < 2e-16 ***
## gas.l4     0.2689     0.0431    6.24  5.5e-10 ***
## pun.l4    -0.2573     0.0300   -8.59  < 2e-16 ***
## gas.l5     0.1731     0.0429    4.03  5.8e-05 ***
## renew.l5  -0.0707     0.0254   -2.78  0.00552 **
## pun.l5    -0.1703     0.0280   -6.09  1.4e-09 ***
## gas.l6     0.1750     0.0406    4.31  1.7e-05 ***
## pun.l6    -0.1084     0.0248   -4.38  1.3e-05 ***
## gas.l7     0.0794     0.0372    2.13  0.03299 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.087 on 1803 degrees of freedom
## Multiple R-Squared: 0.248,   Adjusted R-squared: 0.242
## F-statistic: 39.6 on 15 and 1803 DF,  p-value: <2e-16
##
##
##
## Covariance matrix of residuals:
##             gas    renew     pun
## gas   0.003100 0.000444 0.00164
## renew 0.000444 0.002613 0.00146
## pun   0.001636 0.001456 0.00759
##
## Correlation matrix of residuals:
##         gas renew   pun
## gas   1.000 0.156 0.337
## renew 0.156 1.000 0.327
## pun   0.337 0.327 1.000
```

6.5.3 VAR(14)

```
mod_var14 <- vars::VAR(df_var, p = 14, type = "const")
mod_var14_res <- restrict(mod_var14, method = "ser", thresh = 1.96)
```

```
lrtest(mod_var14, mod_var14_res)
```

```
## Likelihood ratio test
##
## Model 1: vars::VAR(y = df_var, p = 14, type = "const")
```

```
## Model 2: vars::VAR(y = df_var, p = 14, type = "const")
##   #Df LogLik  Df Chisq Pr(>Chisq)
## 1 129   7688
## 2  49   7652 -80  73.2       0.69
```

```
summary(mod_var14_res)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: gas, renew, pun
## Deterministic variables: const
## Sample size: 1811
## Log Likelihood: 7651.675
## Roots of the characteristic polynomial:
## 0.899 0.899 0.885 0.885 0.883 0.883 0.872 0.872 0.871 0.871 0.858 0.858 0.84 0.84 0.839 0.83
## Call:
## vars::VAR(y = df_var, p = 14, type = "const")
##
##
## Estimation results for equation gas:
## ====================================
## gas = gas.l1 + renew.l1 + gas.l2 + gas.l3 + renew.l3 + gas.l6 + gas.l7 + gas.l8 + renew.l8 +
##
##          Estimate Std. Error t value Pr(>|t|)
## gas.l1     0.1415     0.0236    5.99  2.5e-09 ***
## renew.l1   0.0509     0.0191    2.66  0.00779 **
## gas.l2    -0.2354     0.0229  -10.30  < 2e-16 ***
## gas.l3     0.0758     0.0232    3.26  0.00113 **
## renew.l3  -0.0495     0.0208   -2.38  0.01728 *
## gas.l6     0.0857     0.0232    3.69  0.00023 ***
## gas.l7     0.0697     0.0235    2.97  0.00302 **
## gas.l8     0.0560     0.0236    2.37  0.01770 *
## renew.l8  -0.0878     0.0264   -3.32  0.00091 ***
## gas.l9    -0.0688     0.0232   -2.96  0.00307 **
## renew.l9   0.0864     0.0254    3.40  0.00069 ***
## pun.l11   -0.0298     0.0134   -2.23  0.02609 *
## pun.l12   -0.0364     0.0134   -2.71  0.00675 **
## gas.l14    0.0575     0.0225    2.56  0.01053 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0546 on 1797 degrees of freedom
## Multiple R-Squared: 0.109,   Adjusted R-squared: 0.103
## F-statistic: 15.8 on 14 and 1797 DF,  p-value: <2e-16
##
##
```

```
## Estimation results for equation renew:
## ======================================
## renew = renew.l1 + renew.l2 + renew.l3 + pun.l4 + gas.l5 + renew.l6 + renew.l8 + gas.l10 + r
##
##           Estimate Std. Error t value Pr(>|t|)
## renew.l1    0.8021     0.0231   34.76  < 2e-16 ***
## renew.l2   -0.0741     0.0297   -2.49  0.01279 *
## renew.l3    0.0954     0.0245    3.89  0.00010 ***
## pun.l4      0.0271     0.0121    2.24  0.02530 *
## gas.l5     -0.0422     0.0209   -2.02  0.04312 *
## renew.l6    0.1786     0.0202    8.86  < 2e-16 ***
## renew.l8   -0.0774     0.0206   -3.75  0.00018 ***
## gas.l10     0.0408     0.0206    1.98  0.04778 *
## renew.l11  -0.0393     0.0168   -2.34  0.01957 *
## const       1.2003     0.1489    8.06  1.4e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0507 on 1801 degrees of freedom
## Multiple R-Squared:    1,    Adjusted R-squared:    1
## F-statistic: 7.74e+06 on 10 and 1801 DF,  p-value: <2e-16
##
##
## Estimation results for equation pun:
## ====================================
## pun = gas.l1 + renew.l1 + pun.l1 + gas.l2 + pun.l2 + gas.l3 + pun.l3 + gas.l4 + pun.l4 + gas
##
##          Estimate Std. Error t value Pr(>|t|)
## gas.l1     0.6018     0.0393   15.33  < 2e-16 ***
## renew.l1   0.0741     0.0232    3.19  0.00142 **
## pun.l1    -0.5678     0.0254  -22.35  < 2e-16 ***
## gas.l2     0.1955     0.0417    4.69  3.0e-06 ***
## pun.l2    -0.4599     0.0292  -15.75  < 2e-16 ***
## gas.l3     0.4215     0.0434    9.71  < 2e-16 ***
## pun.l3    -0.3988     0.0313  -12.73  < 2e-16 ***
## gas.l4     0.3115     0.0440    7.07  2.2e-12 ***
## pun.l4    -0.3205     0.0325   -9.87  < 2e-16 ***
## gas.l5     0.2326     0.0446    5.21  2.1e-07 ***
## pun.l5    -0.2559     0.0328   -7.81  9.4e-15 ***
## gas.l6     0.2798     0.0451    6.21  6.6e-10 ***
## pun.l6    -0.2079     0.0322   -6.47  1.3e-10 ***
## gas.l7     0.1519     0.0420    3.62  0.00031 ***
## pun.l7    -0.1352     0.0319   -4.24  2.3e-05 ***
## gas.l8     0.1455     0.0447    3.25  0.00117 **
## renew.l8  -0.0740     0.0232   -3.19  0.00144 **
## pun.l8    -0.1526     0.0292   -5.23  1.9e-07 ***
## pun.l9    -0.1251     0.0270   -4.64  3.7e-06 ***
```

```
## gas.l10    0.0991     0.0418    2.37  0.01778 *
## pun.l10   -0.1150     0.0281   -4.09  4.6e-05 ***
## gas.l11    0.1244     0.0393    3.17  0.00156 **
## pun.l11   -0.1130     0.0268   -4.22  2.6e-05 ***
## pun.l12   -0.1350     0.0236   -5.71  1.3e-08 ***
## pun.l13   -0.1235     0.0218   -5.66  1.7e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0857 on 1786 degrees of freedom
## Multiple R-Squared: 0.276,   Adjusted R-squared: 0.266
## F-statistic: 27.2 on 25 and 1786 DF,  p-value: <2e-16
##
##
##
## Covariance matrix of residuals:
##            gas    renew     pun
## gas   0.003034 0.000451 0.00159
## renew 0.000451 0.002614 0.00148
## pun   0.001592 0.001481 0.00741
##
## Correlation matrix of residuals:
##         gas renew   pun
## gas   1.000 0.160 0.336
## renew 0.160 1.000 0.337
## pun   0.336 0.337 1.000
```

In tutti e tre i modelli il test del rapporto di verosimiglianza non consente di rifiutare l'ipotesi nulla, indicando che le restrizioni imposte non comportano una perdita significativa di informazione rispetto al modello non ristretto.

Emerge una marcata componente autoregressiva del Gas in tutte le specificazioni considerate, influenzato anche da alcuni ritardi delle Rinnovabili. Le Rinnovabili nel VAR(3) appaiono sostanzialmente autoregressive, mentre nei VAR(7) e VAR(14) emergono effetti significativi di Gas e PUN rispettivamente a lag medi e medio-lunghi. Il PUN si conferma come la variabile più endogena del sistema, risultando significativamente influenzato da tutte le variabili considerate.

6.6 Confronto dei Modelli tramite BIC

```r
bic_values <- c(
  VAR3 = BIC(mod_var3_res)[1],
  VAR7 = BIC(mod_var7_res)[1],
  VAR14 = BIC(mod_var14_res)[1]
)
print(sort(bic_values, decreasing = FALSE))
```

```
##   VAR7  VAR14   VAR3
## -14972 -14936 -14892
```

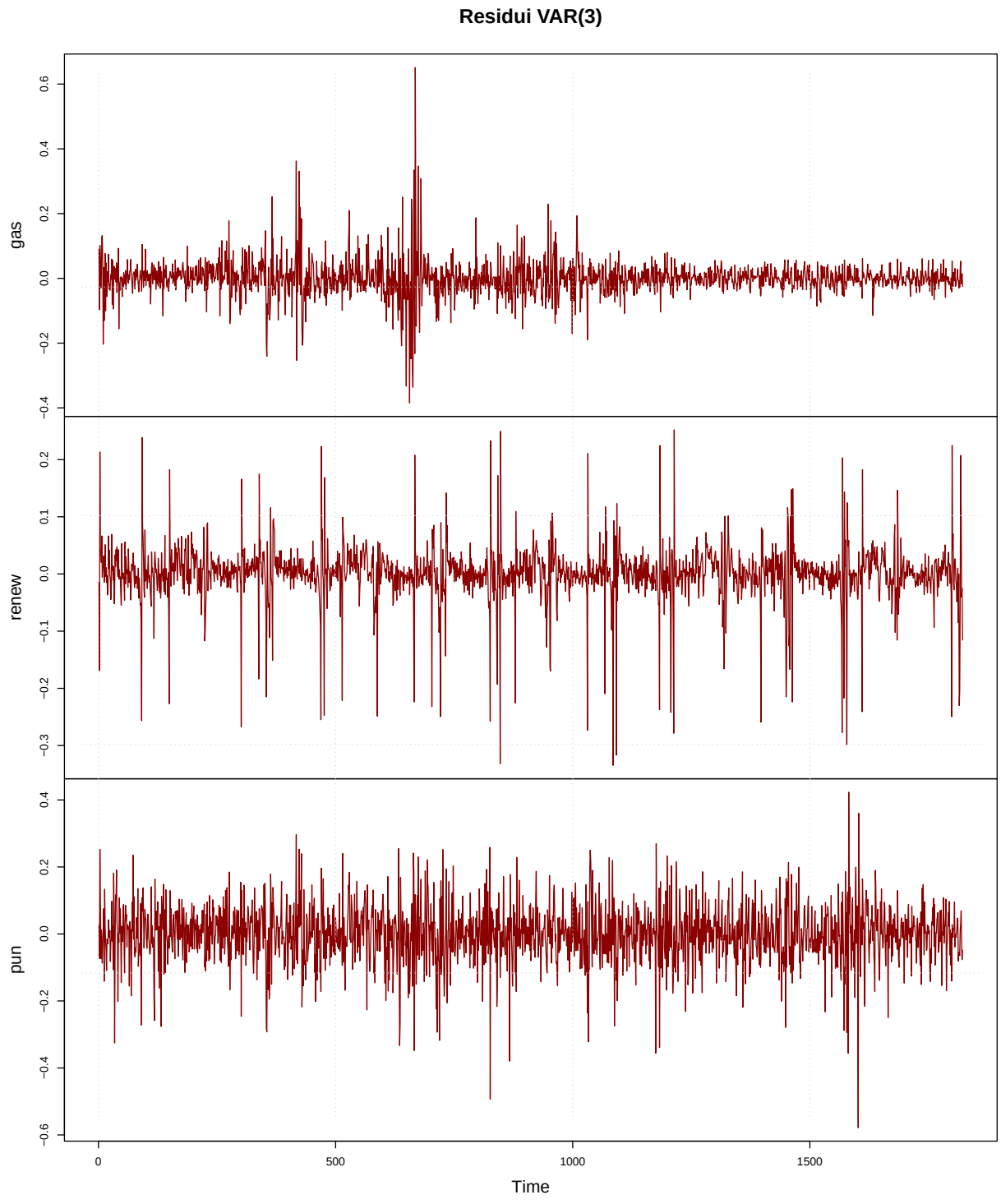
Il BIC seleziona il modello con ordine 7, confermando la presenza di un pattern settimanale.

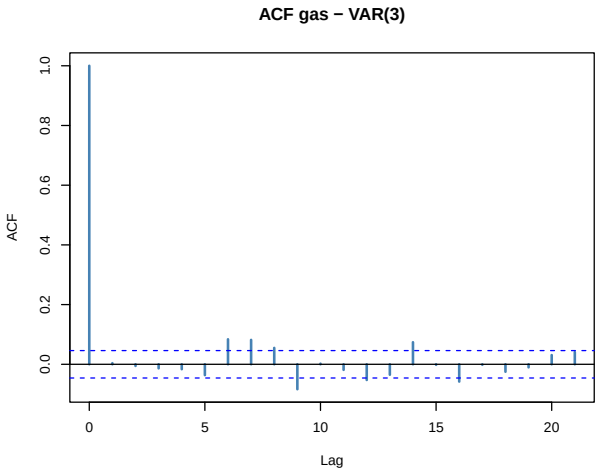
6.7 Diagnostica dei Residui

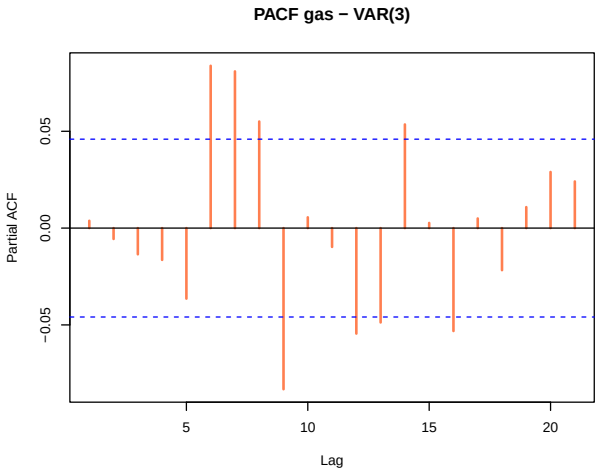
6.7.1 Analisi Grafica

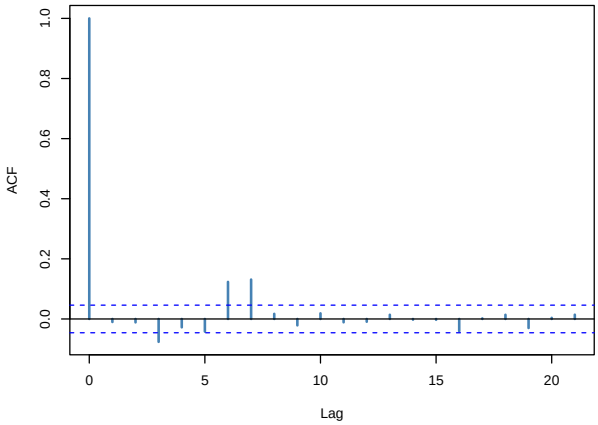
Esaminiamo graficamente i residui dei tre modelli per valutare l'adeguatezza della specificazione.

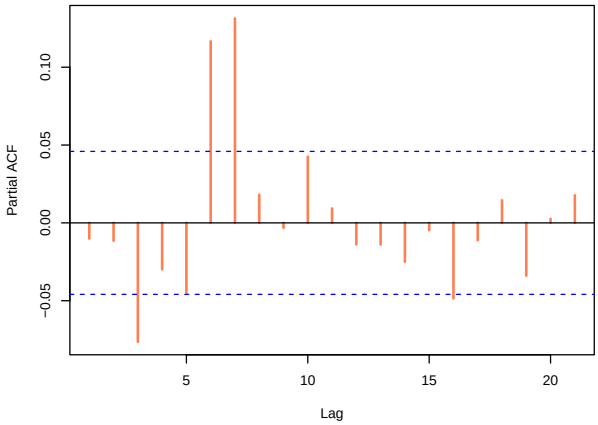
```
for(i in 1:3){
  par(mfrow = c(1, 1))
  plot.ts(res_var_list[[i]], main = paste("Residui", var_names[i]),
          col = c("darkred", "darkgreen", "darkblue"))
  grid(col = "gray85", lty = "dotted")

  par(mfrow = c(3, 2))
  for(j in 1:3){
    acf(res_var_list[[i]][, j], lag.max = 21,
        main = paste("ACF", colnames(res_var_list[[i]])[j], "-", var_names[i]),
        col = "steelblue", lwd = 2)
    pacf(res_var_list[[i]][, j], lag.max = 21,
         main = paste("PACF", colnames(res_var_list[[i]])[j], "-", var_names[i]),
         col = "coral", lwd = 2)
  }
  par(mfrow = c(1, 1))
}
```

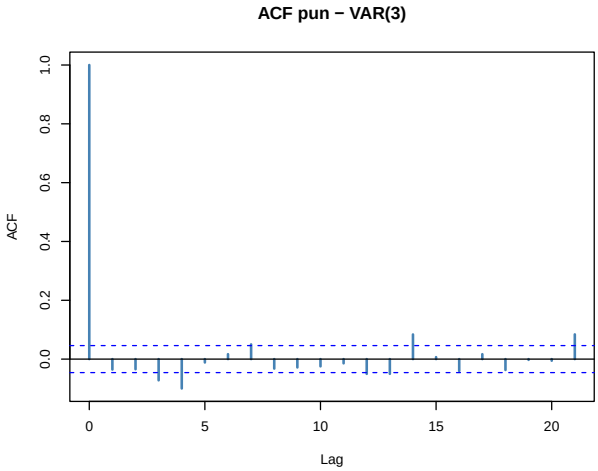

Residui VAR(3)
gas
renew
pun
Time
0
500
1000
1500

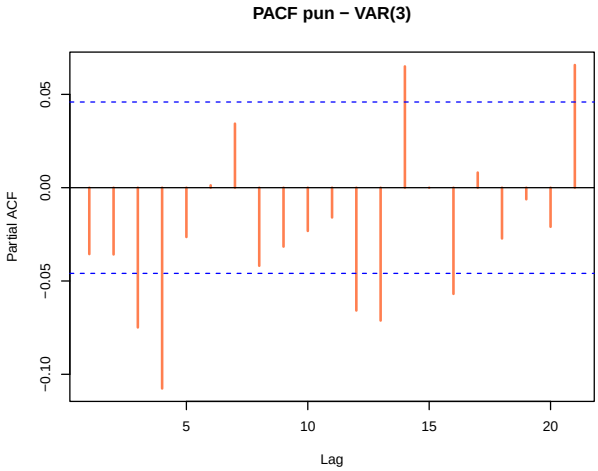

ACF gas − VAR(3)
ACF
Lag

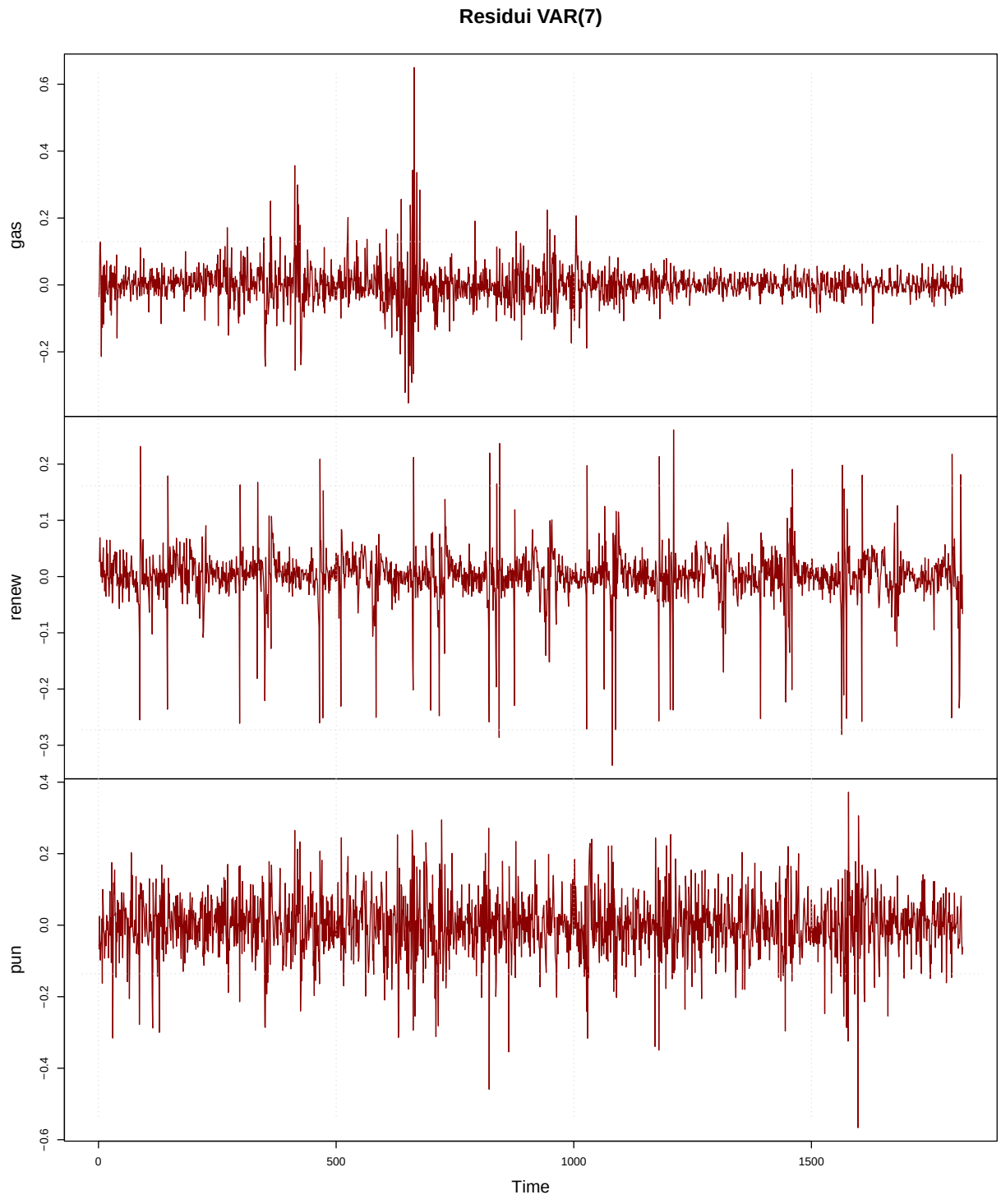

PACF gas − VAR(3)
Partial ACF
Lag

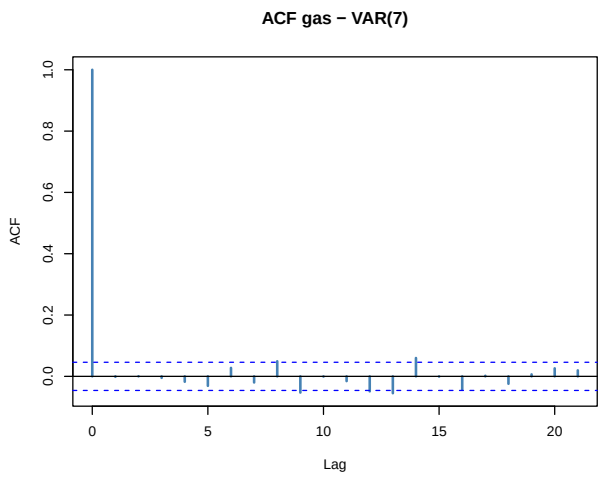

ACF renew − VAR(3)
ACF
Lag

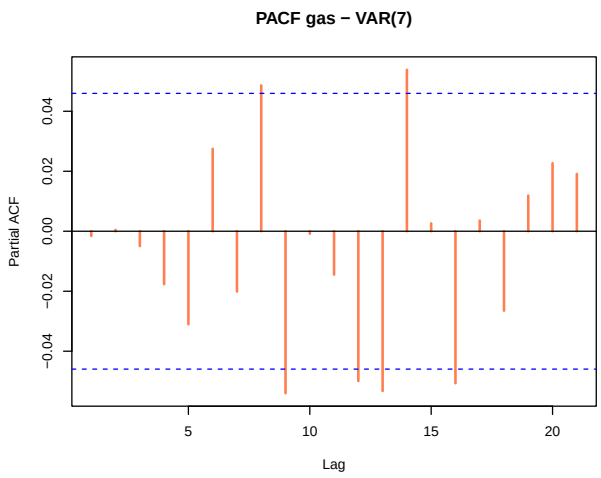

PACF renew − VAR(3)
Partial ACF
Lag

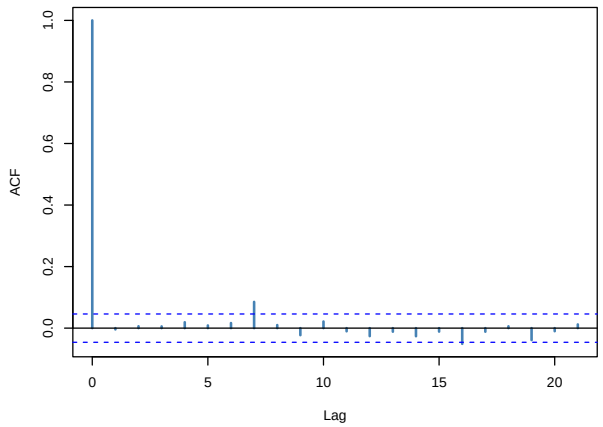

ACF pun − VAR(3)
ACF
Lag

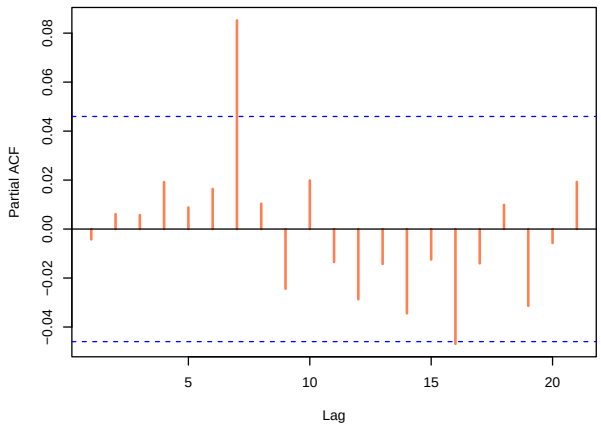

PACF pun − VAR(3)
Partial ACF
Lag

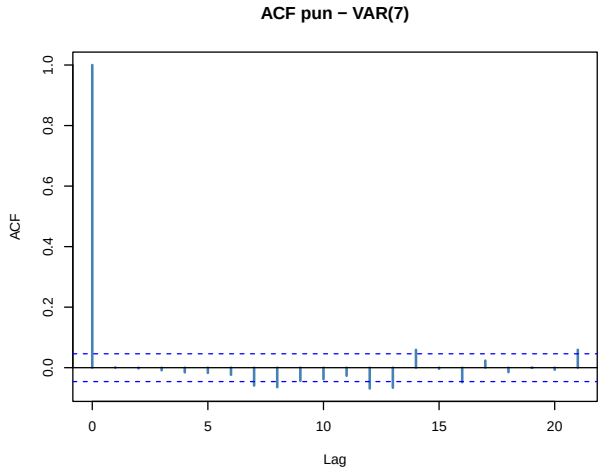

Residui VAR(7)
gas
renew
pun
Time
0
500
1000
1500

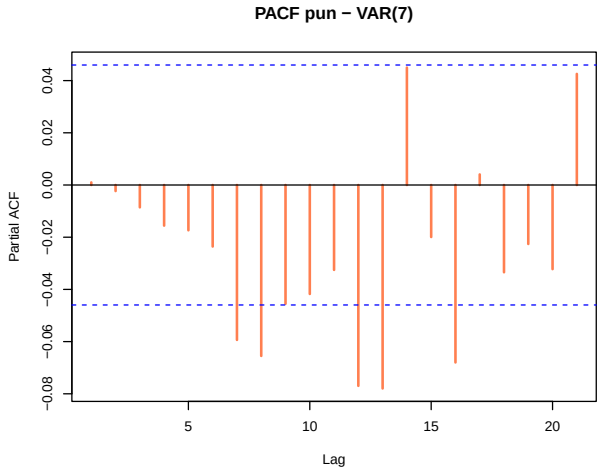

ACF gas − VAR(7)
ACF
Lag

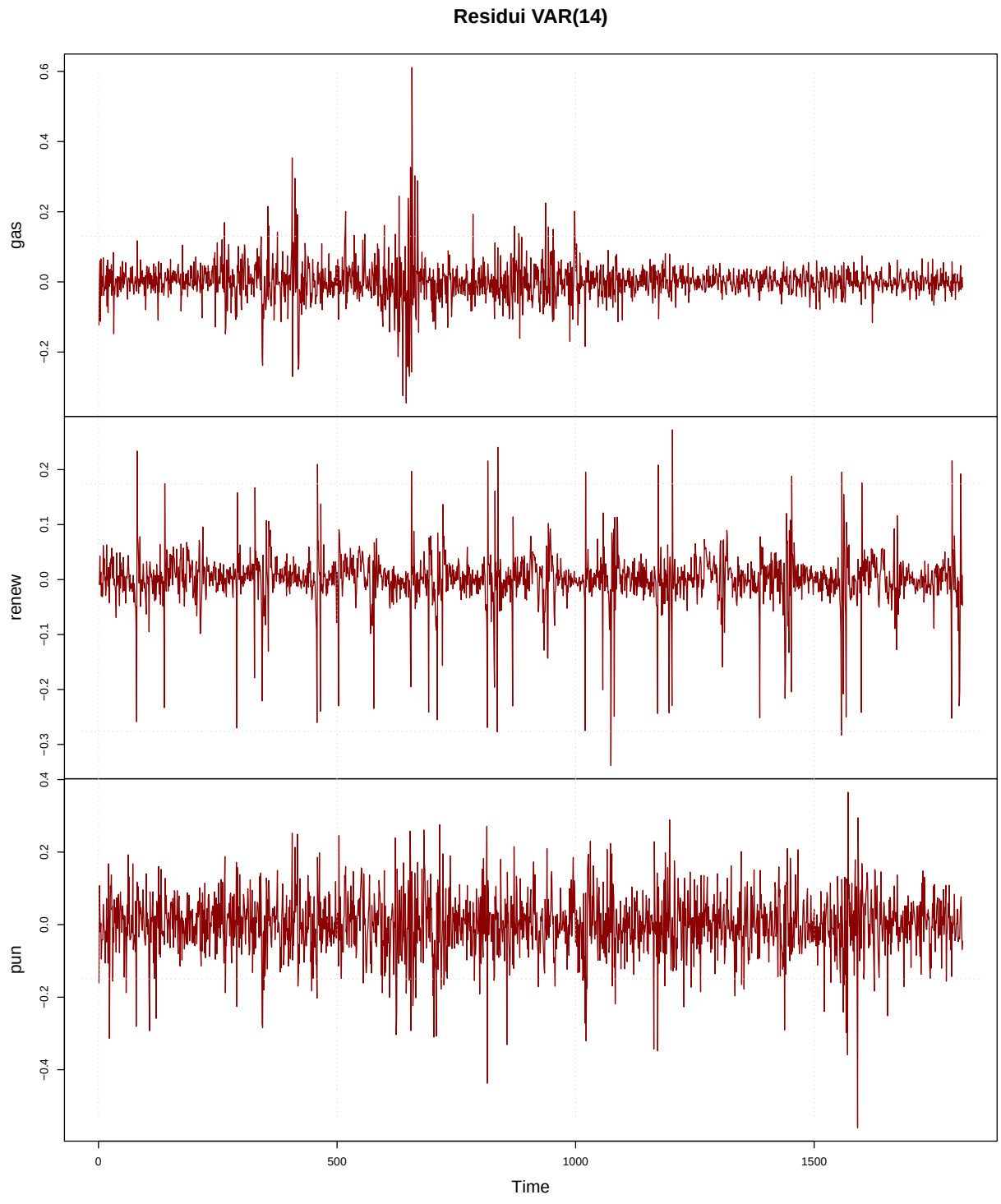

PACF gas − VAR(7)
Partial ACF
Lag

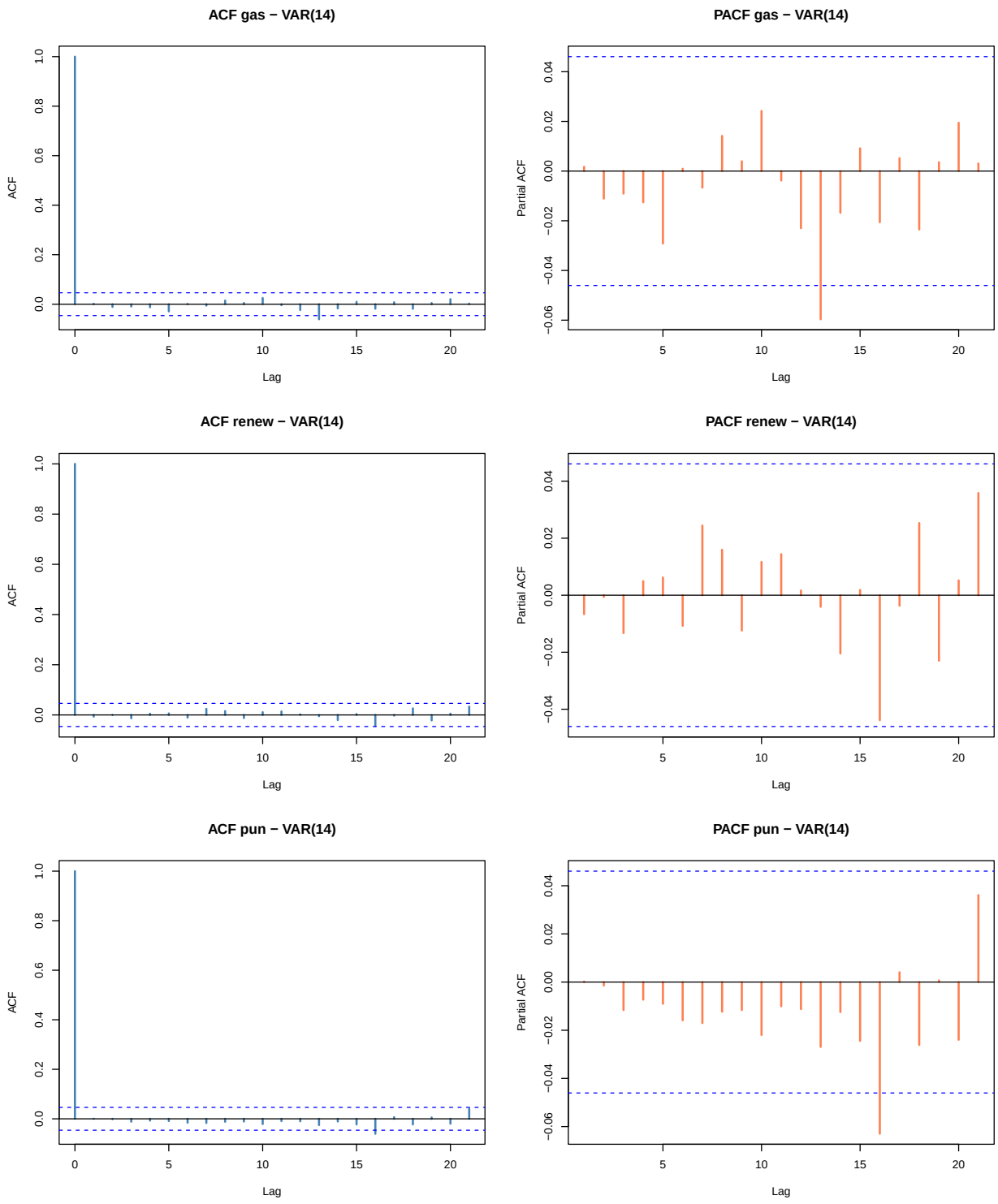

ACF renew − VAR(7)
ACF
Lag


PACF renew − VAR(7)
Partial ACF
Lag


ACF pun − VAR(7)
ACF
Lag


PACF pun − VAR(7)
Partial ACF
Lag


Residui VAR(14)
gas
renew
pun
Time
0
500
1000
1500

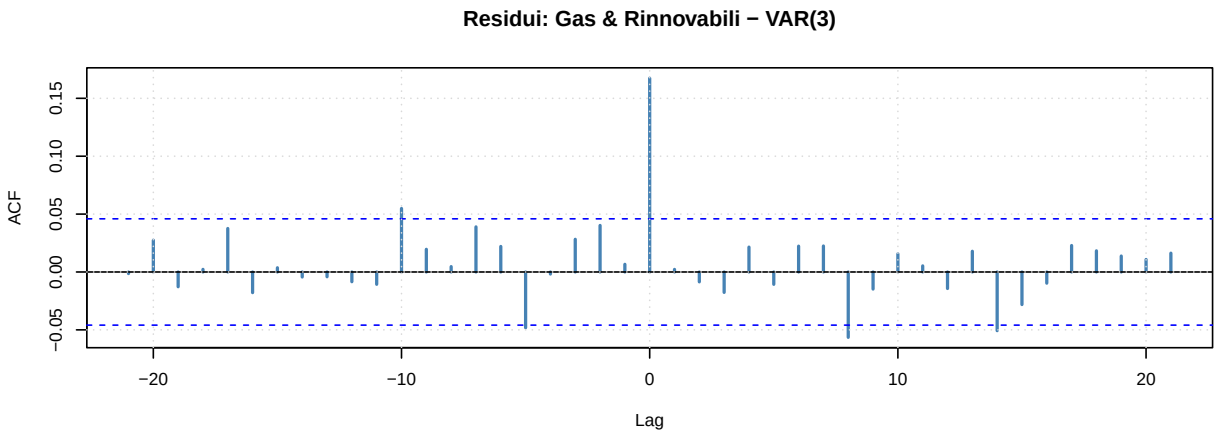


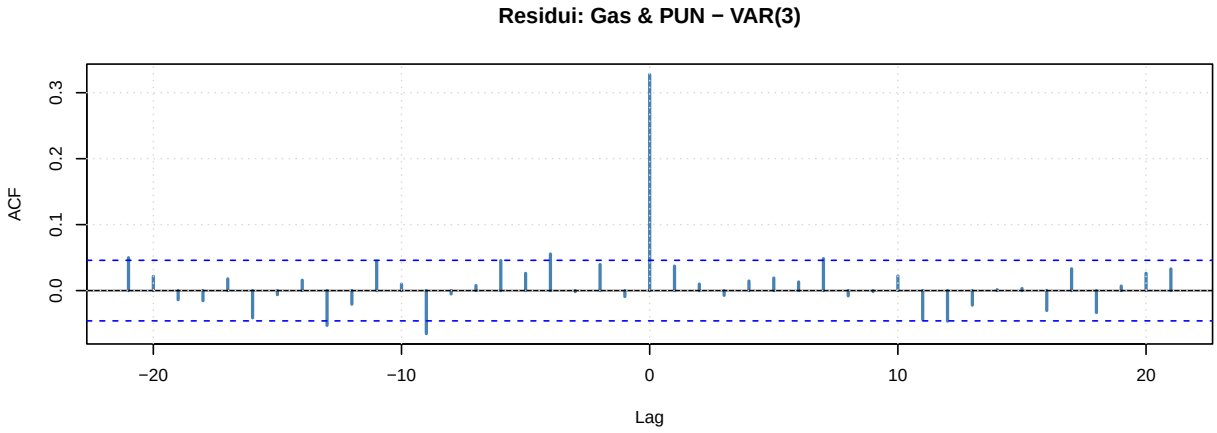
In tutti i modelli, i residui sono centrati attorno a zero, ma mostrano evidenze di eteroschedasticità. I residui del VAR(3) non si comportano come White Noise, con diversi coefficienti ancora significativi nelle funzioni ACF e PACF. Al contrario, i residui dei modelli VAR(7) e VAR(14) risultano più coerenti con le ipotesi di White Noise, senza differenze marcate tra le due specificazioni.

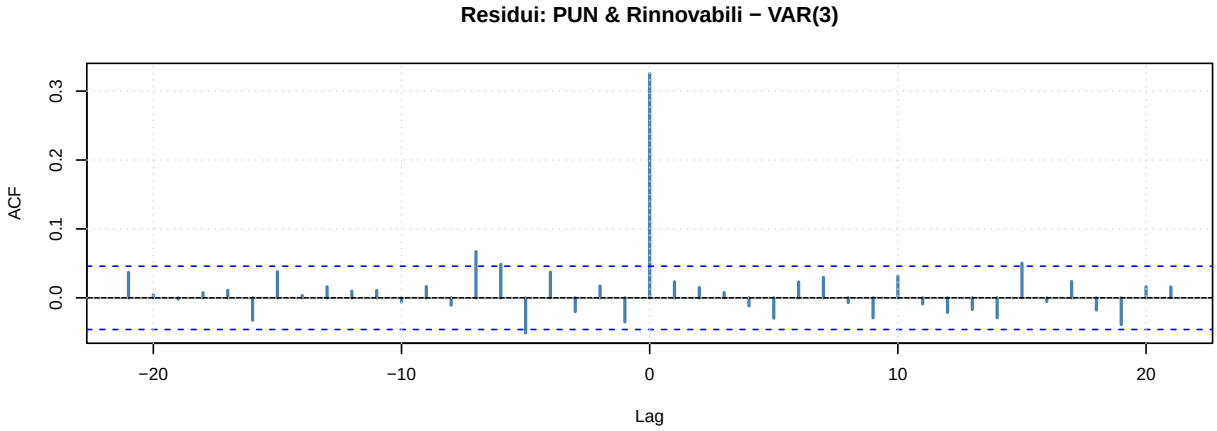
6.7.2 Cross-Correlazione dei Residui

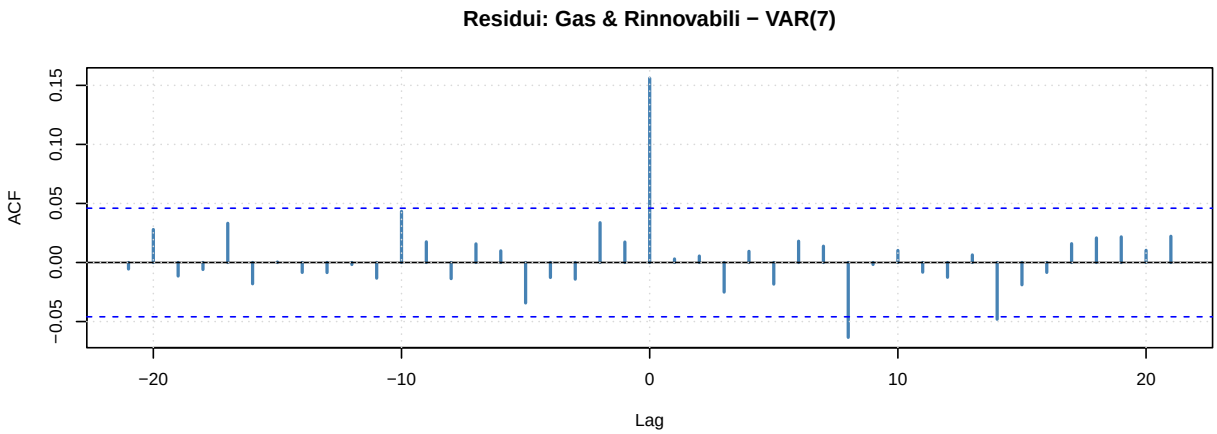
Verifichiamo l'indipendenza tra i residui delle diverse equazioni.

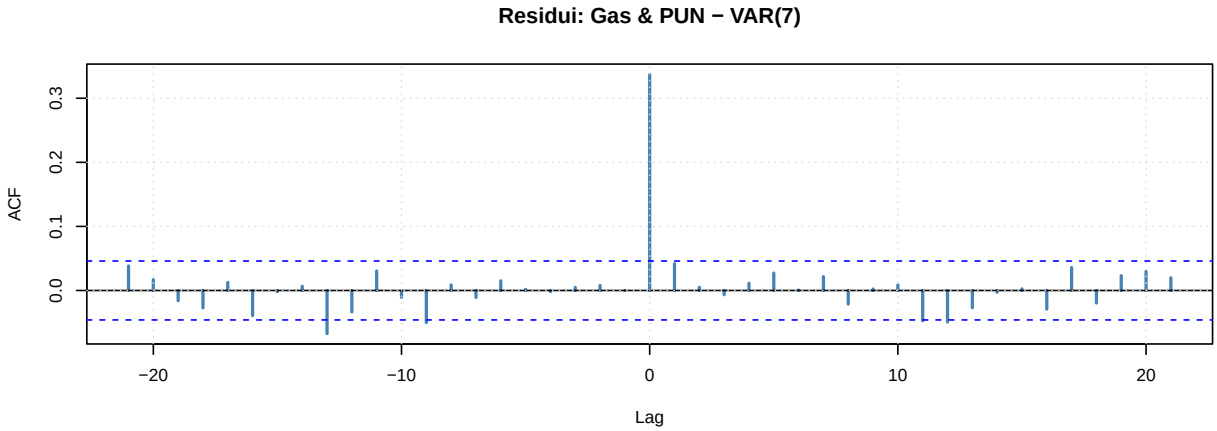
```
for(i in 1:3){
  par(mfrow = c(3, 1))
  ccf(res_var_list[[i]][, "gas"], res_var_list[[i]][, "renew"],
      lag.max = 21, main = paste("Residui: Gas & Rinnovabili -", var_names[i]),
      col = "steelblue", lwd = 2)
  grid(col = "gray85", lty = "dotted")
  ccf(res_var_list[[i]][, "gas"], res_var_list[[i]][, "pun"],
      lag.max = 21, main = paste("Residui: Gas & PUN -", var_names[i]),
      col = "steelblue", lwd = 2)
  grid(col = "gray85", lty = "dotted")
  ccf(res_var_list[[i]][, "pun"], res_var_list[[i]][, "renew"],
      lag.max = 21, main = paste("Residui: PUN & Rinnovabili -", var_names[i]),
      col = "steelblue", lwd = 2)
  grid(col = "gray85", lty = "dotted")
  par(mfrow = c(1, 1))
}
```

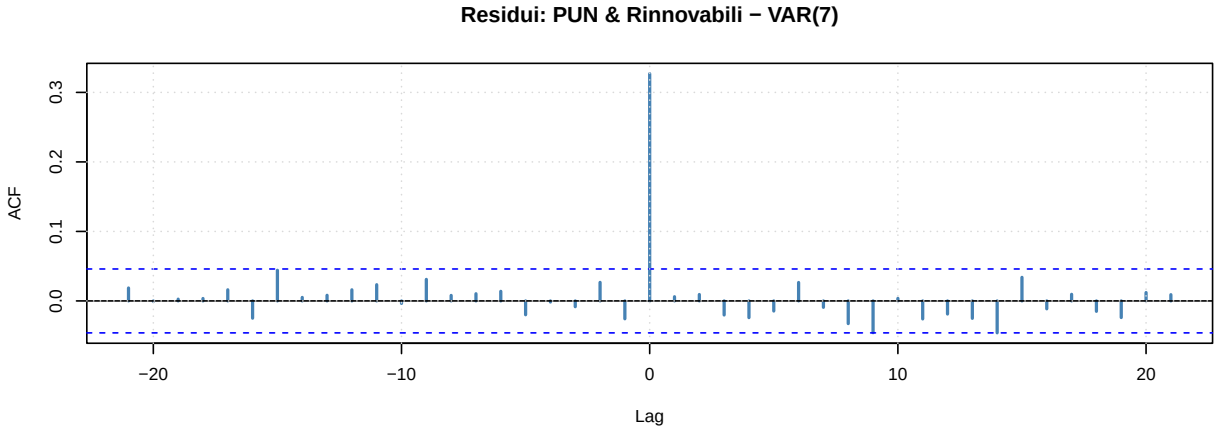

Residui: Gas & Rinnovabili − VAR(3)
ACF
0.15
0.10
0.05
0.00
−0.05
−20
−10
0
10
20
Lag

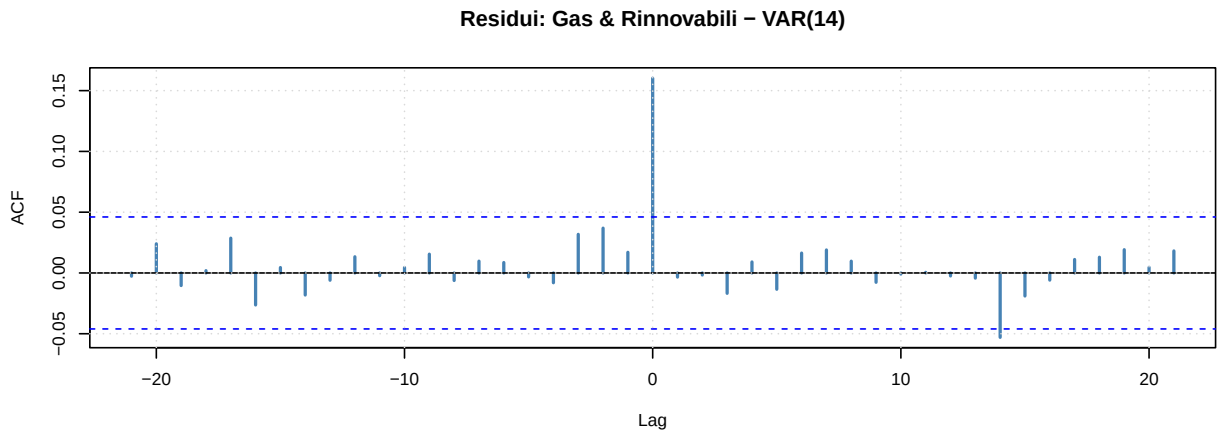

Residui: Gas & PUN − VAR(3)
ACF
0.3
0.2
0.1
0.0
−20
−10
0
10
20
Lag

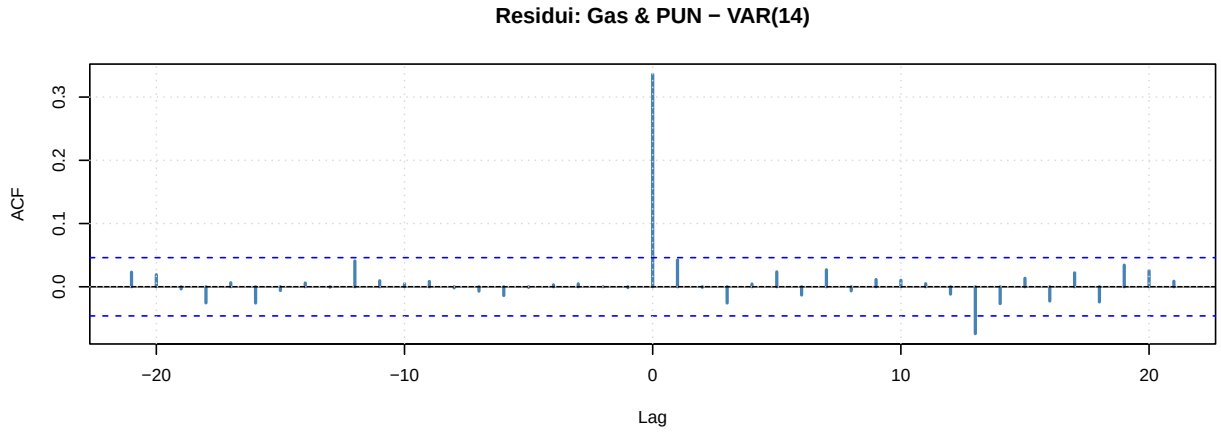

Residui: PUN & Rinnovabili − VAR(3)
ACF
0.3
0.2
0.1
0.0
−20
−10
0
10
20
Lag

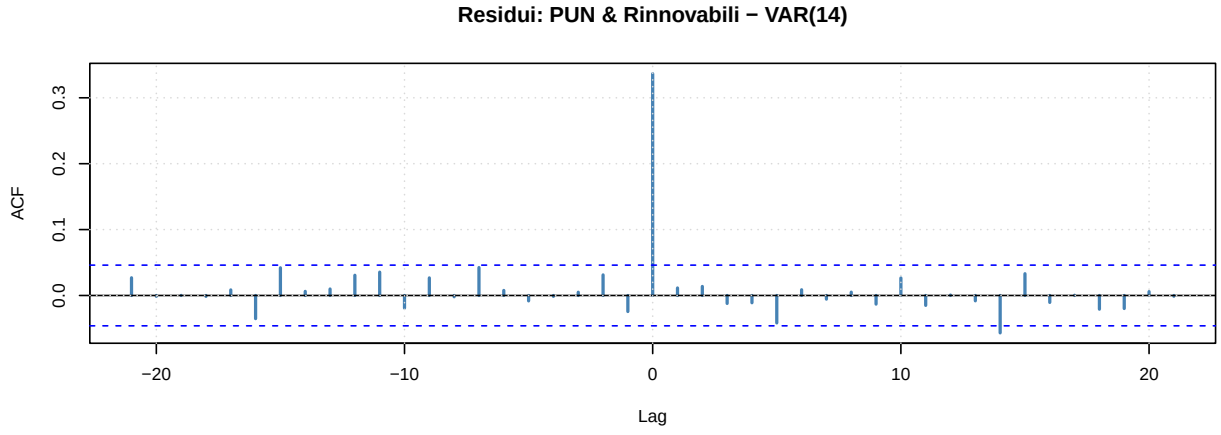

Residui: Gas & Rinnovabili − VAR(7)
ACF
0.15
0.10
0.05
0.00
−0.05
−20
−10
0
10
20
Lag


Residui: Gas & PUN − VAR(7)
ACF
0.3
0.2
0.1
0.0
−20
−10
0
10
20
Lag


Residui: PUN & Rinnovabili − VAR(7)
ACF
0.3
0.2
0.1
0.0
−20
−10
0
10
20
Lag







Nel VAR(3) i residui non risultano completamente incorrelati, come indicano alcuni coefficienti significativi nelle funzioni di cross-correlazione. Al contrario, i residui dei modelli VAR(7) e VAR(14) appaiono sostanzialmente incorrelati.

6.7.3 Correlazione Contemporanea

```
for(i in 1:3){
  cat("\n", var_names[i], ":\n")
```

```
  print(round(cor(res_var_list[[i]]), 3))
}
```

```
##
##  VAR(3) :
##         gas renew   pun
## gas   1.000 0.167 0.328
## renew 0.167 1.000 0.325
## pun   0.328 0.325 1.000
##
##  VAR(7) :
##         gas renew   pun
## gas   1.000 0.156 0.337
## renew 0.156 1.000 0.327
## pun   0.337 0.327 1.000
##
##  VAR(14) :
##         gas renew   pun
## gas   1.000 0.160 0.336
## renew 0.160 1.000 0.337
## pun   0.336 0.337 1.000
```

Le tre matrici di correlazione dei residui sono molto simili tra loro, con valori di correlazione tra le variabili relativamente contenuti.

6.8 Test Diagnostici

6.8.1 Test Univariati di Ljung-Box

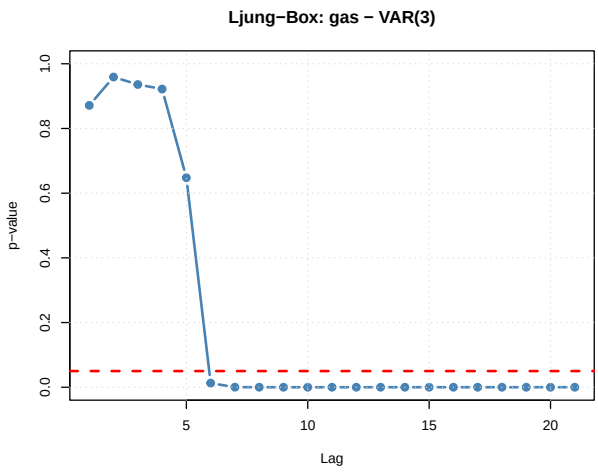
Verifichiamo l'assenza di autocorrelazione seriale nei residui di ciascuna equazione, sia sui residui che sui residui al quadrato (per testare eteroschedasticità).

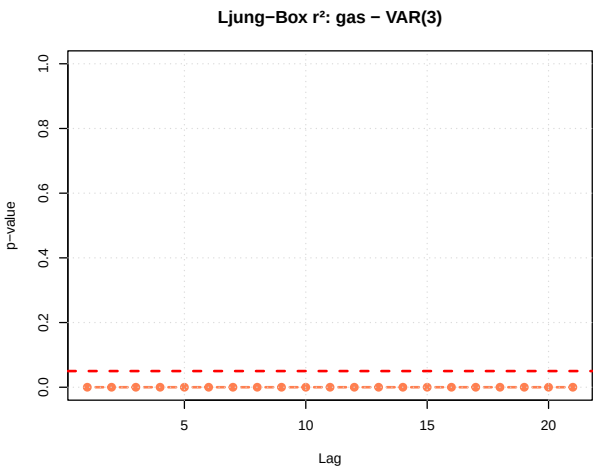
```
for(i in 1:3){
  cat("\n", var_names[i], ":\n")
  pval <- matrix(NA, 2, 3)
  colnames(pval) <- colnames(res_var_list[[i]])
  rownames(pval) <- c("Ljung-Box r", "Ljung-Box r^2")

  for(j in 1:NCOL(res_var_list[[i]])){
    pval[1, j] <- Box.test(res_var_list[[i]][, j], lag = 21, type = "Lj")$p.value
    pval[2, j] <- Box.test(res_var_list[[i]][, j]^2, lag = 21, type = "Lj")$p.value
  }
  print(round(pval, 4))
}
```

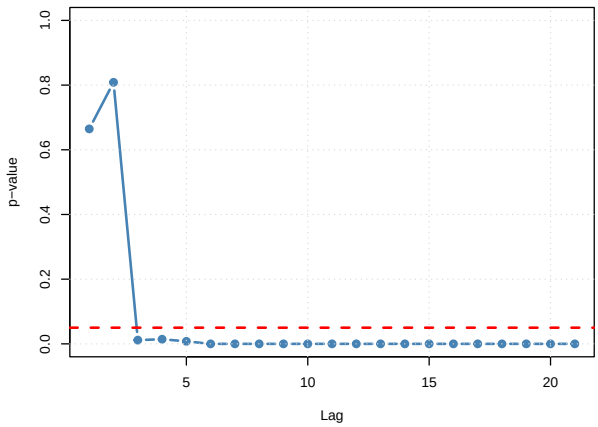
```
##
##  VAR(3) :
##                gas renew pun
## Ljung-Box r      0     0   0
## Ljung-Box r^2    0     0   0
```

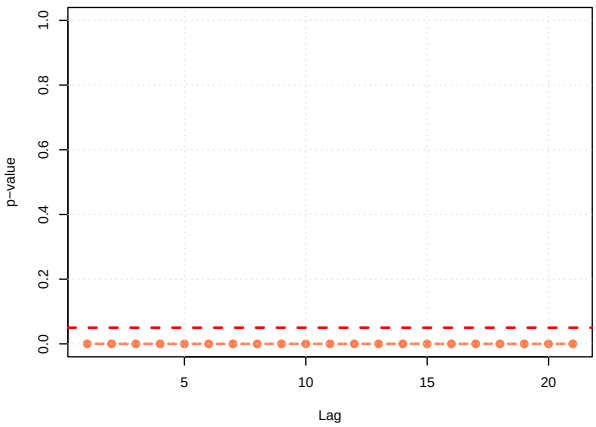
```
##
##  VAR(7) :
##                   gas renew pun
## Ljung-Box r   0.0149 0.132   0
## Ljung-Box r^2 0.0000 0.000   0
##
##  VAR(14) :
##                   gas  renew    pun
## Ljung-Box r   0.8495 0.9472 0.7184
## Ljung-Box r^2 0.0000 0.0000 0.0000
```

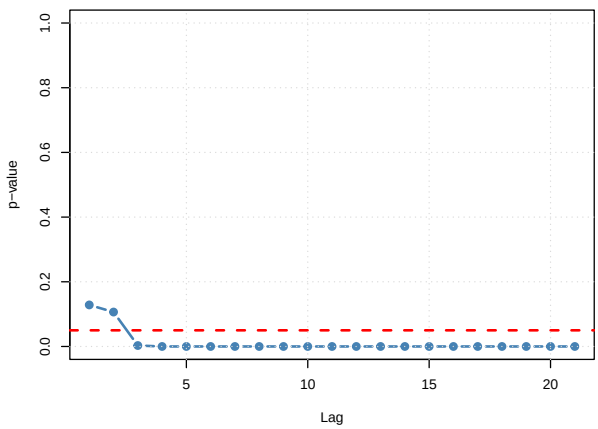
```r
for(i in 1:3){
  par(mfrow = c(3, 2))
  for(j in 1:3){
    pvals_r <- sapply(1:21, function(l)
      Box.test(res_var_list[[i]][, j], lag = l, type = "Ljung-Box")$p.value)
    plot(pvals_r, type = "b", ylim = c(0, 1), ylab = "p-value",
         xlab = "Lag", main = paste("Ljung-Box:", colnames(res_var_list[[i]])[j],
                                    "-", var_names[i]),
         pch = 19, col = "steelblue", lwd = 2)
    abline(h = 0.05, col = "red", lty = 2, lwd = 2)
    grid(col = "gray85", lty = "dotted")

    pvals_r2 <- sapply(1:21, function(l)
      Box.test(res_var_list[[i]][, j]^2, lag = l, type = "Ljung-Box")$p.value)
    plot(pvals_r2, type = "b", ylim = c(0, 1), ylab = "p-value",
         xlab = "Lag", main = paste("Ljung-Box r²:", colnames(res_var_list[[i]])[j],
                                    "-", var_names[i]),
         pch = 19, col = "coral", lwd = 2)
    abline(h = 0.05, col = "red", lty = 2, lwd = 2)
    grid(col = "gray85", lty = "dotted")
  }
  par(mfrow = c(1, 1))
}
```

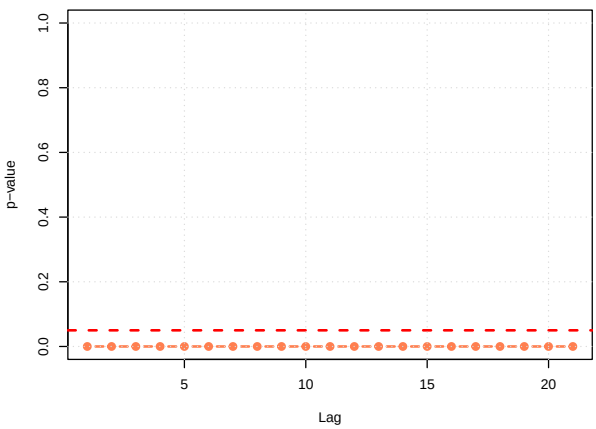

Ljung−Box: gas − VAR(3)
p−value
Lag

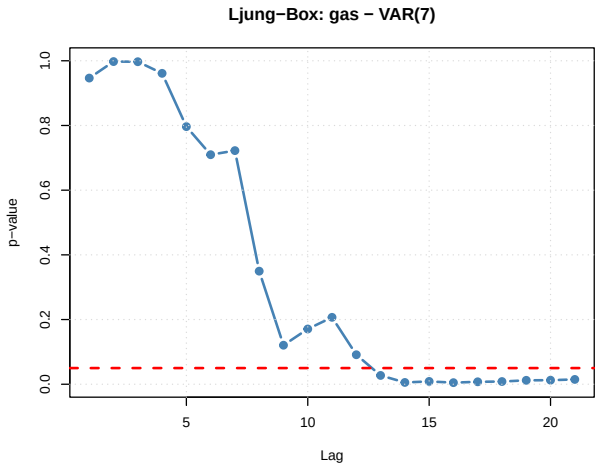

Ljung−Box r²: gas − VAR(3)
p−value
Lag

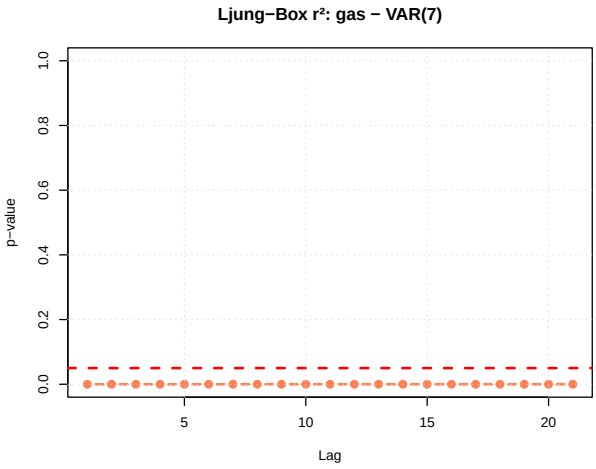

Ljung−Box: renew − VAR(3)
p−value
Lag

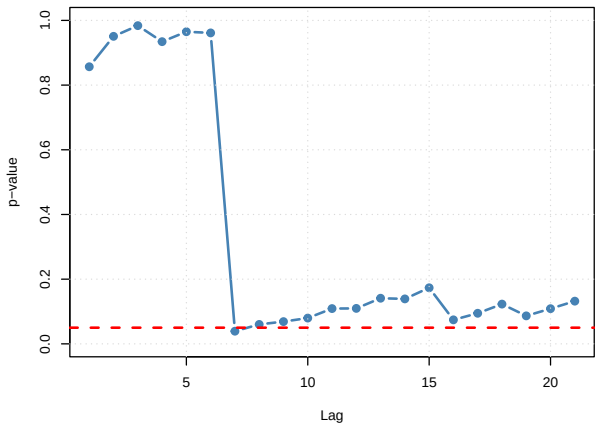

Ljung−Box r²: renew − VAR(3)
p−value
Lag

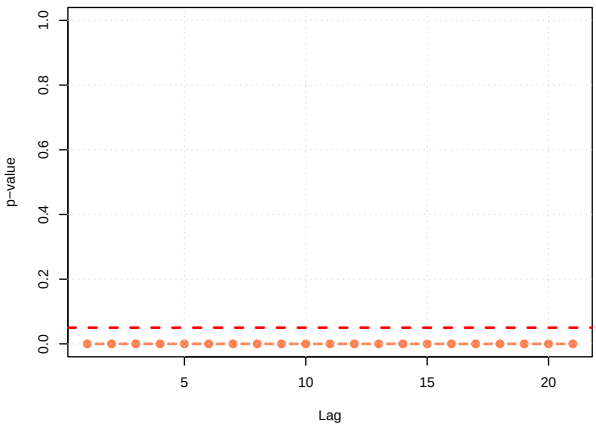

Ljung−Box: pun − VAR(3)
p−value
Lag

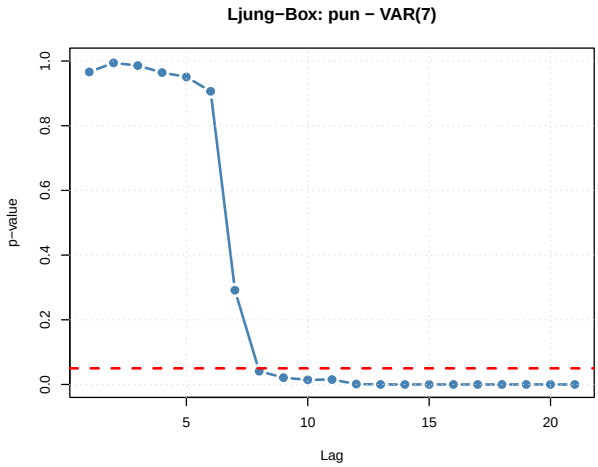

Ljung−Box r²: pun − VAR(3)
p−value
Lag

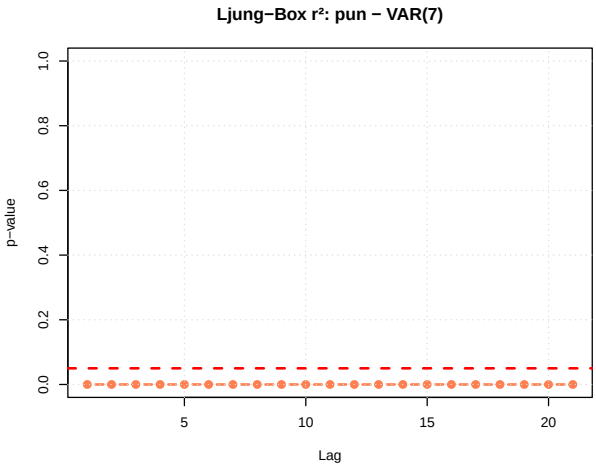

Ljung−Box: gas − VAR(7)
p−value
Lag

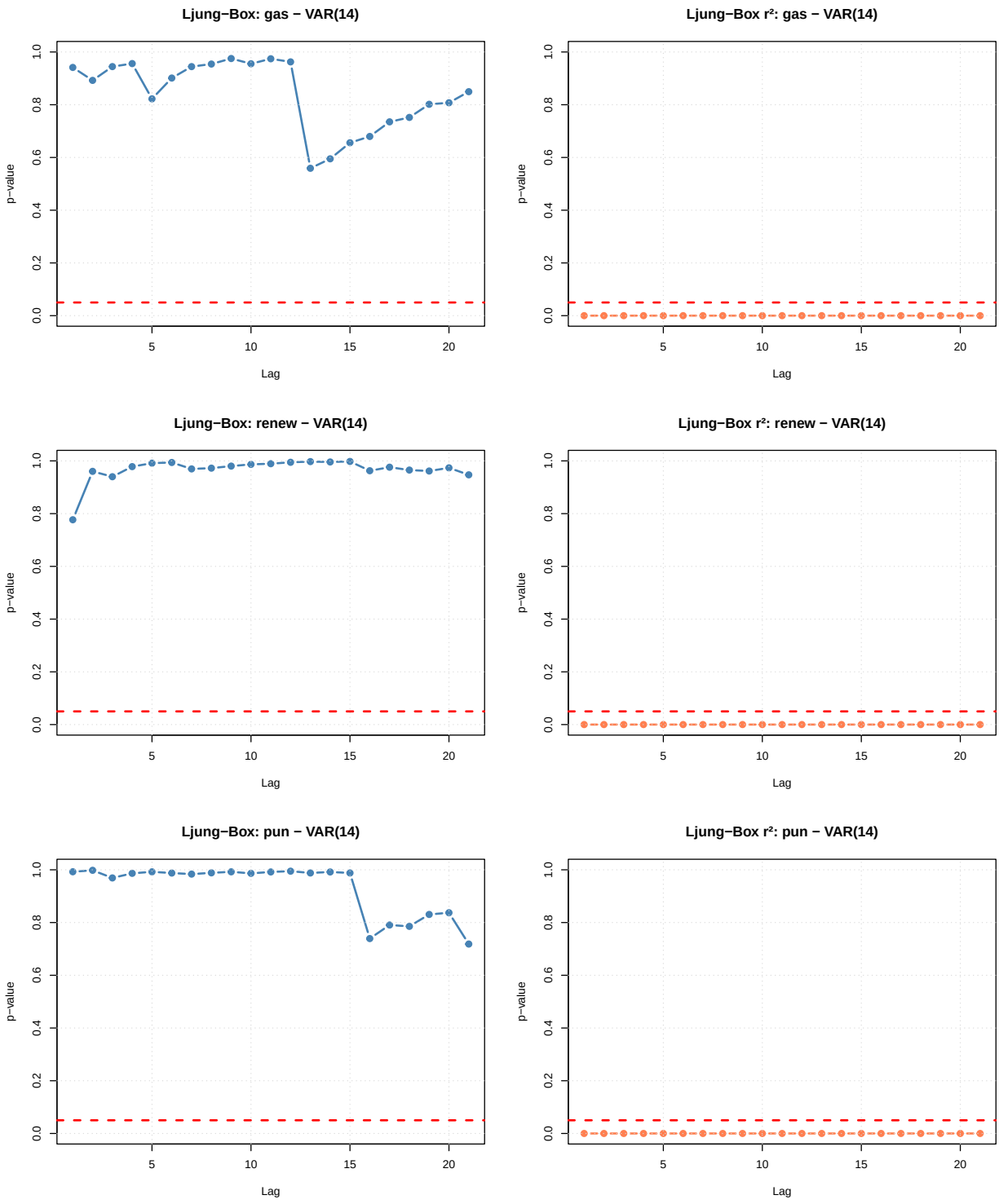

Ljung−Box r²: gas − VAR(7)
p−value
Lag


Ljung−Box: renew − VAR(7)
p−value
Lag


Ljung−Box r²: renew − VAR(7)
p−value
Lag


Ljung−Box: pun − VAR(7)
p−value
Lag


Ljung−Box r²: pun − VAR(7)
p−value
Lag



Il test di Ljung-Box sui residui conferma come il VAR(3) non colga tutta la struttura delle variabili, con residui che presentano autocorrelazione seriale. Anche il VAR(7) presenta autocorrelazione seriale residua per Gas e PUN, mentre il VAR(14) riesce a cogliere interamente la struttura di autocorrelazione delle serie. Il test sui residui al quadrato conferma la presenza di eteroschedasticità in tutti i modelli, come osservato precedentemente nell'analisi grafica.

6.8.2 Test di Stabilità

Verifichiamo che le radici del polinomio caratteristico siano inferiori a uno, condizione necessaria per la stazionarietà del processo VAR.

```
cat("VAR(3):\n"); print(roots(mod_var3_res))
```

```
## VAR(3):
```

```
## [1] 0.9078 0.5754 0.5754 0.5355 0.5355 0.5159 0.3987 0.3987 0.2697
```

```
cat("\nVAR(7):\n"); print(roots(mod_var7_res))
```

```
##
## VAR(7):
```

```
##  [1] 9.398e-01 7.604e-01 7.604e-01 7.556e-01 7.556e-01 7.509e-01 7.509e-01
##  [8] 7.509e-01 7.013e-01 7.013e-01 6.947e-01 6.947e-01 6.820e-01 6.820e-01
## [15] 6.660e-01 6.567e-01 6.567e-01 6.093e-01 6.093e-01 3.760e-01 1.977e-16
```

```
cat("\nVAR(14):\n"); print(roots(mod_var14_res))
```

```
##
## VAR(14):
```

```
##  [1] 0.8993 0.8993 0.8845 0.8845 0.8826 0.8826 0.8717 0.8717 0.8711 0.8711
## [11] 0.8582 0.8582 0.8403 0.8403 0.8392 0.8392 0.8298 0.8298 0.8292 0.8283
## [21] 0.8283 0.8257 0.8091 0.8091 0.8032 0.8032 0.7887 0.7887 0.7813 0.7531
## [31] 0.7531 0.7413 0.7413 0.7162 0.6546 0.6546 0.6499 0.6499 0.0000 0.0000
## [41] 0.0000 0.0000
```

Tutte le radici sono minori di 1, confermando la stabilità dei tre modelli stimati.

6.8.3 Test di Normalità Multivariata

```
cat("VAR(3):\n"); print(normality.test(mod_var3_res))
```

```
## VAR(3):
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 36370, df = 6, p-value <2e-16
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 934, df = 3, p-value <2e-16
```

```
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 35436, df = 3, p-value <2e-16
```

```
## $jb.mul
## $jb.mul$JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 36370, df = 6, p-value <2e-16
##
##
## $jb.mul$Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 934, df = 3, p-value <2e-16
##
##
## $jb.mul$Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 35436, df = 3, p-value <2e-16
```

```
cat("\nVAR(7):\n"); print(normality.test(mod_var7_res))
```

```
##
## VAR(7):
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 35189, df = 6, p-value <2e-16
##
##
## $Skewness
##
##  Skewness only (multivariate)
```

```
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 984, df = 3, p-value <2e-16
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 34205, df = 3, p-value <2e-16
```

```
## $jb.mul
## $jb.mul$JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 35189, df = 6, p-value <2e-16
##
##
## $jb.mul$Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 984, df = 3, p-value <2e-16
##
##
## $jb.mul$Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 34205, df = 3, p-value <2e-16
```

```
cat("\nVAR(14):\n"); print(normality.test(mod_var14_res))
```

```
##
## VAR(14):
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 29821, df = 6, p-value <2e-16
##
##
```

```
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 850, df = 3, p-value <2e-16
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 28971, df = 3, p-value <2e-16
```

```
## $jb.mul
## $jb.mul$JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 29821, df = 6, p-value <2e-16
##
##
## $jb.mul$Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 850, df = 3, p-value <2e-16
##
##
## $jb.mul$Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 28971, df = 3, p-value <2e-16
```

Tutti i test rifiutano le ipotesi di normalità dei residui.

6.8.4 Test di Autocorrelazione Seriale

```
cat("VAR(3):\n"); print(serial.test(mod_var3_res, lags.pt = 21, type = "PT.asymptotic"))
```

6.8.4.1 Test Portmanteau

```
## VAR(3):
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 432, df = 162, p-value <2e-16

## $serial
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 432, df = 162, p-value <2e-16
```

```
cat("\nVAR(7):\n"); print(serial.test(mod_var7_res, lags.pt = 21, type = "PT.asymptotic"))
```

```
##
## VAR(7):

##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 252, df = 126, p-value = 2e-10

## $serial
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 252, df = 126, p-value = 2e-10
```

```
cat("\nVAR(14):\n"); print(serial.test(mod_var14_res, lags.pt = 21, type = "PT.asymptotic"))
```

```
##
## VAR(14):

##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 144, df = 63, p-value = 3e-08

## $serial
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 144, df = 63, p-value = 3e-08
```

```
cat("VAR(3):\n"); print(serial.test(mod_var3_res, lags.bg = 21, type = "BG"))
```

6.8.4.2 Test Breusch-Godfrey

```
## VAR(3):

##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 493, df = 189, p-value <2e-16

## $serial
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 493, df = 189, p-value <2e-16
```

```
cat("\nVAR(7):\n"); print(serial.test(mod_var7_res, lags.bg = 21, type = "BG"))
```

```
##
## VAR(7):

##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 312, df = 189, p-value = 4e-08

## $serial
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 312, df = 189, p-value = 4e-08
```

```
cat("\nVAR(14):\n"); print(serial.test(mod_var14_res, lags.bg = 21, type = "BG"))
```

```
##
## VAR(14):

##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 219, df = 189, p-value = 0.07

## $serial
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 219, df = 189, p-value = 0.07
```

I due test concordano nell'indicare autocorrelazione residua nei VAR(3) e VAR(7), mentre il test BG mostra un p-value borderline (0.07) per il VAR(14), suggerendo assenza di autocorrelazione al 5%.

6.8.5 Test ARCH

Verifichiamo la presenza di eteroschedasticità condizionale (effetti ARCH) nei residui multivariati.

```
cat("VAR(3):\n"); print(arch.test(mod_var3_res, lags.multi = 21))
```

```
## VAR(3):
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 2750, df = 756, p-value <2e-16
```

```
## $arch.mul
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object mod_var3_res
## Chi-squared = 2750, df = 756, p-value <2e-16
```

```
cat("\nVAR(7):\n"); print(arch.test(mod_var7_res, lags.multi = 21))
```

```
##
## VAR(7):
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 2641, df = 756, p-value <2e-16
```

```
## $arch.mul
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object mod_var7_res
## Chi-squared = 2641, df = 756, p-value <2e-16
```

```
cat("\nVAR(14):\n"); print(arch.test(mod_var14_res, lags.multi = 21))
```

```
##
## VAR(14):
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 2551, df = 756, p-value <2e-16
```

```
## $arch.mul
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object mod_var14_res
## Chi-squared = 2551, df = 756, p-value <2e-16
```

I test ARCH mostrano evidenza di eteroschedasticità condizionale, coerentemente con quanto osservato nei test Ljung-Box sui residui al quadrato.

6.9 Test di Causalità di Granger

La causalità di Granger verifica se i valori passati di una variabile aiutano a prevedere un'altra variabile, controllando per i valori passati di quest'ultima. Conduciamo sia test multivariati sia test bivariati pairwise.

```
vars <- c("gas", "pun", "renew")
p_vec <- c(3, 7, 14)

for(i in seq_along(var_models)){
  p_i <- p_vec[i]

  cat("\n====================================\n")
  cat("Modello:", var_names[i], "| lag =", p_i, "\n")
  cat("====================================\n")

  # Granger multivariato (VAR completo)
  cat("\n--- Granger Multivariato (VAR) ---\n")
  for(cause in vars){
    cat("\nCausa:", cause, "\n")
    print(causality(var_models[[i]], cause = cause)$Granger)
  }

  # Granger pairwise bivariato
  cat("\n--- Granger Pairwise Bivariato ---\n")
  for(cause in vars){
    for(effect in vars){
      if(cause != effect){
        cat("\n", cause, "$\rightarrow$", effect, "\n")
        print(grangertest(df_var[, cause], df_var[, effect], order = p_i))
      }
    }
  }
}
```

```
##
## ====================================
## Modello: VAR(3) | lag = 3
## ====================================
```

```
##
## --- Granger Multivariato (VAR) ---
##
## Causa: gas
##
##  Granger causality H0: gas do not Granger-cause renew pun
##
## data:  VAR object var_models[[i]]
## F-Test = 16, df1 = 6, df2 = 5436, p-value <2e-16
##
##
## Causa: pun
##
##  Granger causality H0: pun do not Granger-cause gas renew
##
## data:  VAR object var_models[[i]]
## F-Test = 1.2, df1 = 6, df2 = 5436, p-value = 0.3
##
##
## Causa: renew
##
##  Granger causality H0: renew do not Granger-cause gas pun
##
## data:  VAR object var_models[[i]]
## F-Test = 1.5, df1 = 6, df2 = 5436, p-value = 0.2
##
##
## --- Granger Pairwise Bivariato ---
##
##  gas $ightarrow$ pun
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:3) + Lags(df_var[, cause], 1:3)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:3)
##   Res.Df Df    F Pr(>F)
## 1   1815
## 2   1818 -3 70.2 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  gas $ightarrow$ renew
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:3) + Lags(df_var[, cause], 1:3)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:3)
##   Res.Df Df    F Pr(>F)
## 1   1815
## 2   1818 -3 1.51   0.21
```

```
##
##  pun $ightarrow$ gas
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:3) + Lags(df_var[, cause], 1:3)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:3)
##   Res.Df Df    F Pr(>F)
## 1   1815
## 2   1818 -3 1.49   0.22
##
##  pun $ightarrow$ renew
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:3) + Lags(df_var[, cause], 1:3)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:3)
##   Res.Df Df    F Pr(>F)
## 1   1815
## 2   1818 -3 1.16   0.32
##
##  renew $ightarrow$ gas
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:3) + Lags(df_var[, cause], 1:3)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:3)
##   Res.Df Df    F Pr(>F)
## 1   1815
## 2   1818 -3 1.59   0.19
##
##  renew $ightarrow$ pun
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:3) + Lags(df_var[, cause], 1:3)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:3)
##   Res.Df Df    F Pr(>F)
## 1   1815
## 2   1818 -3 2.51  0.057 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ====================================
## Modello: VAR(7) | lag = 7
## ====================================
##
## --- Granger Multivariato (VAR) ---
##
## Causa: gas
##
##  Granger causality H0: gas do not Granger-cause renew pun
```

```
##
## data:  VAR object var_models[[i]]
## F-Test = 8.9, df1 = 14, df2 = 5388, p-value <2e-16
##
##
## Causa: pun
##
##  Granger causality H0: pun do not Granger-cause gas renew
##
## data:  VAR object var_models[[i]]
## F-Test = 1.3, df1 = 14, df2 = 5388, p-value = 0.2
##
##
## Causa: renew
##
##  Granger causality H0: renew do not Granger-cause gas pun
##
## data:  VAR object var_models[[i]]
## F-Test = 1.2, df1 = 14, df2 = 5388, p-value = 0.3
##
##
## --- Granger Pairwise Bivariato ---
##
##  gas $ightarrow$ pun
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:7) + Lags(df_var[, cause], 1:7)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:7)
##   Res.Df Df    F Pr(>F)
## 1   1803
## 2   1810 -7 41.5 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  gas $ightarrow$ renew
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:7) + Lags(df_var[, cause], 1:7)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:7)
##   Res.Df Df   F Pr(>F)
## 1   1803
## 2   1810 -7 1.6   0.13
##
##  pun $ightarrow$ gas
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:7) + Lags(df_var[, cause], 1:7)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:7)
```

```
##   Res.Df Df    F Pr(>F)
## 1   1803
## 2   1810 -7 1.96  0.057 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  pun $ightarrow$ renew
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:7) + Lags(df_var[, cause], 1:7)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:7)
##   Res.Df Df    F Pr(>F)
## 1   1803
## 2   1810 -7 2.29  0.025 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  renew $ightarrow$ gas
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:7) + Lags(df_var[, cause], 1:7)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:7)
##   Res.Df Df    F Pr(>F)
## 1   1803
## 2   1810 -7 1.25   0.27
##
##  renew $ightarrow$ pun
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:7) + Lags(df_var[, cause], 1:7)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:7)
##   Res.Df Df    F Pr(>F)
## 1   1803
## 2   1810 -7 1.65   0.12
##
## ====================================
## Modello: VAR(14) | lag = 14
## ====================================
##
## --- Granger Multivariato (VAR) ---
##
## Causa: gas
##
##  Granger causality H0: gas do not Granger-cause renew pun
##
## data:  VAR object var_models[[i]]
## F-Test = 5.1, df1 = 28, df2 = 5304, p-value <2e-16
##
```

```
##
## Causa: pun
##
##  Granger causality H0: pun do not Granger-cause gas renew
##
## data:  VAR object var_models[[i]]
## F-Test = 1, df1 = 28, df2 = 5304, p-value = 0.4
##
##
## Causa: renew
##
##  Granger causality H0: renew do not Granger-cause gas pun
##
## data:  VAR object var_models[[i]]
## F-Test = 1.1, df1 = 28, df2 = 5304, p-value = 0.3
##
##
## --- Granger Pairwise Bivariato ---
##
##  gas $ightarrow$ pun
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:14) + Lags(df_var[, cause], 1:14)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:14)
##   Res.Df  Df    F Pr(>F)
## 1   1782
## 2   1796 -14 23.5 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  gas $ightarrow$ renew
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:14) + Lags(df_var[, cause], 1:14)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:14)
##   Res.Df  Df    F Pr(>F)
## 1   1782
## 2   1796 -14 0.93   0.52
##
##  pun $ightarrow$ gas
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:14) + Lags(df_var[, cause], 1:14)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:14)
##   Res.Df  Df    F Pr(>F)
## 1   1782
## 2   1796 -14 1.57   0.08 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  pun $ightarrow$ renew
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:14) + Lags(df_var[, cause], 1:14)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:14)
##   Res.Df  Df    F Pr(>F)
## 1   1782
## 2   1796 -14 1.04   0.41
##
##  renew $ightarrow$ gas
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:14) + Lags(df_var[, cause], 1:14)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:14)
##   Res.Df  Df    F Pr(>F)
## 1   1782
## 2   1796 -14 1.68  0.053 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  renew $ightarrow$ pun
## Granger causality test
##
## Model 1: df_var[, effect] ~ Lags(df_var[, effect], 1:14) + Lags(df_var[, cause], 1:14)
## Model 2: df_var[, effect] ~ Lags(df_var[, effect], 1:14)
##   Res.Df  Df    F Pr(>F)
## 1   1782
## 2   1796 -14 1.12   0.33
```

Nei test multivariati condotti sul sistema completo, solo il Gas risulta causare secondo Granger le altre variabili in modo robusto. I test pairwise bivariati mostrano invece relazioni aggiuntive: Gas influenza PUN in tutti i modelli, con una relazione bidirezionale in VAR(7) e VAR(14). Emergono inoltre risultati discordanti tra le altre coppie di variabili, con relazioni che variano al cambiare della specificazione del modello.

Questa apparente contraddizione riflette una differenza metodologica fondamentale: i test pairwise ignorano le interazioni multivariate e possono quindi identificare relazioni spurie o indirette che emergono solo quando si escludono le altre variabili dal condizionamento. Al contrario, i test multivariati forniscono una visione più affidabile della struttura causale del sistema perché considerano simultaneamente tutte le interdipendenze. La sensibilità dei risultati all'ordine del VAR (che diminuisce la potenza del test all'aumentare di p) e l'assunzione di residui White Noise rafforzano ulteriormente l'importanza di privilegiare l'interpretazione dei test multivariati.

6.10 Sintesi e Scelta del Modello

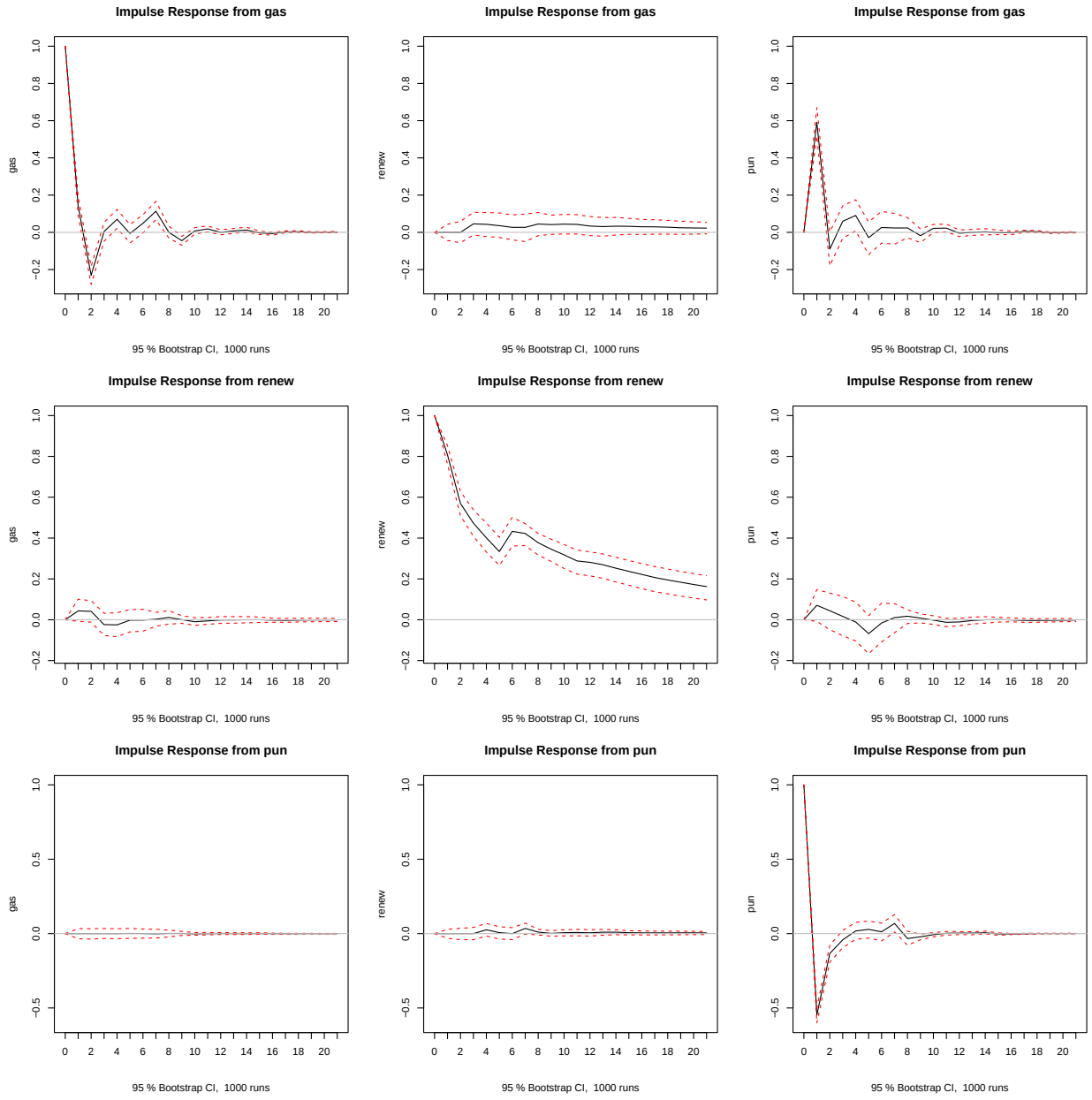
Sulla base dei criteri di informazione, dei test diagnostici e dell'analisi di causalità, il modello VAR(7) emerge come il compromesso ottimale. Esso offre maggiore parsimonia rispetto al VAR(14) con

un numero inferiore di parametri da stimare, diagnostica superiore al VAR(3) con residui che si avvicinano maggiormente alle ipotesi di White Noise, capacità di catturare efficacemente le relazioni dinamiche del sistema attraverso la finestra settimanale, e un BIC competitivo che conferma la presenza di un pattern stagionale settimanale persistente.

6.11 Analisi IRF

L'analisi della Funzione di Risposta all'Impulso (IRF) consente di studiare l'effetto dinamico di uno shock su una variabile del sistema sulle altre variabili nel tempo, permettendo di interpretare non solo la direzione e la significatività delle relazioni, ma anche l'ampiezza e la persistenza degli effetti.

```
irf_std <- irf(mod_var7_res,
               n.ahead = 21,
               ortho = FALSE,
               boot = TRUE,
               runs = 1000)
par(mfrow = c(3, 3))
plot(irf_std, plot.type = "single")
```

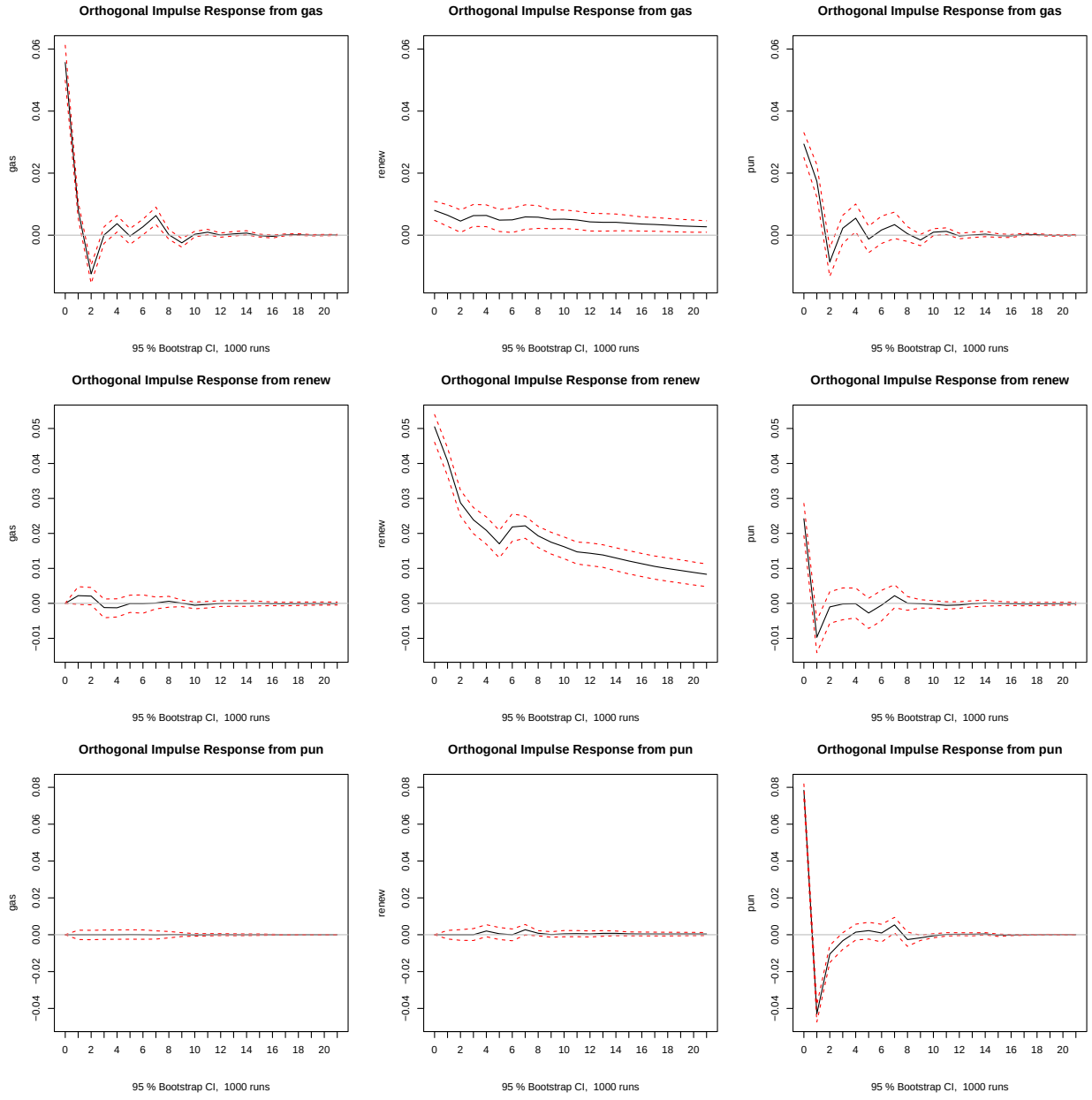



```
par(mfrow = c(1, 1))
```

Dall'analisi delle IRF standard emerge che ciascuna variabile risponde principalmente ai propri shock. Il Gas mostra effetti autoregressivi fino a 3 ritardi, le Rinnovabili rispondono in modo persistente esclusivamente a shock su sé stesse, mentre il PUN reagisce sia ai propri shock sia a quelli del Gas, con effetti fino a 3 ritardi.

```
irf_ortho <- irf(mod_var7_res,
                 impulse = c("gas", "renew", "pun"),
                 response = c("gas", "renew", "pun"),
                 n.ahead = 21,
                 ortho = TRUE,
                 boot = TRUE,
```

```
                  runs = 1000)
par(mfrow = c(3, 3))
plot(irf_ortho, plot.type = "single")
```




```
par(mfrow = c(1, 1))
```

L'ortogonalizzazione delle IRF mediante decomposizione di Cholesky produce risultati più chiari e interpretabili. I risultati sul Gas rimangono invariati, confermandone il ruolo di variabile principalmente esogena. Per le Rinnovabili emergono invece effetti più definiti: esse mostrano una risposta persistente a uno shock sul Gas, coerente con la programmazione della produzione rinnovabile che può reagire ai segnali di prezzo del mercato fossile attraverso decisioni di dispacciamento. Il PUN reagisce fino a due ritardi a uno shock delle Rinnovabili, effetto non evidente nelle IRF non

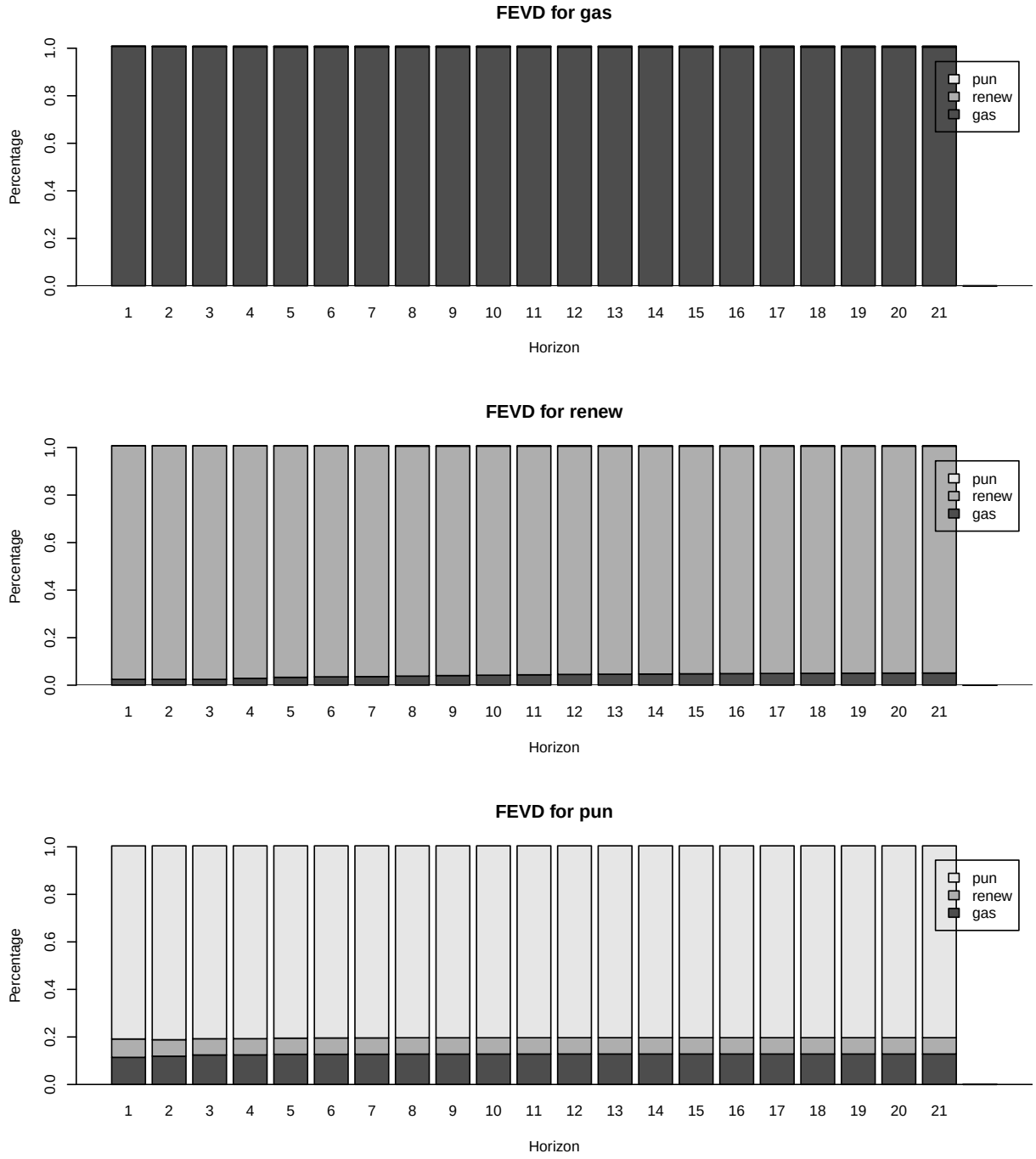
ortogonalizzate.

La differenza tra le due specificazioni deriva dall'ortogonalizzazione di Cholesky, che trasforma i residui correlati del VAR in shock decorrelati e interpretabili come innovazioni "pure" di ciascuna variabile. Questo processo isola l'impatto diretto di uno shock da quello indiretto dovuto alla correlazione contemporanea tra residui, facendo emergere relazioni di causalità e la loro persistenza che altrimenti rimarrebbero confuse nella struttura di correlazione del sistema. L'ordinamento delle variabili (Gas, Rinnovabili, PUN) diventa cruciale in questa trasformazione, riflettendo l'assunzione che il Gas non reagisca istantaneamente alle altre variabili, le Rinnovabili possano essere influenzate contemporaneamente dal Gas ma non dal PUN, e il PUN reagisca istantaneamente a entrambe.

6.12 Analisi FEVD

L'analisi della FEVD (Forecast Error Variance Decomposition) quantifica quanto ciascun shock contribuisce alla variabilità futura di ciascuna variabile del sistema.

```
fevd_result <- fevd(mod_var7_res, n.ahead = 21)
plot(fevd_result)
```



I risultati confermano quanto osservato finora: il Gas è influenzato quasi esclusivamente dai propri shock (oltre il 95% della varianza), le Rinnovabili dipendono principalmente da sé stesse con un modesto contributo del Gas (circa 5-10%), mentre il PUN è anch'esso guidato principalmente dalla propria dinamica, con effetti minori provenienti da Gas (circa 10-15%) e Rinnovabili (circa 5%).

Dall'analisi del modello VAR emerge come il sistema catturi efficacemente la struttura trivariata della serie, confermando quanto emerso dall'analisi di cointegrazione e dai modelli a funzioni di trasferimento. Il Gas si conferma come variabile prevalentemente esogena che guida il sistema, le

Rinnovabili mostrano una leggera dipendenza dal Gas ma mantengono una dinamica largamente autonoma determinata da fattori climatici e di capacità, mentre il PUN risulta la variabile più endogena, influenzata da entrambe le altre variabili del sistema. La decomposizione della varianza conferma questa gerarchia causale, mostrando come la maggior parte della variabilità di ciascuna variabile sia spiegata dai propri shock, con il Gas che mantiene un ruolo di driver esterno e il PUN che assorbe gli impulsi provenienti da tutte le componenti del mercato energetico.

7 Confronto Predittivo dei Modelli

In questa sezione confrontiamo la capacità predittiva out-of-sample di tutti i modelli stimati attraverso un'analisi di rolling forecast. Questo approccio permette di valutare le performance dei modelli in condizioni realistiche, simulando previsioni ripetute su finestre temporali mobili.

7.1 Setup e Parametri

Definiamo i parametri principali per l'analisi di rolling forecast. L'orizzonte di previsione è fissato a 14 giorni, la dimensione della finestra di training a 365 giorni, e lo step tra iterazioni a 5 giorni. Questa configurazione genera previsioni multi-step mantenendo una finestra di training sufficientemente ampia per catturare la dinamica del sistema, senza appesantire eccessivamente il carico computazionale.

L'uso di una finestra di training pari a un anno permette di isolare le osservazioni del 2022, periodo in cui le variabili Gas e PUN hanno subito uno shock improvviso, evitando che l'andamento anomalo influenzi eccessivamente le stime dei modelli nelle finestre temporali più recenti. Lo step di 5 giorni tra iterazioni bilancia la granularità dell'analisi con la sostenibilità computazionale della procedura.

```
h_max <- 14
train_size <- 365
step_size <- 5

target <- log(pun_ts)
n_total <- length(target)

first_t <- train_size + 1
last_t <- n_total - h_max
n_iter <- floor((last_t - first_t + 1) / step_size)
```

7.2 Rolling Forecast Loop

Implementiamo il rolling forecast per ciascuno dei cinque modelli considerati: ARIMA (modello SARIMA univariato), TFM_Gas (Transfer Function Model con Gas come input), TFM_Renew (Transfer Function Model con Rinnovabili come input), VECM (Vector Error Correction Model tra Gas e PUN con r=1), e VAR (VAR(7) ristretto su serie destagionalizzate).

Per ciascuna iterazione stimiamo il modello sulla finestra di training corrente, generiamo previsioni per i successivi 14 giorni, e calcoliamo gli errori di previsione rispetto ai valori osservati.

Siccome ogni modello è stimato su una trasformazione diversa delle serie (destagionalizzate, differenziate, ...), si è resa necessaria una attenta operazione di trasformazione inversa per rendere confrontabili le previsioni dei vari modelli.

```r
models <- c("ARIMA", "TFM_Gas", "TFM_Renew", "VECM", "VAR")
n_models <- length(models)

residuals_list <- vector("list", n_models)
names(residuals_list) <- models

for(m in models) {
  residuals_list[[m]] <- matrix(NA, nrow = n_iter, ncol = h_max)
}

# Rolling Forecast Loop
for(ii in 1:n_iter) {

  t_end <- first_t + (ii - 1) * step_size - 1
  t_start <- t_end - train_size + 1

  # Training data
  train_idx <- t_start:t_end
  test_idx <- (t_end + 1):(t_end + h_max)

  # Serie in log
  l_pun_train <- as.numeric(log(pun_ts)[train_idx])
  l_gas_train <- as.numeric(log(gas_ts)[train_idx])
  l_renew_train <- as.numeric(log(renew_ts)[train_idx])

  # True values per test
  y_true <- as.numeric(log(pun_ts)[test_idx])

  # ============================================================================
  # MODELLO 1: ARIMA UNIVARIATO
  # ============================================================================
  tryCatch({
    mod_arima <- Arima(ts(l_pun_train, frequency = 7),
                       order = c(0, 1, 2),
                       seasonal = list(order = c(1, 0, 1), period = 7))

    fc_arima <- forecast(mod_arima, h = h_max)
    residuals_list[["ARIMA"]][ii, ] <- y_true - as.numeric(fc_arima$mean)

  }, error = function(e) {})

  # ============================================================================
  # MODELLO 2: TFM GAS $\rightarrow$ PUN
  # ============================================================================
  tryCatch({
    # Destagionalizza PUN
    pun_ts_train <- ts(l_pun_train, frequency = 7)
```

```r
pun_stl <- stl(pun_ts_train, s.window = "periodic")
pun_seasonal <- pun_stl$time.series[, "seasonal"]
pun_trend_remainder <- pun_stl$time.series[, "trend"] +
  pun_stl$time.series[, "remainder"]

# Destagionalizza Gas
gas_ts_train <- ts(l_gas_train, frequency = 7)
gas_stl <- stl(gas_ts_train, s.window = "periodic")
gas_deseason <- gas_stl$time.series[, "trend"] +
  gas_stl$time.series[, "remainder"]

# Stima ARIMA su Gas destagionalizzato per forecast
mod_gas_arima <- auto.arima(gas_deseason, seasonal = FALSE,
                            max.p = 3, max.q = 3)
fc_gas_deseason <- forecast(mod_gas_arima, h = h_max)
gas_deseason_future <- as.numeric(fc_gas_deseason$mean)

# Costruisco regressori TFM con lag
n_train <- length(pun_trend_remainder)

X_train <- cbind(
  gas_t0 = gas_deseason[2:n_train],
  gas_t1 = gas_deseason[1:(n_train-1)]
)
y_train_tfm <- pun_trend_remainder[2:n_train]

# Future regressors
gas_extended <- c(tail(gas_deseason, 1), gas_deseason_future)
X_future <- cbind(
  gas_t0 = gas_deseason_future,
  gas_t1 = gas_extended[1:h_max]
)

# Stima TFM come ARIMA con xreg
mod_tfm_gas <- auto.arima(y_train_tfm, xreg = X_train,
                          seasonal = FALSE,
                          max.p = 2, max.q = 2)

# Forecast
fc_tfm <- forecast(mod_tfm_gas, h = h_max, xreg = X_future)

# Re-stagionalizza usando pattern stagionale del train
pun_seasonal_pattern <- tail(pun_seasonal, 7)
pun_seasonal_future <- rep(pun_seasonal_pattern, length.out = h_max)

fc_tfm_final <- as.numeric(fc_tfm$mean) + pun_seasonal_future
```

```
    residuals_list[["TFM_Gas"]][ii, ] <- y_true - fc_tfm_final

  }, error = function(e) {})

  # ============================================================================
  # MODELLO 3: TFM RENEW $\rightarrow$ PUN
  # ============================================================================
  tryCatch({
    # Destagionalizza Renew
    renew_ts_train <- ts(l_renew_train, frequency = 7)
    renew_stl <- stl(renew_ts_train, s.window = "periodic")
    renew_deseason <- renew_stl$time.series[, "trend"] +
      renew_stl$time.series[, "remainder"]

    # Forecast Renew destagionalizzato
    mod_renew_arima <- auto.arima(renew_deseason, seasonal = FALSE,
                                  max.p = 3, max.q = 3)
    fc_renew_deseason <- forecast(mod_renew_arima, h = h_max)
    renew_deseason_future <- as.numeric(fc_renew_deseason$mean)

    # Costruisco regressori TFM
    n_train <- length(pun_trend_remainder)

    X_train_r <- cbind(
      renew_t0 = renew_deseason[2:n_train],
      renew_t1 = renew_deseason[1:(n_train-1)]
    )
    y_train_tfm_r <- pun_trend_remainder[2:n_train]

    renew_extended <- c(tail(renew_deseason, 1), renew_deseason_future)
    X_future_r <- cbind(
      renew_t0 = renew_deseason_future,
      renew_t1 = renew_extended[1:h_max]
    )

    # Stima e forecast
    mod_tfm_renew <- auto.arima(y_train_tfm_r, xreg = X_train_r,
                                seasonal = FALSE,
                                max.p = 2, max.q = 2)
    fc_tfm_r <- forecast(mod_tfm_renew, h = h_max, xreg = X_future_r)

    # Re-stagionalizza
    fc_tfm_renew_final <- as.numeric(fc_tfm_r$mean) + pun_seasonal_future

    residuals_list[["TFM_Renew"]][ii, ] <- y_true - fc_tfm_renew_final

  }, error = function(e) {})
```

```r
# ============================================================================
# MODELLO 4: VECM (r=1 fisso)
# ============================================================================
tryCatch({
  # Dataset per cointegrazione
  data_coint <- ts(cbind(l_pun_train, l_gas_train), frequency = 7)
  colnames(data_coint) <- c("l_PUN", "l_Gas")

  # Test Johansen
  vecm_test <- ca.jo(data_coint, type = "trace", ecdet = "const",
                     K = 14, spec = "longrun")

  # Converti a VAR con r=1 fisso
  vecm_var <- vec2var(vecm_test, r = 1)

  # Previsione
  fc_vecm <- predict(vecm_var, n.ahead = h_max)
  fc_vecm_pun <- fc_vecm$fcst$l_PUN[, "fcst"]

  residuals_list[["VECM"]][ii, ] <- y_true - fc_vecm_pun

}, error = function(e) {})

# ============================================================================
# MODELLO 5: VAR(7) RESTRICTED
# ============================================================================
tryCatch({
  # Differenze prima
  gas_diff <- diff(l_gas_train)
  pun_diff <- diff(l_pun_train)
  renew_log <- l_renew_train[-1]  # allinea

  # Destagionalizza sul train
  gas_diff_ts <- ts(gas_diff, frequency = 7)
  pun_diff_ts <- ts(pun_diff, frequency = 7)
  renew_ts_var <- ts(renew_log, frequency = 7)

  stl_gas_train <- stl(gas_diff_ts, s.window = "periodic")
  stl_pun_train <- stl(pun_diff_ts, s.window = "periodic")
  stl_renew_train <- stl(renew_ts_var, s.window = "periodic")

  gas_ds <- stl_gas_train$time.series[, "trend"] +
    stl_gas_train$time.series[, "remainder"]
  pun_ds <- stl_pun_train$time.series[, "trend"] +
    stl_pun_train$time.series[, "remainder"]
  renew_ds <- stl_renew_train$time.series[, "trend"] +
    stl_renew_train$time.series[, "remainder"]
```

```
    # VAR su dati destagionalizzati
    df_var <- ts(cbind(gas_ds, pun_ds, renew_ds), frequency = 7)
    colnames(df_var) <- c("gas", "pun", "renew")

    mod_var <- vars::VAR(df_var, p = 7, type = "const")
    mod_var_res <- restrict(mod_var, method = "ser", thresh = 1.96)

    # Forecast
    fc_var <- predict(mod_var_res, n.ahead = h_max)
    fc_pun_ds <- fc_var$fcst$pun[, "fcst"]

    # Re-stagionalizza usando pattern del train
    pun_seasonal_pattern_var <- tail(stl_pun_train$time.series[, "seasonal"], 7)
    pun_seasonal_future_var <- rep(pun_seasonal_pattern_var, length.out = h_max)

    fc_pun_reseason <- fc_pun_ds + pun_seasonal_future_var

    # Back-transform da diff a levels
    last_log_pun <- tail(l_pun_train, 1)
    fc_pun_levels <- cumsum(c(last_log_pun, fc_pun_reseason))[-1]

    residuals_list[["VAR"]][ii, ] <- y_true - fc_pun_levels

  }, error = function(e) {})

}
```

Il rolling forecast è stato completato su 289 iterazioni. Eventuali fallimenti nella stima sono registrati come valori mancanti e verranno esclusi dal calcolo delle metriche. Tali fallimenti nella stima sono imputabili alla copertura di finestre temporali caratterizzate da forte volatilità, che rende le stime instabili.

7.3 Metriche di Performance

Calcoliamo le metriche Mean Absolute Error (MAE) e Root Mean Squared Error (RMSE) per ciascun orizzonte di previsione (h=1 a h=14) e in media.

```
lag_names_mae <- paste0("MAE_h", 1:14)
lag_names_rmse <- paste0("RMSE_h", 1:14)

mae_summary <- data.frame(matrix(NA, nrow = n_models, ncol = 17))
colnames(mae_summary) <- c("Model", lag_names_mae, "MAE_Avg", "N_valid")
mae_summary$Model <- models

rmse_summary <- data.frame(matrix(NA, nrow = n_models, ncol = 17))
colnames(rmse_summary) <- c("Model", lag_names_rmse, "RMSE_Avg", "N_valid")
rmse_summary$Model <- models
```

```
# Loop di Calcolo
for(i in 1:n_models) {
  model_name <- models[i]
  res_mat <- residuals_list[[model_name]]

  # Rimuovi righe con NA
  valid_rows <- complete.cases(res_mat)
  res_clean <- res_mat[valid_rows, , drop = FALSE]

  n_valid <- nrow(res_clean)

  if(n_valid > 0) {
    # Calcolo MAE
    maes_lags <- colMeans(abs(res_clean[, 1:14]))
    mae_avg <- mean(abs(res_clean))

    mae_summary[i, 2:15] <- maes_lags
    mae_summary[i, "MAE_Avg"] <- mae_avg
    mae_summary[i, "N_valid"] <- n_valid

    # Calcolo RMSE
    rmses_lags <- sqrt(colMeans(res_clean[, 1:14]^2))
    rmse_avg <- sqrt(mean(res_clean^2))

    rmse_summary[i, 2:15] <- rmses_lags
    rmse_summary[i, "RMSE_Avg"] <- rmse_avg
    rmse_summary[i, "N_valid"] <- n_valid
  }
}
```

7.3.1 Risultati MAE

```
mae_transposed <- data.frame(
  Orizzonte = c(paste0("h", 1:14), "Media", "N_valid"),
  t(mae_summary[, -1])
)
colnames(mae_transposed)[-1] <- mae_summary$Model

knitr::kable(mae_transposed, digits = 4,
             caption = "Mean Absolute Error per orizzonte di previsione",
             booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position")
```

7.3.2 Risultati RMSE

```
rmse_transposed <- data.frame(
  Orizzonte = c(paste0("h", 1:14), "Media", "N_valid"),
```

Tabella 7: Mean Absolute Error per orizzonte di previsione

	Orizzonte	ARIMA	TFM_Gas	TFM_Renew	VECM	VAR
MAE_h1	h1	0.0725	0.0644	0.0739	0.0756	0.0711
MAE_h2	h2	0.0851	0.0795	0.0852	0.0873	0.0837
MAE_h3	h3	0.0946	0.0850	0.0944	0.0988	0.0916
MAE_h4	h4	0.0961	0.0908	0.0997	0.1000	0.0943
MAE_h5	h5	0.1061	0.0989	0.1074	0.1106	0.1047
MAE_h6	h6	0.1116	0.1070	0.1129	0.1198	0.1164
MAE_h7	h7	0.1163	0.1089	0.1172	0.1183	0.1188
MAE_h8	h8	0.1278	0.1194	0.1283	0.1320	0.1300
MAE_h9	h9	0.1332	0.1255	0.1332	0.1321	0.1353
MAE_h10	h10	0.1354	0.1277	0.1369	0.1323	0.1376
MAE_h11	h11	0.1375	0.1330	0.1396	0.1442	0.1415
MAE_h12	h12	0.1373	0.1323	0.1401	0.1430	0.1403
MAE_h13	h13	0.1526	0.1449	0.1526	0.1524	0.1550
MAE_h14	h14	0.1485	0.1454	0.1511	0.1518	0.1543
MAE_Avg	Media	0.1182	0.1116	0.1195	0.1213	0.1196
N_valid	N_valid	261.0000	289.0000	289.0000	289.0000	285.0000

```
  t(rmse_summary[, -1])
)
colnames(rmse_transposed)[-1] <- rmse_summary$Model

knitr::kable(rmse_transposed, digits = 4,
             caption = "Root Mean Squared Error per orizzonte di previsione",
             booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position")
```

7.4 Visualizzazione Risultati

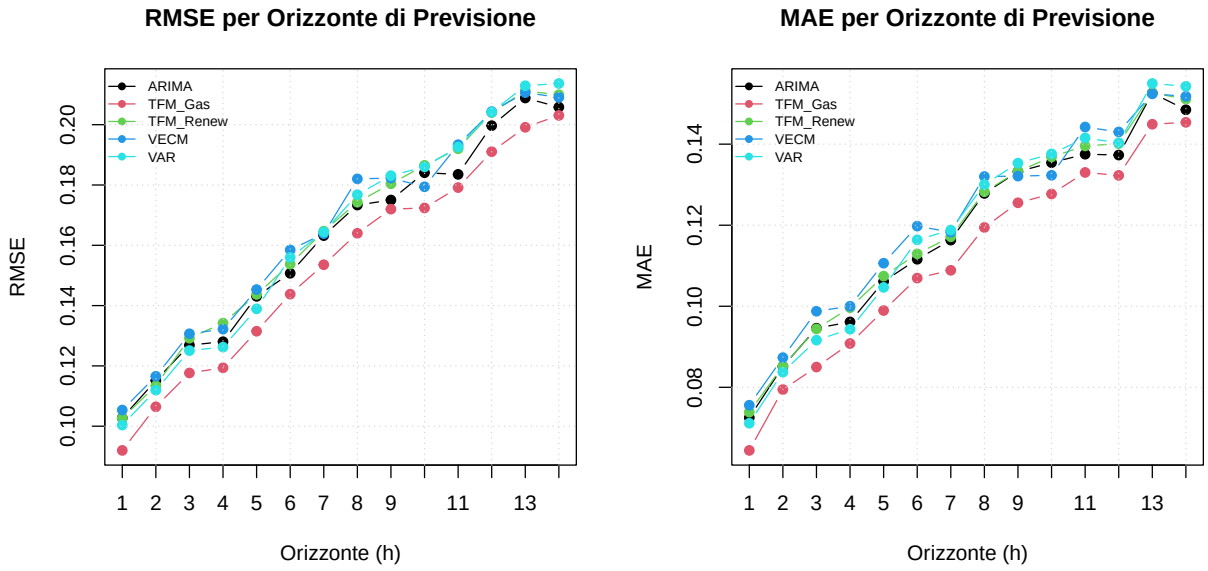
Visualizziamo graficamente l'andamento degli errori MAE e RMSE per ciascun orizzonte di previsione.

```
par(mfrow = c(1, 2))

# Grafico RMSE per Orizzonte
rmse_matrix_plot <- t(rmse_summary[, 2:15])

matplot(rmse_matrix_plot, type = "b", pch = 19,
        xlab = "Orizzonte (h)", ylab = "RMSE",
        main = "RMSE per Orizzonte di Previsione",
        col = 1:n_models, lty = 1, xaxt = "n")
axis(1, at = 1:14, labels = 1:14)
legend("topleft", legend = models, col = 1:n_models,
       lty = 1, pch = 19, cex = 0.7, bty = "n")
```

Tabella 8: Root Mean Squared Error per orizzonte di previsione

	Orizzonte	ARIMA	TFM_Gas	TFM_Renew	VECM	VAR
RMSE_h1	h1	0.1027	0.0920	0.1027	0.1054	0.1004
RMSE_h2	h2	0.1152	0.1064	0.1134	0.1165	0.1119
RMSE_h3	h3	0.1269	0.1176	0.1292	0.1307	0.1251
RMSE_h4	h4	0.1280	0.1194	0.1342	0.1322	0.1262
RMSE_h5	h5	0.1431	0.1315	0.1437	0.1453	0.1389
RMSE_h6	h6	0.1507	0.1438	0.1537	0.1585	0.1560
RMSE_h7	h7	0.1632	0.1536	0.1647	0.1639	0.1644
RMSE_h8	h8	0.1733	0.1640	0.1743	0.1820	0.1768
RMSE_h9	h9	0.1750	0.1720	0.1804	0.1823	0.1831
RMSE_h10	h10	0.1841	0.1724	0.1865	0.1794	0.1860
RMSE_h11	h11	0.1835	0.1791	0.1920	0.1934	0.1926
RMSE_h12	h12	0.1997	0.1910	0.2040	0.2043	0.2042
RMSE_h13	h13	0.2089	0.1991	0.2112	0.2106	0.2129
RMSE_h14	h14	0.2058	0.2031	0.2099	0.2091	0.2137
RMSE_Avg	Media	0.1648	0.1570	0.1678	0.1686	0.1678
N_valid	N_valid	261.0000	289.0000	289.0000	289.0000	285.0000

```
grid()

mae_matrix_plot <- t(mae_summary[, 2:15])

matplot(mae_matrix_plot, type = "b", pch = 19,
        xlab = "Orizzonte (h)", ylab = "MAE",
        main = "MAE per Orizzonte di Previsione",
        col = 1:n_models, lty = 1, xaxt = "n")
axis(1, at = 1:14, labels = 1:14)
legend("topleft", legend = models, col = 1:n_models,
       lty = 1, pch = 19, cex = 0.7, bty = "n")
grid()
```




```
par(mfrow = c(1, 1))
```

Le tabelle e i grafici convergono su un quadro coerente delle performance predittive. Il modello a funzioni di trasferimento con variabile di input Gas (TFM_Gas) risulta generalmente il più preciso su tutti gli orizzonti di previsione, con RMSE e MAE inferiori rispetto agli altri modelli. L'andamento degli errori al crescere dell'orizzonte temporale mostra una sostanziale stabilità per TFM_Gas, mentre gli altri modelli mostrano performance comparabili, leggermente peggiori del TFM_Gas. Questi risultati confermano quanto osservato nelle analisi precedenti: il Gas è la variabile principale in grado di spiegare le dinamiche del PUN, rendendo il modello a funzione di trasferimento con Gas come input quello più efficace sia nel breve che nel medio termine.

7.5 Test di Diebold-Mariano

Per valutare se le differenze di performance tra i modelli sono statisticamente significative, conduciamo il test di Diebold-Mariano confrontando i due modelli con RMSE medio più basso.

```
valid_models <- rmse_summary[!is.na(rmse_summary$RMSE_Avg), ]
top2_idx <- order(valid_models$RMSE_Avg)[1:2]
top2_models <- valid_models$Model[top2_idx]

dm_results <- data.frame(
  Horizon = integer(),
  DM_Statistic = numeric(),
  P_Value = numeric(),
  stringsAsFactors = FALSE
)

for(h in 1:h_max) {
  res1 <- residuals_list[[top2_models[1]]][, h]
  res2 <- residuals_list[[top2_models[2]]][, h]
```

```
  valid_idx <- !is.na(res1) & !is.na(res2)
  res1_clean <- res1[valid_idx]
  res2_clean <- res2[valid_idx]

  if(length(res1_clean) > 30) {
    dm_test <- dm.test(res1_clean, res2_clean,
                       alternative = "two.sided", h = h)

    dm_results <- rbind(dm_results,
                        data.frame(
                          Horizon = h,
                          DM_Statistic = dm_test$statistic,
                          P_Value = dm_test$p.value
                        ))
  }
}
```

```
knitr::kable(
  dm_results,
  digits = 4,
  caption = paste(
  "Test di Diebold-Mariano:",
  gsub("_", "\\\\_", top2_models[1]),
  "vs",
  gsub("_", "\\\\_", top2_models[2])
),
  booktabs = TRUE,
  escape = TRUE
)
```

Per tutti i 14 orizzonti di previsione, la statistica test risulta significativamente negativa, indicando una migliore capacità predittiva del modello TFM_Gas rispetto a ARIMA (secondo miglior modello). I p-value estremamente bassi confermano che la superiorità del primo modello non è dovuta al caso ma riflette una reale differenza strutturale nella capacità di catturare le dinamiche del PUN. Pertanto, concludiamo che il modello a funzione di trasferimento con Gas come variabile esplicativa è il più efficace tra quelli stimati, offrendo previsioni più accurate e stabili su tutto l'orizzonte temporale considerato.

8 Conclusioni

Questo studio ha analizzato le relazioni dinamiche tra prezzo del Gas naturale, produzione da Rinnovabili e Prezzo Unico Nazionale dell'energia elettrica nel mercato italiano durante il quinquennio 2021-2025, un periodo caratterizzato da turbolenze straordinarie dovute alla crisi energetica globale.

L'analisi univariata mediante modelli SARIMA ha rivelato marcata stagionalità settimanale in tutte le serie, riflesso della struttura dei consumi e della programmazione del mercato elettrico. Tuttavia, la forte eteroschedasticità e gli shock esogeni del 2022 hanno evidenziato i limiti dei modelli lineari univariati.

Tabella 9: Test di Diebold-Mariano: TFM_Gas vs ARIMA

	Horizon	DM_Statistic	P_Value
DM	1	-2.487	0.0135
DM1	2	-2.877	0.0043
DM2	3	-3.524	0.0005
DM3	4	-3.204	0.0015
DM4	5	-2.802	0.0055
DM5	6	-2.432	0.0157
DM6	7	-2.572	0.0107
DM7	8	-3.881	0.0001
DM8	9	-2.753	0.0063
DM9	10	-5.096	0.0000
DM10	11	-3.921	0.0001
DM11	12	-3.053	0.0025
DM12	13	-4.200	0.0000
DM13	14	-2.217	0.0275

L'analisi di cointegrazione ha confermato l'esistenza di una relazione di equilibrio di lungo periodo tra il prezzo del Gas e il PUN, con un'elasticità stimata attorno allo 0.84. La produzione rinnovabile, risultata stazionaria nei livelli, è stata esclusa dalla relazione di cointegrazione, suggerendo che il suo impatto sul PUN si manifesti principalmente attraverso dinamiche di breve periodo. Il Gas si è confermato come variabile debolmente esogena, guidando il sistema senza reagire agli squilibri, mentre il PUN si comporta come variabile endogena che assorbe gli shock con una velocità di aggiustamento del 13% per periodo.

I modelli a funzione di trasferimento hanno isolato le relazioni causali unidirezionali verso il PUN. Il modello Gas $\rightarrow$ PUN ha evidenziato un impatto immediato (0.54%) seguito da un effetto ritardato (0.24%), confermando il ruolo dominante del Gas nella formazione del prezzo elettrico. Il modello Rinnovabili $\rightarrow$ PUN ha mostrato una struttura più complessa, con un effetto positivo contemporaneo seguito da un effetto negativo ritardato, suggerendo dinamiche di compensazione che meriterebbero ulteriori approfondimenti con dati ad alta frequenza.

L'analisi VAR multivariata ha catturato le interdipendenze dinamiche tra le tre variabili. Il modello VAR(7) ristretto ha confermato la gerarchia causale emersa dalle analisi precedenti. Le funzioni di risposta all'impulso ortogonalizzate hanno rivelato come il PUN reagisca immediatamente agli shock del Gas e delle Rinnovabili, mentre queste ultime mostrano una risposta persistente agli shock del Gas. La decomposizione della varianza ha confermato che ciascuna variabile è guidata principalmente dai propri shock, con il Gas che mantiene un'influenza del 10-15% sul PUN e le Rinnovabili che contribuiscono per circa il 5%.

Il confronto predittivo out-of-sample ha fornito evidenza sulla superiorità del modello a funzione di trasferimento con Gas come input (TFM_Gas). L'analisi di rolling forecast su 14 giorni ha mostrato errori RMSE e MAE sistematicamente inferiori su ogni orizzonte temporale. Il test di Diebold-Mariano ha confermato statisticamente che tale superiorità riflette una reale maggiore capacità di catturare le dinamiche del PUN.

Questi risultati confermano che il prezzo del gas naturale rimane il principale driver del PUN, rendendo fondamentale il monitoraggio dei mercati internazionali del gas. Nonostante la crescente penetrazione delle fonti rinnovabili, il loro impatto sulla formazione del prezzo rimane secondario e mediato da meccanismi complessi di dispacciamento. La superiorità predittiva del modello TFM_Gas fornisce uno strumento operativo efficace per la previsione di breve-medio termine del PUN.

Lo studio presenta limitazioni che suggeriscono direzioni per ricerche future. La frequenza giornaliera dei dati non permette di catturare le dinamiche intra-giornaliere del mercato elettrico, particolarmente rilevanti per comprendere l'impatto delle rinnovabili. L'uso di dati orari potrebbe rivelare pattern più sofisticati. La presenza di eteroschedasticità persistente suggerisce che modelli GARCH multivariati o di switching regime potrebbero catturare meglio la volatilità del sistema. L'inclusione di variabili aggiuntive come la domanda di energia elettrica, la capacità di interconnessione e variabili meteorologiche potrebbe arricchire il quadro interpretativo e migliorare la capacità predittiva dei modelli.

8.1 Uso dell'AI

Nella produzione di questa relazione sono stati usati i seguenti strumenti di intelligenza artificiale generativa: Claude, Gemini, ChatGPT. Le finalità sono state le seguenti:

- Brainstorming iniziale per confrontare diverse idee
- Aiuto nell'individuazione delle banche dati necessarie
- Automazione ed efficientazione di alcune parti del codice (ad esempio, importazione e pulizia dati, o il ciclo del rolling forecast)
- Velocizzazione nella stesura del codice Markdown
- Validazione delle conclusioni tratte dai modelli
- Aiuto nel rendere più scorrevole la scrittura
- Controllo finale per incongruenze concettuali ed errori grammaticali