Università degli Studi di Padova

Dipartimento di Scienze Statistiche

Corso di Laurea Triennale in Statistica per l'Economia e l'Impresa


# Test per media e varianza (erroneamente specificate) variabili nel tempo

**Relatrice:**

**Prof.ssa Luisa Bisaglia**

**Laureanda:**

**Giorgia Barzan**

**Matricola: 1218944**

**Anno accademico 2021/2022**

# Riassunto

Lo studio che verrà presentato descrive le performance statistiche di test per proprietà variabili nel tempo quando media e/o varianza non sono correttamente specificate. Quando vengono testate le proprietà della media condizionale, in presenza di eteroschedasticità e media non variabile nel tempo, i test asintotici conducono a risultati distorti che vengono migliorati da test *wild bootstrap*. Anche i test asintotici per la varianza condizionale, in presenza di omoschedasticità e media variabile nel tempo, presentano differenze più o meno elevate tra livelli empirici e il livello nominale che, tuttavia, non vengono migliorate usando tecniche *bootstrap*. Nei capitoli che seguiranno verrà osservato come i test per la media abbiano delle performance adeguate sia in presenza di media che di varianza variabile nel tempo, mentre i test per la varianza conducano a risultati spuri in presenza di media variabile nel tempo.

# Indice

# Capitolo 1

# Introduzione

Nel corso degli ultimi decenni, in molte serie storiche economiche e finanziarie si è osservata la presenza di media e varianza condizionali variabili nel tempo. La media può presentare sia le costanti sia coefficienti autoregressivi variabili nel tempo, tali caratteristiche vengono studiate con l'ausilio di alcuni modelli, come lo STAR (*Smooth Transition Autoregressive*) introdotto da Chan e Tong nel 1986, o anche un modello variabile nel tempo a transizione graduale sviluppato da Lundbergh *et al.* nel 2003. La varianza del termine d'errore di un modello di regressione viene solitamente adattata dal modello ARCH (*Autoregressive Conditional Heteroskedasticity*) presentato inizialmente da Engle nel 1982. Tuttavia, come accade per la media, i parametri della varianza possono mutare nel tempo. Questa caratteristica viene studiata da alcuni modelli come lo STARCH (*Smooth Transition Autoregressive Conditional Heteroskedasticity*) introdotto da Gonzalez Rivera nel 1998, , il *time varying* ARCH (Dahlhaus e Rao (2006)) e da altri, più recenti, sviluppati da Amado e Teräsvirta (2013) e Kim e Kim (2016).

Per indagare le proprietà variabili nel tempo di serie storiche è comune utilizzare i test di linearità. Quando si vuole testare la linearità della media si presuppone l'omoschedasticità della varianza, ponendo come ipotesi nulla che la media sia lineare e come alternativa che la media vari nel tempo. Per testare la linearità della varianza del termine d'errore si agisce in modo analogo e si presuppone la linearità della media.

Analizzando una serie storica diventa arduo conoscere a priori la presenza di proprietà variabili nel tempo. Tale ostacolo porta indubbiamente i ricercatori a studiare

la presenza di media variabile nel tempo quando, al contrario, è la varianza che muta e viceversa. È stato infatti dimostrato da Pitarakis (2004) e Perron e Yamamoto (2019) che testare per un cambiamento nei coefficienti di regressione ignorando o non specificando correttamente la presenza di un cambiamento nella varianza, e viceversa, provoca scarse proprietà statistiche.

Lo studio che propongo riprende il lavoro effettuato da Maki e Ota (2021) ed ha lo scopo di esaminare i comportamenti di test sotto media e varianza condizionali non correttamente specificate. In questo elaborato evidenzio in particolar modo il comportamento dei test per la media in presenza di varianza variabile nel tempo, e quello dei test per la varianza in presenza di media variabile nel tempo.

Lo studio condotto da Lumsdaine e Ng (1999) ha dimostrato che i test per effetti ARCH sotto condizioni di media variabile nel tempo rifiutano eccessivamente l'ipotesi nulla; Van Dijk *et al.* (1999) hanno invece osservato che i test per effetti ARCH presentano differenze tra il livello nominale ed empirico quando un processo ha outliers additivi; Balke e Kapetanios (2007) hanno evidenziato che, ignorando la non linearità della media, si manifestano effetti ARCH spuri. Considerando queste possibili problematiche è necessario adoperare test adeguati e far luce sull'influenza di questi su modelli non correttamente specificati, così come è importante definire in modo adatto le proprietà di media e varianza variabili nel tempo per ottenere risultati affidabili e considerare il modello corretto.

Per la realizzazione di questo studio è stato utilizzato il modello STAR per modellare la media variabile nel tempo e il modello STARCH per modellare la varianza del termine d'errore. I test per linearità che sono stati utilizzati in questa tesi sono basati su un metodo introdotto da Luukkonen *et al.* (1988a) che adopera un'approssimazione di Taylor per superare il problema di identificazione evidenziato da Davies (1977,1987). Nel capitolo 2 viene presentata l'analisi di questi due modelli e nel capitolo successivo viene proposta una descrizione dei metodi *bootstrap* utilizzati nei test poi presentati nel capitolo 4. Nel capitolo 5 si esaminano le simulazioni, seguono poi un capitolo riguardante l'applicazione empirica e, infine, le conclusioni.

# Capitolo 2

# Modelli non lineari per serie storiche

L'identificazione e la stima dei modelli ARIMA, introdotti da Box e Jenkins (1970), sono stati i protagonisti dell'analisi delle serie storiche economiche per la maggior parte del ventesimo secolo. A partire dagli anni '90 la teoria economica ha cominciato a suggerire che le relazioni fra le variabili potessero non essere lineari, spingendo così gli statistici a sviluppare modelli non lineari e a testare nelle serie l'ipotesi nulla di linearità contro l'alternativa di non linearità. I modelli non lineari, a differenza di quelli lineari, permettono di tenere in considerazione grandi asimmetrie dei dati, salti improvvisi e irregolari, irreversibilità della serie storica e importanti dipendenze non lineari dei dati. Di seguito verranno presentate le caratteristiche di due modelli non lineari, il modello STAR (*Smooth Transition Autoregressive*) per la media e lo STARCH (*Smooth Transition Autoregressive Conditional Heteroskedasticity*) per la varianza, che saranno i protagonisti dello studio che verrà esposto nei capitoli successivi.

## 2.1 Il modello STAR

Le serie storiche vengono generalmente modellate utilizzando delle particolari estensioni dei modelli autoregressivi: il SETAR (*Self-Exciting Threshold Autoregressive*) e il TAR (*Threshold Autoregressive*), i quali forniscono una performance migliore dei modelli lineari attraverso un cambio di regime.

### 2.1.1 SETAR e TAR

Dato $y_t$, nella forma:

$$y_t = f(y_{t-1}, y_{t-2}, ..., y_{t-p}) + u_t \tag{2.1}$$

dove $u_t \sim (0, \sigma^2)$, il modello soglia, o SETAR, introdotto da Tong e Lim (1980) e Tong (1983), è un'approssimazione lineare a tratti della forma generale di autoregressione non lineare e si presenta nella seguente forma:

$$y_t = \alpha_{0j} + \sum_{i=1}^{p} \beta_{0ij} y_{t-i} + u_{tj} \quad se \quad c_{j-1} \leq y_{t-d} \leq c_j \tag{2.2}$$

dove $j = 1, ..., k$. $k$ è un intero positivo che indica il numero dei regimi, $u_{tj} \sim IID(0, \sigma_j^2)$, $d$ è l'operatore ritardo, $y_{t-d}$ è la variabile di soglia e $c_j$ con $j = 1, ..., k$ sono i parametri di soglia tali che $-\infty = c_0 < c_1 < ... < c_{k-1} < c_k = \infty$.
Per semplicità si considerano due regimi. Il modello SETAR può dunque essere scritto nel modo seguente:

$$y_t = (\alpha_0 + \sum_{i=1}^{p} \beta_{0i} y_{t-i}) I(y_{t-d} \leq c) + (\alpha_1 + \sum_{i=1}^{p} \beta_{1i} y_{t-i}) I(y_{t-d} > c) + u_t \tag{2.3}$$

dove $I(\cdot)$ è la funzione indicatrice che assume il valore 1 quando $(\cdot)$ è vera e 0 altrimenti.
Se alla variabile di soglia $y_{t-d}$ viene sostituita una variabile di soglia debolmente esogena $z_{t-d}$ si ottiene il modello TAR:

$$y_t = (\alpha_0 + \sum_{i=1}^{p} \beta_{0i} y_{t-i}) I(z_{t-d} \leq c) + (\alpha_1 + \sum_{i=1}^{p} \beta_{1i} y_{t-i}) I(z_{t-d} > c) + u_t. \tag{2.4}$$

Quando la variabile di soglia supera un determinato valore avviene un cambiamento di regime ciascuno caratterizzato da un diverso modello AR(p), pertanto, l'equazione della media condizionale non è continua, causando un cambiamento brusco fra un regime e l'altro.

### 2.1.2 STAR

Il cambiamento brusco fra regimi che avviene nei modelli SETAR non può essere considerato valido per una serie storica reale: è più ragionevole credere che avvenga un cambiamento graduale.
Sostituendo a $I(\cdot)$ una funzione liscia $F(\cdot)$ si genera il modello autoregressivo a transizione graduale che permette, dunque, di ottenere dei cambiamenti progressivi fra regimi. Tale modello è lo STAR (Chan and Tong (1986), Teräsvirta (1994), Van Dijk *et al.* (2002)), oppure LSTAR (*Logistic* STAR), e si presenta come come segue:

$$y_t = \alpha_0 + \sum_{i=1}^{p} \beta_{0i} y_{t-i} + (\alpha_1 + \sum_{i=1}^{p} \beta_{1i} y_{t-i}) F(y_{t-d}, \gamma, c) + u_t \tag{2.5}$$

$F(\cdot)$ è la funzione logistica di transizione:

$$F(y_{t-d}, \gamma, c) = \frac{1}{(1 + exp(-\gamma(y_{t-d} - c)))} - \frac{1}{2} \tag{2.6}$$

dove $u_t \sim N(0,1)$, $d$ è l'operatore ritardo, $y_{t-d}$ è la variabile di soglia, $\gamma$ è un numero reale e parametro di lisciamento, che rappresenta la velocità del processo di transizione, mentre $c$ è il parametro di soglia. Assumiamo $\gamma > 0$ e $c > 0$. Se $\gamma = \infty$ la $F(\cdot)$ assume il valore -1/2 o 1/2. Quando $y_{t-d} > c$ e $\gamma(y_{t-d} - c)$ è grande, allora $F(\cdot)$ tende a 1/2, mentre quando $y_{t-d} < c$ e $\gamma(y_{t-d} - c)$ è piccolo, $F(\cdot)$ tende a 1/2. In questo caso, l'equazione 2.5 diventa un modello TAR. Quando $\gamma \to 0$, $F(y_{t-d}, \gamma, c) = 0$ e l'equazione 2.5 risulta in un semplice modello AR(p).
La media condizionale del modello LSTAR si presenta come una combinazione lineare pesata delle equazioni 2.7, i cui pesi sono determinati in modo continuo dalla funzione di transizione $F(y_{t-d}, \gamma, c)$.

$$\mu_1 = \alpha_0 + \sum_{i=1}^{p} \beta_i y_{t-i}, \quad \mu_2 = \alpha_0 + \beta_0 + \sum_{i=1}^{p} (\alpha_i + \beta_i) y_{t-d} \tag{2.7}$$

La stima del parametro di lisciamento non è immediata, per questo motivo Teräsvirta (1994) ha suggerito di standardizzare la funzione di transizione $F(\cdot)$ per rendere $\gamma$ privo di scala tramite la divisione di tale parametro con la deviazione standard della variabile soglia $y_{t-d}$. Una volta conosciuti la variabile di transizione e il parametro

di lisciamento si possono stimare i parametri rimanenti tramite il metodo dei minimi quadrati non lineari (NLS) o la funzione di verosimiglianza. Dopo aver stimato il modello si può selezionare quello con il numero di ritardi più appropriati tramite l'uso dell'AIC (*Akaike Information Criteria*).

## 2.2 Il modello STARCH

Se la varianza del termine d'errore di una serie storica non è costante nel tempo i dati sono eteroschedastici. Utilizzando la procedura standard di stima dei minimi quadrati (OLS) si ottengono stime degli *standard error* e intervalli di confidenza troppo ristretti, dando così un falso senso di precisione. I modelli ARCH (*Autoregressive Conditional Heteroschedasticity*) permettono di modellare l'eteroschedasticità fornendo una misura della volatilità che può essere utilizzata per esempio in decisioni finanziarie.

### 2.2.1 ARCH

Il primo modello con eteroschedasticità condizionale, è il modello ARCH introdotto da Engle nel 1982 ed ha la seguente struttura:

$$y_t = \alpha_0 + \sum_{i=1}^{p} \beta_{0i} y_{t-i} + u_t \tag{2.8}$$

$$u_t = h_t \epsilon \tag{2.9}$$

$$h_t^2 = a_0 + \sum_{j=1}^{q} b_{0j} u_{t-j}^2 \tag{2.10}$$

dove $\epsilon \sim N(0,1)$, $a_0 > 0$, $b_{0j} \geq 0$, $j = 1, ..., q$, condizioni necessarie e sufficienti per la positività della varianza condizionale. Il processo ARCH ha media nulla, varianza non condizionale costante e varianza condizionale linearmente dipendente dai quadrati della storia passata di $u_t$.

### 2.2.2 STARCH

Il modello STARCH (Gonzalez Rivera (1998), Lee e Degennaro (2000)) è un tipo di modello ARCH che permette di studiare i cambiamenti nel tempo dei parametri dell'equazione della varianza condizionale.
Si definisce il modello a transizione graduale con eteroschedasticità condizionale come segue:

$$y_t = \alpha_0 + \sum_{i=1}^{p} \beta_{0i} y_{t-i} + u_t \tag{2.11}$$

$$u_t = h_t \epsilon \tag{2.12}$$

$$h_t^2 = a_0 + \sum_{j=1}^{q} b_{0j} u_{t-j}^2 + (a_1 + \sum_{j=1}^{q} b_{1j} u_{t-j}^2) F(u_{t-d}, \gamma, c). \tag{2.13}$$

Analogamente a quanto visto per il modello 2.5, la funzione di transizione logistica $F(\cdot)$ è

$$F(u_{t-d}, \gamma, c) = \frac{1}{(1 + exp(-\gamma(u_{t-d} - c)))} - \frac{1}{2}, \tag{2.14}$$

dove $u_t \sim N(0, ht^2)$ è la variabile di transizione, $h_t^2$ la varianza condizionale di $u_t$ che è il termine di errore del processo AR(p) (2.8), $\gamma$ è un numero reale che individua il parametro di lisciamento ed infine $c$ è il parametro di soglia. Si assume $\gamma > 0$ e $c > 0$. Se $\gamma = \infty$ la $F(\cdot)$ assume il valore -1/2 o 1/2. Quando $u_{t-d} > c$ e $\gamma(u_{t-d} - c)$ è grande allora $F(\cdot)$ tende a 1/2, mentre quando $u_{t-d} < c$ e $\gamma(u_{t-d} - c)$ è piccolo, $F(\cdot)$ tende a 1/2. In questo caso, l'equazione 2.13 diventa un modello threshold ARCH. Quando $\gamma \to 0$, $F(u_{t-d}, \gamma, c) = 0$ e l'equazione 2.13 risulta nel modello ARCH(p) di Engle (1982). In generale, dunque, la funzione di transizione logistica assume valori compresi fra -1/2 e 1/2.
La relazione tra i modelli STARCH e STAR è chiara: il modello STAR introduce il termine di transizione graduale nell'equazione della media (2.8), permettendo non linearità nei modelli di serie storiche univariate (Granger e Teräsvirta (1993), Teräsvirta (1994)), mentre i modelli STARCH estendono questa *smooth transition* all'equazione della varianza condizionale.
I parametri del modello vengono stimati con il metodo della massima verosimiglianza (MLE) e con il metodo dei minimi quadrati non lineari (NLS), analogamente al

modello 2.5. Si adopera il criterio AIC per scegliere il modello più adatto.

# Capitolo 3

# La tecnica bootstrap

Il *bootstrap*, introdotto nel 1979 da Efron, è una classe di metodi non parametrici che permette di fare inferenza statistica sul modello identificato senza effettuare ipotesi a priori. Questo metodo consente di stimare la distribuzione di un dato parametro della popolazione sulla base dei dati osservati. È stato dimostrato (Hesterberg (2011)) che l'efficacia del metodo *bootstrap* può superare di molto quella di metodi fondati su forti assunzioni strutturali. Viene infatti utilizzato per vari scopi, tra cui, come in questo studio, per ridurre la differenza tra il livello nominale ed empirico e per quantificare il comportamento del parametro di interesse.

## 3.1 Procedimento

Si supponga di avere a disposizione la distribuzione $F_n$ di una distribuzione qualsiasi da cui si estrae un campione di dati *i.i.d.* (indipendenti e identicamente distribuiti) di ampiezza n, $\boldsymbol{X_n} = \{X_1, \ldots, X_n\}$. Con questo campione si stima un parametro di interesse $\theta$, $\widehat{\theta}$, e la stima della distribuzione $F_n$, $\widehat{F_n}$. Se si ha a che fare con un *bootstrap* parametrico la distribuzione della popolazione sarà nota e dunque non è necessario stimarla. A questo punto $F_n$ viene considerata come la vera distribuzione della popolazione, da cui vengono estratti casualmente n elementi per formare un nuovo campione. Questa procedura viene ripetuta numerose volte generando una serie di campioni *bootstrap*. Per ciascun campione *bootstrap* viene calcolata la stima *bootstrap*, ossia il valore del parametro di interesse. L'insieme delle

stime *bootstrap* costituisce la distribuzione *bootstrap*, $F_n^*$, un'approssimazione della reale distribuzione campionaria del parametro d'interesse. Da questa distribuzione si ricava infine il valore *bootstrap* della statistica $\theta$ data dalla distribuzione *bootstrap*. Nelle sezioni successive vengono presentati i diversi tipi di *bootstrap* che sono stati utilizzati per realizzare questo progetto di tesi.

## 3.2 Bootstrap basato sui residui

Il metodo *bootstrap*, originariamente ideato da Efron, rappresenta un utile strumento se applicato ad un campione di variabili casuali indipendenti e identicamente distribuite, ma può non operare in modo ottimale in presenza di dipendenze tra i dati del campione come nel caso delle serie storiche. Per questo motivo la metodologia viene applicata ai residui assumendo che essi siano *i.i.d.*.
Definiamo $\boldsymbol{X_n} = \{X_1, \ldots, X_n\}$ una serie storica di dimensioni $n$ e $\widehat{m_n}(X_{t-1}, ..., X_{t-p})$ lo stimatore parametrico o non parametrico di $E[X_t|X_{t-1}, X_{t-2}, ..., X_{t-p}]$. Ricaviamo dunque i residui:

$$\widehat{e_t} = X_t - \widehat{m_n}(X_{t-1}, ..., X_{t-p}), \quad t = p+1, \ldots, n, \tag{3.1}$$

e successivamente la serie storica *bootstrap*:

$$X_t^* = \widehat{m_n}(X_{t-1}^*, ..., X_{t-p}^*) + e_t^*, \quad t = 1, \ldots, n, \tag{3.2}$$

dove gli $e_t^*$ vengono estratti casualmente con reinserimento dai residui stimati centrati con la loro media.

## 3.3 Wild bootstrap

Nel caso di assunzione di eteroschedasticità dei residui un ricampionamento casuale di essi porterebbe a distorsioni del modello. Per ovviare a questo problema si può adottare l'approccio *wild bootstrap*, introdotto prima da Wu (1986) e ripreso da Liu (1988), ponendo gli $e_t^*$ dell'equazione 3.2 uguali a $\widehat{e_t}\epsilon_t^*$, con $\epsilon_t^* \sim N(0, 1)$.
Se si utilizza uno stimatore non parametrico nei residui 3.1, indagare sulle proprietà

probabilistiche della serie 3.2 non sarebbe immediato. Questo in genere porta a stimatori non molto affidabili, dunque non può essere garantita la stabilità del processo *bootstrap*. Per stabilire una consistenza asintotica è necessaria la presenza di stabilità nel processo *bootstrap*. Pertanto si definisce, in sostituzione all'equazione 3.2:

$$X_t^* = \widehat{m_n}(X_{t-1}, ..., X_{t-p}) + e_t^*, \quad t = 1, \ldots, n \tag{3.3}$$

con $e_t^*$ pari a $\widehat{e_t}\epsilon_t^*$ con $\epsilon_t^* \sim N(0,1)$. Si osservi che il processo 3.3 non viene generato, a differenza della serie 3.2, dalle osservazioni ritardate *bootstrap*, bensì dalle osservazioni ritardate della serie originale. Questo fa sì che la serie *wild bootstrap* risulti comunque stabile in quanto rifletterà la distribuzione marginale p-dimensionale del processo reale per costruzione.

# Capitolo 4

# Test di linearità per media e varianza

## 4.1 Test per la media

Consideriamo il seguente modello di regressione LSTAR:

$$y_t = \alpha_0 + \sum_{i=1}^{p} \beta_{0i} y_{t-i} + (\alpha_1 + \sum_{i=1}^{p} \beta_{1i} y_{t-i}) F(t, \gamma, c) + u_t \tag{4.1}$$

dove $u_t \sim N(0, \sigma^2)$ e $F(\cdot)$ è la funzione di transizione logistica definita da:

$$F(t, \gamma, c) = \frac{1}{(1 + exp(-\gamma(t - c)))} - \frac{1}{2} \tag{4.2}$$

che, a differenza di quella mostrata nel capitolo 2, presenta la variabile $t$ come variabile di soglia. Gli altri parametri che descrivono il modello e il comportamento dello stesso al variare di essi sono analoghi a quelli di un generale modello LSTAR descritti precedentemente.

Per eseguire il test per la media variabile nel tempo si desiderano verificare le seguenti ipotesi:

$$H_0 : \gamma = 0, \quad H_1 : \gamma > 0. \tag{4.3}$$

Come precedentemente descritto nel paragrafo 2.1, sotto l'ipotesi nulla il modello 4.1 si riduce ad un modello autoregressivo di ordine p, mentre sotto l'ipotesi alternativa presenta una costante e una pendenza variabili nel tempo con $\alpha_1$ e/o $\beta_{1i}$. Testando

direttamente per $\gamma = 0$ si va incontro ad un problema di identificazione riguardo $\alpha_1$ e $\beta_{1i}$, che vengono identificati soltanto sotto l'ipotesi alternativa. Per ovviare a questo ostacolo viene quindi proposto un altro modello avente ipotesi nulla ed alternativa differenti, sfruttando una soluzione ideata da Luukkonen *et al.* (1988b), che vede l'utilizzo di un'approssimazione di Taylor per le serie storiche. Questa approssimazione indica che intorno a $\gamma = 0$ la funzione di transizione logistica assume un comportamento quasi lineare, come si può osservare dai seguenti grafici:

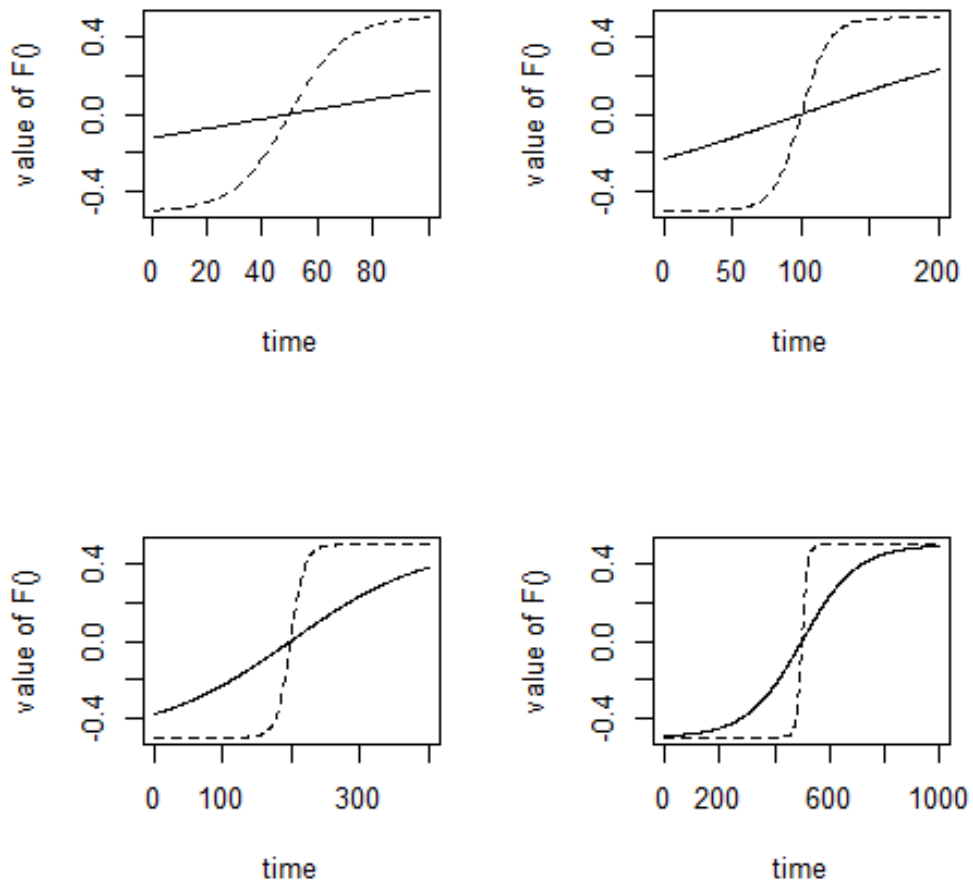


Figura 4.1: Valore della funzione di transizione logistica con T=100, 200, 400 e 1000. Le linee continue indicano la funzione $F(\cdot)$ con $\gamma = 0.01$, quelle tratteggiate con $\gamma = 0.1$.

Con numerosità pari a 100 e 200 e parametro di lisciamento pari a 0.01 la funzione $F(\cdot)$ è quasi lineare, mentre presenta un cambiamento graduale con $\gamma = 0.1$. Con una numerosità maggiore si evidenzia invece un cambiamento graduale con

parametro di lisciamento in prossimità dello 0 e un cambiamento brusco quando questo valore è uguale a 0.1.

Tenendo conto di queste correzioni, il modello di regressione 4.1 si trasforma in:

$$y_t = \phi_{10} + \sum_{i=1}^{p} \phi_{1i} y_{t-i} + \phi_{20} t + \sum_{i=1}^{p} \phi_{2i} t y_{t-i} + e_t \tag{4.4}$$

Dove $e_t$ è il termine di errore. A questo punto sarà necessario verificare le seguenti ipotesi:

$$H_0 : \phi_{20} = \phi_{21} = ... = \phi_{2p} = 0, \quad H_1 : \overline{H_0}. \tag{4.5}$$

Per testare la linearità della media definiamo la statistica di Wald in questo modo:

$$M_a = \widehat{\Phi_2}' V(\widehat{\Phi_2})^{-1} \widehat{\Phi_2} \tag{4.6}$$

dove $\widehat{\Phi_2}$ è la stima $\Phi_2 = (\phi_{20}, \phi_{21}, ..., \phi_{2p})'$, che è il vettore dei parametri legati alla componente variabile nel tempo, mentre $V(\widehat{\Phi_2})$ è la varianza di tale stima. La statistica test $M_a$ si distribuisce come una $\chi^2$ con $p+1$ gradi di libertà sotto l'ipotesi nulla.

I test asintotici, come quello appena presentato, possono segnalare la presenza di non linearità quando gli errori hanno varianza eteroschedastica. Per far fronte a questo problema si utilizzano dei test *bootstrap*.

Di seguito vengono presentati i passi dei tre test *bootstrap* per media variabile nel tempo:

1. Stimare il modello di regressione 4.4 con la serie storica a disposizione e calcolare la statistica $M_a$ 4.6 con il metodo dei minimi quadrati ordinari (OLS);

2. Stimare il modello di regressione sotto l'ipotesi nulla e ottenere le stime dei parametri e dei residui, $\widehat{e_{0t}}$ , tramite OLS;

3. Generare il seguente campione *bootstrap*:

$$y_t^* = \widehat{\phi_{10}} + \sum_{i=1}^{p} \widehat{\phi_{1i}} y_{t-i}^* + e_t^* \tag{4.7}$$

dove gli $e_t^*$ vengono estratti casualmente dai residui centrati del modello stimato sotto l'ipotesi nulla;

4. Stimare il modello di regressione 4.4 usando il campione *bootstrap* 4.7 e calcolare la statistica test $M_a$ 4.6 con OLS, denominata $M_b$;

5. Ripetere i punti 3 e 4 molte volte;

6. Calcolare il p-value del test *bootstrap* come segue:

$$M_b = \frac{1}{M}\sum_{m=1}^{M} I(M_{b(m)} > Ma) \tag{4.8}$$

dove $M$ è il numero di replicazioni *bootstrap* e $I(\cdot)$ è la funzione indicatrice che assume il valore 1 quando $(\cdot)$ è vera e 0 altrimenti. L'ipotesi nulla viene rifiutata quando il p-value è minore del livello di significatività del 5%.

Il secondo test *bootstrap* è simile a quello precedentemente presentato, a differenza del terzo punto in cui il campione ha una diversa determinazione degli errori, ovvero $e_t^* = \widehat{e_{0t}}\epsilon_t^*$, con $\epsilon_t^* \sim N(0,1)$, per tener conto della possibile eteroschedasticità del modello e rendendo il campione un *wild bootstrap*. Il p-value *bootstrap* viene indicato come $M_{wb1}$.

Viene infine proposto un terzo test dove, nel terzo step, viene generato un campione *wild bootstrap* del tipo:

$$y_t^* = \widehat{\phi_{10}} + \sum_{i=1}^{p} \widehat{\phi_{1i}} y_{t-i} + e_t^* \tag{4.9}$$

dove gli $e_t^* = \widehat{e_{0t}}\epsilon_t^*$, con $\epsilon_t^* \sim N(0,1)$ e p-value *bootstrap* $M_{wb2}$.

## 4.2 Test per la varianza

Consideriamo il seguente processo:

$$y_t = \alpha_0 + \sum_{i=1}^{p} \beta_{0i} y_{t-i} + u_t \tag{4.10}$$

$$u_t = h_t \epsilon \tag{4.11}$$

$$h_t^2 = a_0 + \sum_{j=1}^{q} b_{0j} u_{t-j}^2 + (a_1 + \sum_{j=1}^{q} b_{1j} u_{t-j}^2) F(t,\gamma,c) \tag{4.12}$$

Dove la funzione di transizione logistica si presenta come:

$$F(t,\gamma,c) = \frac{1}{(1 + exp(-\gamma(t-c)))} - \frac{1}{2}, \tag{4.13}$$

che, come per la media, presenta la variabile di soglia $t$.
Il modello rappresentato è un AR-STARCH di ordine p e q, dunque un processo AR(p) con residui che presentano una varianza eteroschedastica e che ha un comportamento STARCH(q).
Anche nel caso della varianza, si desiderano verificare le seguenti ipotesi:

$$H_0 : \gamma = 0, \quad H_1 : \gamma > 0. \tag{4.14}$$

Come si è visto in precedenza per la media, anche l'equazione 4.12 presenta un problema di identificazione, pertanto viene riscritta come:

$$h_t^2 = \rho_{10} + \sum_{j=1}^{q} \rho_{1j} u_{t-j}^2 + \rho_{20} t + \sum_{j=1}^{q} \rho_{2j} t u_{t-j}^2 + \nu_t \tag{4.15}$$

dove $\nu_t$ è il termine di errore. L'ipotesi nulla e alternativa si presentano nel modo seguente:

$$H_0 : \rho_{11} = ... = \rho_{1q} = \rho_{20} = \rho_{21} = ... = \rho_{2q} = 0, \quad H_1 : \overline{H_0}. \tag{4.16}$$

Per verificare la presenza di errori STARCH si conduce il test dei moltiplicatori di Lagrange (LM) di Engle (1982) e si definisce la statistica test in questo modo:

$$V_a = \frac{(SSR(h)_0 - SSR(h)_1)/(2q+1)}{SSR(h)_1/(T-(2q+2))}. \tag{4.17}$$

Per ottenere questa statistica si stima il modello 4.10 con i dati a disposizione, si ottengono i residui OLS e si stima il modello 4.15. Successivamente si calcola la somma dei quadrati dei residui ottenuti dalla stima di quest'ultimo sotto $H_0$, dando luogo a $SSR(h)_0$, dopodiché si ricava la somma dei quadrati dei residui ottenuti dalla stima sotto $H_1$, generando $SSR(h)_1$. La statistica $V_a$ ha una distribuzione $F$ con $(q+1, T-(2q+2))$ gradi di libertà e si distribuisce asintoticamente come una $\chi^2$ con $(2q+1)$ gradi di libertà.
Per diminuire la differenza tra i livelli empirici e il livello nominale ed aumentare la potenza del test asintotico viene utilizzato un approccio *bootstrap* per gli ARCH introdotto da Gel e Chen (2012).

Di seguito vengono presentati i passi per i due test *bootstrap* presi in considerazione:

1. Stimare il modello di regressione 4.15 con i dati in esame e calcolare la statistica $V_a$, 4.17, con il metodo dei minimi quadrati ordinari (OLS);
2. Stimare il modello di regressione sotto l'ipotesi nulla e ottenere le stime del parametro $\rho_{10}$ e dei residui, $\widehat{e_{0t}}$, tramite OLS;
3. Generare il seguente campione *bootstrap*:

$$h_t^* = \widehat{\rho_{10}} + \nu_t^* \tag{4.18}$$

dove gli $e_t^*$ vengono estratti casualmente dai residui del modello stimato sotto l'ipotesi nulla;

4. Stimare il modello di regressione 4.15 usando il campione *bootstrap* 4.18 e calcolare la statistica test $V_a$, 4.17, con OLS, denominata $V_b$;
5. Ripetere i punti 3 e 4 molte volte;
6. Calcolare il p-value del test *bootstrap* come segue:

$$V_b = \frac{1}{M} \sum_{m=1}^{M} I(V_{b(m)} > V_a) \tag{4.19}$$

dove $M$ è il numero di replicazioni *bootstrap* e $I(\cdot)$ è la funzione indicatrice, che assume il valore 1 quando $(\cdot)$ è vera e 0 altrimenti. L'ipotesi nulla viene rifiutata quando il p-value è minore del livello di significatività del 5%

Il secondo test è il *wild bootstrap* e differisce da quello presentato sopra nel terzo punto, dove gli $e_t^* = \widehat{e_{0t}}\epsilon_t^*$, con $\epsilon_t^* \sim N(0,1)$. Il p-value del *wild bootstrap* diventerà dunque:

$$V_w b = \frac{1}{M} \sum_{m=1}^{M} I(V_{wb(m)} > V_a) \tag{4.20}$$

.

# Capitolo 5

# Frequenze di rigetto dei test per media e varianza variabili nel tempo

Per confrontare il livello e la potenza dei test statistici presentati nei capitoli precedenti sono stati condotti degli esperimenti Monte Carlo. Per ogni test sono state eseguite 10000 replicazioni, ad eccezione per i test *bootstrap* che sono stati ripetuti 1000 volte con 200 replicazioni *bootstrap*. Le numerosità dei campioni prese in considerazione sono T=100, 200, 400 e 1000, mentre il livello nominale è stato posto al 5%.

Per maggiore chiarezza e per semplificare il procedimento le simulazioni che si andranno a descrivere riguardano processi con lag 1. Le stesse operazioni possono essere eseguite anche su processi con lag maggiori.

Di seguito verranno presentati i test per la media, assumendo omoschedasticità del processo, e i test per la varianza, effettuati assumendo linearità della media.

## 5.1 Simulazioni sotto modello AR con errore omoschedastico

Le prime simulazioni sono state effettuate a partire da un semplice processo autoregressivo di ordine 1 con errore omoschedastico, che si presenta in questo modo:

$$y_t = \alpha_0 + \beta_0 y_{t-1} + u_t \tag{5.1}$$

dove $u_t \sim i.i.d.N(0,1)$. Per analizzare l'influenza del passato si considerano i valori del parametro di persistenza $\beta_0 = 0.3$ e 0.9.
La tabella 5.1 mostra le frequenze di rigetto per i 7 test descritti nel capitolo 3. Per ottenere i risultati per il test $M_a$ sono state svolte le operazioni indicate nel capitolo 3, calcolando il test di Wald. È stata registrata la quantità di volte in cui il valore del p-value restituito dal test di Wald fosse risultato inferiore al valore nominale 0.05, in seguito si è ottenuta la frequenza di rigetto dividendo tale quantità per il numero di simulazioni effettuate.

| | $M_a$ | $M_b$ | $M_{wb1}$ | $M_{wb2}$ | $V_a$ | $V_b$ | $V_{wb}$ |
|---|---|---|---|---|---|---|---|
| $\beta_0 = 0.3$ | | | | | | | |
| T=100 | 0.048 | 0.046 | 0.038 | 0.051 | 0.037 | 0.044 | 0.018 |
| T=200 | 0.047 | 0.051 | 0.019 | 0.048 | 0.068 | 0.050 | 0.015 |
| T=400 | 0.048 | 0.052 | 0.029 | 0.048 | 0.058 | 0.052 | 0.023 |
| T=1000 | 0.047 | 0.046 | 0.038 | 0.049 | 0.057 | 0.051 | 0.035 |
| $\beta_0 = 0.9$ | | | | | | | |
| T=100 | 0.107 | 0.049 | 0.048 | 0.101 | 0.080 | 0.050 | 0.019 |
| T=200 | 0.077 | 0.051 | 0.050 | 0.075 | 0.057 | 0.057 | 0.025 |
| T=400 | 0.058 | 0.052 | 0.061 | 0.064 | 0.051 | 0.051 | 0.028 |
| T=1000 | 0.053 | 0.060 | 0.052 | 0.056 | 0.070 | 0.082 | 0.031 |

Tabella 5.1: Frequenze di rigetto sotto modello AR con errore omoschedastico

Le frequenze di rigetto per i test *bootstrap* sulla media sono state ottenute in modo analogo, a differenza del fatto che in ogni replicazione sono state create delle statistiche *bootstrap*, si è calcolato il p-value *bootstrap*, come indicato nel capitolo 3, e infine si è contato il numero di volte in cui tale p-value è risultato inferiore a 0.05. Questi conteggi infine sono stati divisi per il numero di simulazioni effettuate.
Per quanto riguarda invece i test per la varianza sono stati svolti gli step presentati nel capitolo precedente per la creazione della statistica $V_a$. La realizzazione dei

conteggi è avvenuta sommando la quantità di volte in cui la statistica $V_a$ è risultata maggiore del quantile della $F$ al livello $(q+1, T-(2q+2))$, dunque al livello $(2, T-4)$. Le frequenze di rigetto per i test *bootstrap* della varianza sono poi state calcolate analogamente ai test *bootstrap* per la media.

Il processo 5.1 non presenta media o varianza variabile nel tempo, perciò le frequenze di rigetto presentate nella tabella 1 corrispondono ai livelli empirici del modello 5.1. Il test $M_a$ con il valore del parametro di persistenza uguale a 0.3 registra livelli empirici attorno al valore del livello nominale, indipendentemente dalla grandezza del campione. Per il valore di $\beta_0$ pari a 0.9 si registrano rifiuti maggiori del 5%, che decrescono però con l'aumentare della numerosità campionaria. Il test $M_{wb1}$ presenta delle leggere distorsioni rispetto al valore nominale e un comportamento conservativo per $\beta_0 = 0.3$. Questo non accade per $M_{wb2}$ se si prende in considerazione $\beta_0 = 0.3$, mentre con $\beta_0 = 0.9$ si registrano valori più elevati con successiva decrescenza, analogamente a quanto accade per $M_a$. Il test $M_b$ registra per entrambi i valori di $\beta_0$ delle frequenze di rigetto attorno al valore nominale. I test per la varianza variabile nel tempo riportano invece dei valori pressoché attorno al 5%, ad eccezione per la $V_a$ con T=100 e $\beta_0 = 0.3$ in cui si nota un comportamento conservativo. Per $\beta_0 = 0.9$ e T=100 e 1000 si ottiene un rigetto maggiore rispetto al valore nominale. In quanto al test $V_b$ si registra un valore più elevato per $\beta_0 = 0.9$ e T=1000, mentre si osservano valori nella norma per $\beta_0 = 0.3$. Il test $V_{wb}$ invece risulta essere abbastanza conservativo.

## 5.2 Simulazioni sotto modello STAR con ed errore omoschedastico

Il secondo processo di dati che è stato preso in considerazione è il seguente:

$$y_t = 1 + 0.3y_{t-1} + (\alpha_1 + \beta_1 y_{t-1})F(t, \gamma, c) + u_t \tag{5.2}$$

$$F(t, \gamma, c) = \frac{1}{(1 + exp(-\gamma(t - c)))} - \frac{1}{2} \tag{5.3}$$

con $u_t \sim N(0, \sigma^2)$.

I risultati riportati nella tabella 2 sono stati ricavati ponendo $(\alpha_1, \beta_1)$=(0,0.3),

(0,0.6), (0.5,0.3), (1,0.3). Il parametro di lisciamento $\gamma$ è invece stato posto a 0.01, per verificare i test nel caso di un cambiamento più leggero, e poi a 0.1. Il parametro di soglia c assume il valore T/2, con T=100, 200, 400, 1000.

| $\gamma = 0.01$ | | | | | $\gamma = 0.1$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $M_a$ | $M_b$ | $M_{wb1}$ | $M_{wb2}$ | $M_a$ | $M_b$ | $M_{wb1}$ | $M_{wb2}$ |
| $\alpha_1 = 0, \beta_1 = 0.3$ | | | | | | | | |
| T=100 | 0.062 | 0.065 | 0.065 | 0.067 | 0.398 | 0.375 | 0.390 | 0.390 |
| T=200 | 0.132 | 0.110 | 0.125 | 0.128 | 0.772 | 0.795 | 0.770 | 0.810 |
| T=400 | 0.631 | 0.621 | 0.642 | 0.595 | 0.985 | 0.975 | 0.990 | 0.960 |
| T=1000 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\alpha_1 = 0, \beta_1 = 0.6$ | | | | | | | | |
| T=100 | 0.092 | 0.075 | 0.104 | 0.075 | 0.938 | 0.950 | 0.950 | 0.925 |
| T=200 | 0.451 | 0.445 | 0.445 | 0.465 | 1 | 1 | 1 | 1 |
| T=400 | 0.998 | 1 | 0.991 | 1 | 1 | 1 | 1 | 1 |
| T=1000 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\alpha_1 = 0.5, \beta_1 = 0.3$ | | | | | | | | |
| T=100 | 0.095 | 0.081 | 0.090 | 0.105 | 0.891 | 0.880 | 0.874 | 0.883 |
| T=200 | 0.382 | 0.362 | 0.451 | 0.382 | 0.999 | 1 | 1 | 1 |
| T=400 | 0.992 | 0.992 | 0.983 | 0.985 | 1 | 1 | 1 | 1 |
| T=1000 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\alpha_1 = 1, \beta_1 = 0.3$ | | | | | | | | |
| T=100 | 0.144 | 0.105 | 0.147 | 0.145 | 0.995 | 0.985 | 0.990 | 0.990 |
| T=200 | 0.709 | 0.675 | 0.683 | 0.685 | 1 | 1 | 1 | 1 |
| T=400 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| T=1000 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Tabella 5.2: Frequenze di rigetto dei test sulla media sotto modello AR con media condizionale variabile nel tempo ed errore omoschedastico

Osservando la tabella 5.2 si può chiaramente notare come la potenza empirica dei test per la media variabile nel tempo sia soggetta sia a cambiamenti dei parametri $\alpha_1$, $\beta_1$ e $\gamma$ che all'aumentare della numerosità del campione. Si nota come, al crescere del valore dei parametri $\alpha_1$ e $\beta_1$, aumenti anche la potenza dei test. L'influenza maggiore viene imposta dal parametro di lisciamento e, con $\gamma = 0.1$, si evidenzia la media variabile nel tempo con errore omoschedastico più velocemente rispetto ai test sul modello con $\gamma = 0.01$. All'aumentare della numerosità campionaria aumenta anche la potenza del test: ciò può essere attribuito al fatto che, come illustrato nella figura 4.1 la funzione di transizione logistica $F(\cdot)$ con T=100 e 200 si avvicina molto ad una funzione lineare con $\gamma = 0.01$ ed ha un cambiamento graduale al livello T/2 con $\gamma = 0.1$, con T=400 e T=1000 si osserva invece una transizione graduale con

| $\gamma = 0.01$ | | | | $\gamma = 0.1$ | | |
|---|---|---|---|---|---|---|
| | $V_a$ | $V_b$ | $V_{wb}$ | $V_a$ | $V_b$ | $V_{wb}$ |
| $\alpha_1 = 0, \beta_1 = 0.3$ | | | | | | |
| T=100 | 0.065 | 0.075 | 0.019 | 0.079 | 0.091 | 0.015 |
| T=200 | 0.095 | 0.087 | 0.013 | 0.077 | 0.088 | 0.010 |
| T=400 | 0.074 | 0.083 | 0.019 | 0.075 | 0.072 | 0.035 |
| T=1000 | 0.084 | 0.107 | 0.035 | 0.087 | 0.105 | 0.040 |
| $\alpha_1 = 0, \beta_1 = 0.6$ | | | | | | |
| T=100 | 0.083 | 0.087 | 0.015 | 0.076 | 0.087 | 0.011 |
| T=200 | 0.072 | 0.080 | 0.013 | 0.114 | 0.133 | 0.012 |
| T=400 | 0.115 | 0.092 | 0.015 | 0.233 | 0.274 | 0.022 |
| T=1000 | 0.346 | 0.325 | 0.190 | 0.636 | 0.714 | 0.055 |
| $\alpha_1 = 0.5, \beta_1 = 0.3$ | | | | | | |
| T=100 | 0.084 | 0.080 | 0.015 | 0.073 | 0.075 | 0.005 |
| T=200 | 0.066 | 0.089 | 0.015 | 0.076 | 0.074 | 0.023 |
| T=400 | 0.075 | 0.074 | 0.015 | 0.097 | 0.111 | 0.035 |
| T=1000 | 0.087 | 0.098 | 0.031 | 0.124 | 0.155 | 0.091 |
| $\alpha_1 = 1, \beta_1 = 0.3$ | | | | | | |
| T=100 | 0.089 | 0.081 | 0.010 | 0.066 | 0.069 | 0.025 |
| T=200 | 0.086 | 0.087 | 0.015 | 0.091 | 0.083 | 0.032 |
| T=400 | 0.068 | 0.081 | 0.020 | 0.115 | 0.121 | 0.061 |
| T=1000 | 0.116 | 0.131 | 0.045 | 0.252 | 0.297 | 0.176 |

Tabella 5.3: Frequenze di rigetto dei test sulla varianza sotto modello AR con media condizionale variabile nel tempo ed errore omoschedastico

$\gamma = 0.01$ e un cambiamento rapido con $\gamma = 0.1$, portando ad avere una segnalazione più repentina della presenza di media variabile nel tempo. Si può affermare quindi che i test $M_a$, $M_b$, $M_{wb1}$ e $M_{wb2}$ riescono a segnalare in modo accettabile la presenza di una media condizionale che varia nel tempo.

I test per la varianza riportati nella tabella 5.3 rappresentano le dimensioni empiriche, vista la presenza di omoschedasticità nel processo preso in considerazione. Si registrano delle distorsioni dei test $V_a$ e $V_b$ per campioni più ristretti, che vanno ad aumentare con l'incremento della numerosità campionaria. Le distorsioni aumentano anche con un valore del parametro di lisciamento maggiore, ad eccezione per T=100 e T=200 e $\alpha_1 = 0.5$ e 1, e con l'incremento del parametro $\beta_1$; un'influenza seppur minore si registra anche per l'aumento della costante $\alpha_1$. Le distorsioni dei test $V_a$ e $V_b$ vengono migliorate dall'utilizzo del test *wild bootstrap* che, però, risulta molto conservativo e non riesce a comportarsi in modo adatto con numerosità elevata e $\gamma = 0.1$, avendo una frequenza di rigetto pari a 0.176 quando $\alpha_1 = 1$, $\beta_1 = 0.3$

e T=1000. Si conclude pertanto che, in presenza di media variabile nel tempo e con errore omoschedastico, i test per la varianza non si comportano in modo affidabile. Le differenze tra i livelli empirici e il livello nominale riscontrate sono state attribuite da Balke e Kapetanios (2007) alla trascuratezza della non linearità della media, che causa eteroschedasticità spuria. Non avendo specificato correttamente la media, la non linearità non considerata viene inglobata dai residui, facendo segnalare dai test per la varianza la presenza di eteroschedasticità nei residui.

## 5.3 Simulazioni sotto modello AR con errori ARCH

In questa sezione si analizzano le frequenze di rigetto dei test in presenza di un processo AR-ARCH definito come segue:

$$y_t = 1 + 0.3y_{t-1} + u_t \tag{5.4}$$

$$u_t = h_t\epsilon \tag{5.5}$$

$$h_t^2 = a_0 + b_0u_{t-1}^2. \tag{5.6}$$

Si considera $a_0 = 1$ e $b_0 = 0.3$, 0.6, 0.9 per valutare l'effetto della persistenza. Il processo 5.4 non presenta media variabile nel tempo, dunque i risultati di $M_a$, $M_b$, $M_{wb1}$, $M_{wb2}$ mostrano i livelli empirici. Ci si aspetterebbe che i test per la media non rifiutino l'ipotesi di linearità e che quelli per la varianza segnalino velocemente la presenza dell'effetto ARCH. In realtà dai risultati si può notare che i test $M_a$ e $M_b$ presentano delle distorsioni che crescono con l'aumentare del parametro $b_0$ e della numerosità campionaria, raggiungendo la frequenza di rigetto più alta pari a 0.388 quando $b_0 = 0.9$ e T=1000; tali distorsioni vengono corrette dai test *wild bootstrap* per la media, i quali presentano frequenze di rigetto attorno al valore nominale e riescono dunque a gestire meglio la presenza di eteroschedasticità.

Come si può osservare dai risultati riportati nella tabella 5.3, il comportamento del test $V_{wb}$ è migliore rispetto a quello degli altri test per la varianza in caso di omoschedasticità e media variabile nel tempo. Nella tabella 5.4 è evidente che il test $V_{wb}$ sia più lento di $V_a$ e $V_b$ nel rifiutare l'ipotesi nulla, il primo possiede dunque una potenza inferiore agli altri test in presenza di errori ARCH. Ne consegue, però,

| | $M_a$ | $M_b$ | $M_{wb1}$ | $M_{wb2}$ | $V_a$ | $V_b$ | $V_{wb}$ |
|---|---|---|---|---|---|---|---|
| $b_0 = 0.3$ | | | | | | | |
| T=100 | 0.079 | 0.056 | 0.050 | 0.048 | 0.333 | 0.398 | 0.080 |
| T=200 | 0.086 | 0.073 | 0.053 | 0.052 | 0.627 | 0.697 | 0.132 |
| T=400 | 0.102 | 0.083 | 0.051 | 0.052 | 0.918 | 0.922 | 0.344 |
| T=1000 | 0.106 | 0.093 | 0.052 | 0.051 | 0.969 | 0.989 | 0.587 |
| $b_0 = 0.6$ | | | | | | | |
| T=100 | 0.113 | 0.093 | 0.056 | 0.051 | 0.625 | 0.655 | 0.281 |
| T=200 | 0.142 | 0.168 | 0.055 | 0.052 | 0.916 | 0.954 | 0.397 |
| T=400 | 0.187 | 0.176 | 0.056 | 0.051 | 0.983 | 0.999 | 0.594 |
| T=1000 | 0.236 | 0.230 | 0.057 | 0.055 | 1 | 1 | 0.722 |
| $b_0 = 0.9$ | | | | | | | |
| T=100 | 0.137 | 0.135 | 0.058 | 0.056 | 0.752 | 0.832 | 0.340 |
| T=200 | 0.214 | 0.206 | 0.059 | 0.056 | 0.959 | 0.980 | 0.441 |
| T=400 | 0.285 | 0.274 | 0.062 | 0.050 | 0.986 | 0.999 | 0.523 |
| T=1000 | 0.388 | 0.379 | 0.060 | 0.055 | 0.997 | 1 | 0.579 |

Tabella 5.4: Frequenze di rigetto sotto modello AR con errori ARCH

che le potenze maggiori di $V_a$ e $V_b$ non possono essere considerate attendibili, viste le differenze tra i livelli empirici e il livello nominale evidenziate dalla tabella 5.3: non si può dunque affermare con certezza che $V_{wb}$ possiede una potenza minore degli altri sotto media lineare ed errori ARCH.

## 5.4 Simulazioni sotto modello AR con errori STARCH

L'ultimo processo generatore di dati considerato è un processo AR con errori STARCH, definito come:

$$y_t = 1 + 0.3y_{t-1} + u_t \tag{5.7}$$

$$u_t = h_t\epsilon \tag{5.8}$$

$$h_t^2 = 1 + 0.3u_{t-1}^2 + (a_1 + b_1u_{t-1}^2)F(t, \gamma, c). \tag{5.9}$$

Si studiano le frequenze di rigetto per i parametri $(a_1, b_1)$=(0,0.3), (0,0.6), (0.5,0.3), (1,0.3), $\gamma = 0.01$ e 0.1 e T=100, 200, 400, 1000.

Osservando i risultati dei test sulla media riportati nella tabella 5.5 si nota che per le statistiche $M_a$ e $M_b$ si hanno alcune distorsioni che dipendono unicamente dall'aumento della numerosità campionaria: non sono soggette all'aumento del parametro di persistenza e del parametro di lisciamento. Per i diversi valori del parametro

| $\gamma = 0.01$ | | | | | $\gamma = 0.1$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $M_a$ | $M_b$ | $M_{wb1}$ | $M_{wb2}$ | $M_a$ | $M_b$ | $M_{wb1}$ | $M_{wb2}$ |
| $a_1 = 0, b_1 = 0.3$ | | | | | | | | |
| T=100 | 0.081 | 0.073 | 0.056 | 0.050 | 0.079 | 0.076 | 0.051 | 0.050 |
| T=200 | 0.094 | 0.089 | 0.057 | 0.053 | 0.089 | 0.080 | 0.054 | 0.050 |
| T=400 | 0.105 | 0.103 | 0.056 | 0.053 | 0.098 | 0.101 | 0.056 | 0.052 |
| T=1000 | 0.117 | 0.109 | 0.051 | 0.054 | 0.123 | 0.111 | 0.055 | 0.051 |
| $a_1 = 0, b_1 = 0.6$ | | | | | | | | |
| T=100 | 0.081 | 0.073 | 0.051 | 0.054 | 0.087 | 0.084 | 0.057 | 0.050 |
| T=200 | 0.094 | 0.081 | 0.056 | 0.052 | 0.096 | 0.093 | 0.058 | 0.053 |
| T=400 | 0.113 | 0.112 | 0.050 | 0.051 | 0.107 | 0.108 | 0.060 | 0.053 |
| T=1000 | 0.160 | 0.153 | 0.057 | 0.055 | 0.145 | 0.153 | 0.062 | 0.052 |
| $a_1 = 0.5, b_1 = 0.3$ | | | | | | | | |
| T=100 | 0.079 | 0.083 | 0.054 | 0.051 | 0.083 | 0.085 | 0.054 | 0.054 |
| T=200 | 0.088 | 0.096 | 0.058 | 0.056 | 0.092 | 0.093 | 0.054 | 0.055 |
| T=400 | 0.101 | 0.096 | 0.052 | 0.054 | 0.103 | 0.103 | 0.051 | 0.051 |
| T=1000 | 0.120 | 0.099 | 0.056 | 0.056 | 0.107 | 0.106 | 0.052 | 0.050 |
| $a_1 = 1, b_1 = 0.3$ | | | | | | | | |
| T=100 | 0.076 | 0.085 | 0.057 | 0.050 | 0.076 | 0.087 | 0.052 | 0.050 |
| T=200 | 0.092 | 0.086 | 0.054 | 0.053 | 0.087 | 0.090 | 0.054 | 0.054 |
| T=400 | 0.112 | 0.109 | 0.056 | 0.053 | 0.092 | 0.101 | 0.057 | 0.048 |
| T=1000 | 0.123 | 0.133 | 0.055 | 0.056 | 0.115 | 0.116 | 0.056 | 0.051 |

Tabella 5.5: Frequenze di rigetto dei test della media sotto modello AR con errori STARCH

di persistenza, infatti, si evidenzia che le frequenze di rigettono sono circa uguali all'8% con T=100 e raggiungono il $10-12\%$ con T=1000. Il comportamento del test viene migliorato dai test *wild bootstrap*, che presentano frequenze prossime al valore nominale, grazie alla loro capacità di tenere in considerazione l'eteroschedasticità dei termini di errore. Per quanto riguarda i test per la varianza, nella tabella 5.6 si nota che i test $V_b$ segnalano più velocemente la presenza di errori STARCH rispetto a $V_a$. La grandezza delle potenze empiriche di questi test è soggetta all'incremento dei parametri $a_1$ e $b_1$ ma soprattutto all'aumento di $\gamma$, poiché con il parametro di lisciamento prossimo allo zero si ha una funzione di transizione quasi lineare per numerosità minori e con un salto graduale per T grandi. Come riportato nella tabella 5.4, anche in questo caso $V_{wb}$ risulta avere minore potenza rispetto agli altri test della varianza. Si può quindi affermare che $V_{wb}$ sotto media variabile nel tempo ha performance migliori di $V_a$ e $V_b$ , che a loro volta hanno potenza maggiore di $V_{wb}$.
Le conclusioni che si possono trarre da queste simulazioni di Monte Carlo sono chia-

re: i test *wild boostrap* per la media si comportano bene in presenza di errori ARCH ed errori STARCH, migliorando la performance dei test $M_a$ e $M_b$, mentre i test per la varianza non si comportano in modo adeguato in presenza di media variabile nel tempo.

| $\gamma = 0.01$ | | | | $\gamma = 0.1$ | | |
|---|---|---|---|---|---|---|
| | $V_a$ | $V_b$ | $V_{wb}$ | $V_a$ | $V_b$ | $V_{wb}$ |
| $a_1 = 0, b_1 = 0.3$ | | | | | | |
| T=100 | 0.358 | 0.381 | 0.084 | 0.396 | 0.383 | 0.095 |
| T=200 | 0.649 | 0.673 | 0.120 | 0.737 | 0.761 | 0.155 |
| T=400 | 0.930 | 0.944 | 0.311 | 0.939 | 0.955 | 0.409 |
| T=1000 | 0.999 | 0.999 | 0.603 | 0.999 | 1 | 0.668 |
| $a_1 = 0, b_1 = 0.6$ | | | | | | |
| T=100 | 0.355 | 0.415 | 0.088 | 0.498 | 0.563 | 0.172 |
| T=200 | 0.682 | 0.781 | 0.111 | 0.825 | 0.849 | 0.294 |
| T=400 | 0.963 | 0.980 | 0.412 | 0.971 | 0.994 | 0.497 |
| T=1000 | 1 | 1 | 0.750 | 1 | 1 | 0.792 |
| $a_1 = 0.5, b_1 = 0.3$ | | | | | | |
| T=100 | 0.329 | 0.416 | 0.089 | 0.518 | 0.598 | 0.250 |
| T=200 | 0.663 | 0.781 | 0.137 | 0.878 | 0.879 | 0.452 |
| T=400 | 0.974 | 0.986 | 0.379 | 0.982 | 0.999 | 0.582 |
| T=1000 | 1 | 1 | 0.792 | 1 | 1 | 0.809 |
| $a_1 = 1, b_1 = 0.3$ | | | | | | |
| T=100 | 0.375 | 0.419 | 0.160 | 0.727 | 0.769 | 0.511 |
| T=200 | 0.748 | 0.795 | 0.223 | 0.985 | 0.992 | 0.632 |
| T=400 | 0.989 | 0.996 | 0.541 | 1 | 1 | 0.728 |
| T=1000 | 1 | 1 | 0.844 | 1 | 1 | 0.824 |

Tabella 5.6: Frequenze di rigetto dei test della varianza sotto modello AR con errori STARCH

# Capitolo 6

# Applicazione a serie reali

Le considerazioni che si possono tirare dalle simulazioni riportate nel capitolo precedente indicano come, nel momento in cui applichiamo i test considerati a serie storiche reali, non sia possibile pervenire sempre a conclusioni corrette. Se i test sulla media segnalano la presenza di media variabile nel tempo e quelli per la varianza no, si può affermare con sicurezza che la serie presa in esame ha media variabile nel tempo, la stessa cosa se si verifica la situazione contraria. Se invece i test segnalano la presenza sia di media sia di varianza variabili nel tempo è possibile solo affermare che la media sia effettivamente variabile, mentre non si possono trarre conclusioni sulla varianza.

Di seguito si applicano i test sulla media e varianza variabili nel tempo descritti nei capitoli precedenti a delle serie storiche finanziarie e se ne interpretano i risultati. Le serie storiche che sono state prese in considerazione sono l'indice S&P500, il PIL italiano, l'indice di produzione industriale italiano (IPI), l'indice dei prezzi a consumo italiano (CPI) e infine l'indice azionario della borsa milanese FSTE MIB. Le serie sono state tutte differenziate logaritmicamente. Le prime quattro serie storiche sono state ottenute dalla Federal Reserve Bank of St. Louis mentre l'FTSE MIB da Investing.com. I dati consistono in 1260 osservazioni giornaliere per lo S&P500 (4 aprile 2015-4 aprile 2020), 109 mensili per il PIL (gennaio 1995-gennaio 2022), 748 osservazioni mensili per l'IPI (gennaio 1960-aprile 2022), 749 per il CPI (gennaio 1960-maggio 2022) e 1763 dati giornalieri per FTSE MIB (4 agosto 2015-8 luglio 2022). Il numero di ritardi di ogni test è stato scelto utilizzando il crierio di informazione di Akaike (AIC).

| | $M_a$ | $M_b$ | $M_{wb1}$ | $M_{wb1}$ | $V_a$ | $V_b$ | $V_{wb}$ |
|---|---|---|---|---|---|---|---|
| S&P500 | 0.000 | 0.000 | 0.100 | 0.141 | 0.000 | 0.000 | 0.170 |
| PIL | 0.110 | 0.085 | 0.250 | 0.415 | 0.748 | 0.285 | 0.333 |
| IPI | 0.000 | 0.000 | 0.210 | 0.294 | 0.805 | 0.198 | 0.049 |
| CPI | 0.001 | 0.004 | 0.013 | 0.023 | 0.000 | 0.000 | 0.000 |
| FTSE MIB | 0.660 | 0.625 | 0.890 | 0.920 | 0.000 | 0.010 | 0.000 |

Tabella 6.1: p-value dei test di media e varianza applicati alle serie storiche reali

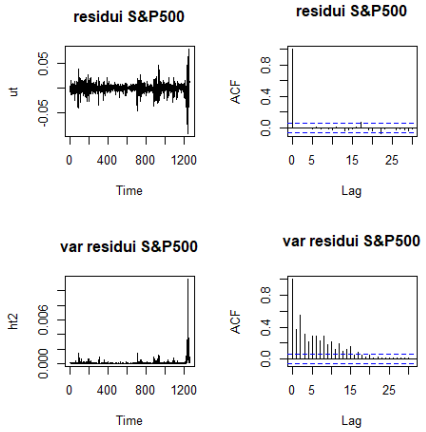


Figura 6.1: Grafico della serie e funzione di autocorrelazione dei residui e della varianza dei residui di S&P500

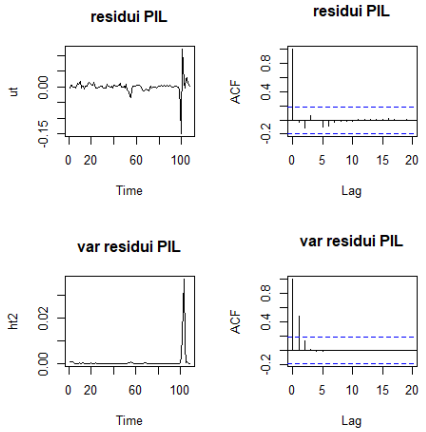


Figura 6.2: Grafico della serie e funzione di autocorrelazione dei residui e della varianza dei residui del PIL

Nella tabella 6.1 si possono osservare i p-value dei test presentati precedentemente e applicati alle serie sopra indicate. Per quanto riguarda lo S&P500, i p-value indicano la presenza di linearità nella media ed effetti ARCH variabili nel tempo. Come osservato nel capitolo precedente, $M_{wb1}$ e $M_{wb2}$ hanno performance maggiori in presenza di eteroschedasticità, indicando in modo più affidabile rispetto agli altri due test per la media se accettare o meno l'ipotesi nulla. Si può anche notare che $V_a$ e $V_b$ hanno potenza maggiore rispetto a $V_{wb}$, il quale in questo caso accetterebbe l'ipotesi nulla di linearità della varianza. La figura 6.1 riporta il grafico e la funzione di autocorrelazione dei residui e il grafico della varianza dei residui ARCH del modello lineare adattato ai dati di S&P500. Osservando in particolare la figura in basso a sinistra si nota che la varianza dei residui non è costante: si possono considerare affidabili i risultati osservati per $V_a$ e $V_b$, che rifiutano l'ipotesi di linearità, confermando ulteriormente la minore potenza del test $V_{wb}$ nell'individuare effetti ARCH variabili nel tempo. Osservando poi i risultati ottenuti per la serie del PIL si nota che tutti i test per la media accettano l'ipotesi nulla di linearità, così

come quelli per la varianza. Come evidenziato nella tabella 5.1, i test per media e varianza si comportano in modo adeguato in presenza di linearità della media ed omoschedasticità, dunque si può affermare con sicurezza questa ipotesi. Nella figura 6.2 si nota che la varianza dei residui rimane pressochè costante, ad eccezione per le ultime osservazioni in cui si registra un picco.

Passando alla serie IPI si verifica per la media la stessa situazione osservata per lo S&P500: $M_{wb1}$ e $M_{wb2}$ accettano in modo più deciso la linearità della media. Per quanto riguarda i test per la varianza, $V_a$ e $V_b$ accettano l'ipotesi nulla di linearità della varianza del termine d'errore, mentre $V_{wb}$ è prossimo al livello di significatività del 5%, il che indica la possibile presenza di effetti ARCH variabili nel tempo. Nonostante, come spiegato in precedenza, $V_{wb}$ risulti essere meno potente rispetto agli altri test sulla la varianza, anche la figura 6.3 evidenzia un leggero cambiamento nel tempo della varianza dei residui.

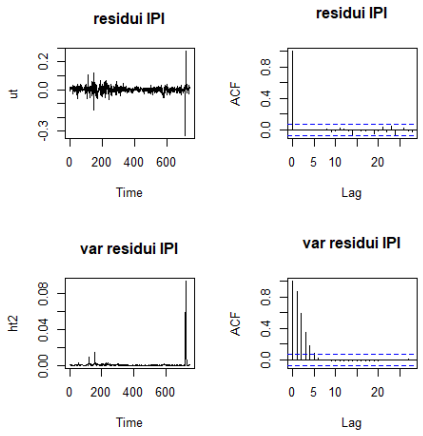


Figura 6.3: Grafico della serie e funzione di autocorrelazione dei residui e della varianza dei residui dell'IPI

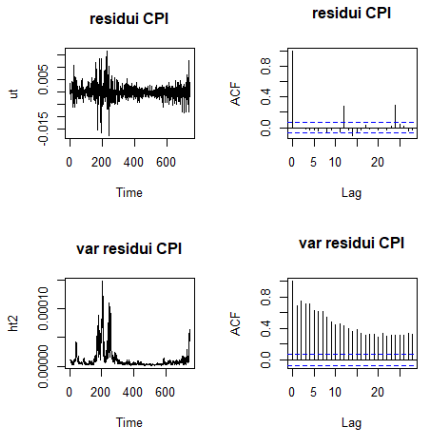


Figura 6.4: Grafico della serie e funzione di autocorrelazione dei residui e della varianza dei residui del CPI

Osserviamo ora i risultati relativi alla serie CPI, riportati in 6.1. Tutti i test indicano la presenza di media variabile nel tempo ed effetti ARCH variabili nel tempo. Come sottolineato in precedenza nella tabella 5.3, in presenza di media variabile nel tempo i test per la varianza portano a risultati spuri indicando la presenza di eteroschedasticità anche in sua assenza. Questa ipotesi viene, tuttavia, sostenuta dalla figura 6.4 che riporta il grafico della varianza dei residui, che evidenzia un chiaro cambiamento nel tempo.

Per quanto concerne infine l'FTSE MIB, i test per la media portano tutti all'accetta-

zione dell'ipotesi nulla di linearità mentre quelli della varianza indicano la presenza di effetti ARCH variabili nel tempo. Anche in questo caso l'ipotesi viene avvalorata dall'osservazione del grafico della varianza dei residui, come si può osservare nella figura 6.5

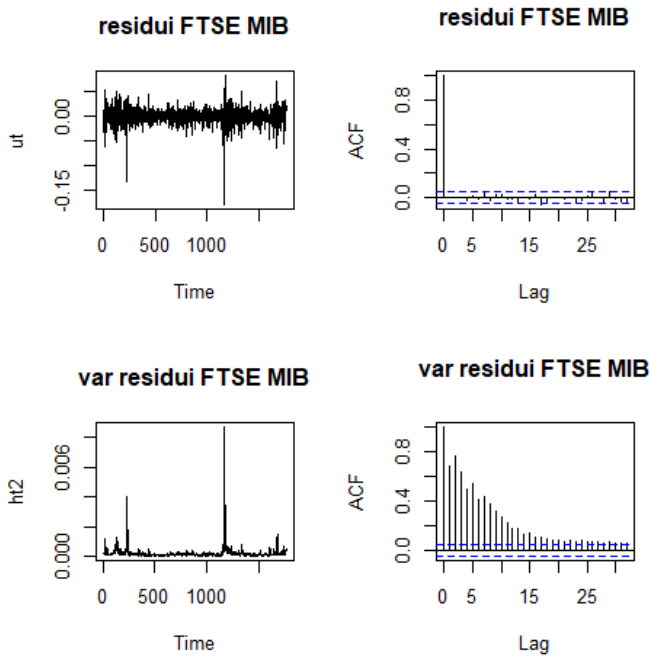


Figura 6.5: Grafico della serie e funzione di autocorrelazione dei residui e della varianza dei residui dell'FTSE MIB

# Conclusioni

In questo elaborato sono state analizzate le performance statistiche di test per media e varianza variabili nel tempo sotto media e varianza non correttamente specificate. Spesso nelle serie storiche finanziarie si osservano media e varianza variabili nel tempo, tuttavia non si può conoscere a priori se sono presenti. Da qui deriva la necessità che i test segnalino in modo accurato e tempestivo la presenza di media e/o varianza variabili nel tempo. Dalle simulazioni attuate si evince che i test asintotici per la media, in presenza di media variabile nel tempo, non hanno buone performace in quanto presentano delle differenze più o meno elevate fra i livelli empirici e il livello nominale, differenze che vengono però ridotte con l'utilizzo di test *wild bootstrap*. Questi ultimi si dimostrano robusti anche in presenza di effetti ARCH variabili nel tempo e non, consentendo di affermare con certezza la presenza o meno di media variabile nel tempo con varianza variabile nel tempo.
Il test asintotico e *bootstrap* per la varianza possiedono una potenza maggiore rispetto al test *wild bootstrap* nella segnalazione di varianza variabile nel tempo, dunque avvertono più rapidamente la presenza di effetti ARCH variabili e non variabili nel tempo. Questi test, tuttavia, non hanno sufficienti prestazioni in presenza di omoschedasticità e media variabile nel tempo, portando dunque a concludere che i test per la varianza variabile nel tempo non possano ritenersi affidabili.
Con ulteriori studi si potrebbe approfondire il comportamento di test per la varianza variabile nel tempo, in presenza di media condizionale non correttamente specificata, utilizzando test robusti.

# Bibliografia


Amado, C., Teräsvirta, T. (2013). Modelling volatility by variance decomposition. *Journal of Econometrics*, 175, 142-153.

Bisaglia, L., Gerolimetto, M. (2014). Testing for (non)linearity in economic time series: a Monte Carlo comparison. *Quaderni di Statistica*, 14, 5-22.

Balke, A. P., Kapetanios, G. (2007). Testing for ARCH in the presence of nonlinearity of unknown form in the conditional mean. *Journal of Econometrics*, 137, 472-488.

Box, G. and Jenkins, G. (1970), Time series analysis: forecasting and control. Holden Day, San Francisco.

Chan, F., MacAleer, M. (2001). Estimating smooth transition autoregressive models with GARCH errors in the presence of extreme observations and outliers. *ISER Discussion Paper*, 539, 1-28.

Chan, K. and Tong, H. (1986). On estimating thresholds in autoregressive models. *Journal of time Series Analysis*, 7, 178-190.

Dahlhaus, R., Rao, S. S. (2006). Statistical inference for time-varying ARCH processes. *The Annals of Statistics*, 34, 1075-1114.

Davies, R. B. (1977). Hypothesis testing when a nuisance parameter is present only under the alternative. *Biometrika*, 64, 247-254.

Davies, R. B. (1987). Hypothesis testing when a nuisance parameter is present only under the alternative. *Biometrika*, 74, 33-43.

Efron, B. (1979), Bootstrap methods: another look at the jackknife. *Annual Statistics*, 7, 1-26.

Engle, R. (1982). Autoregressive conditional heteroskedasticity with estimates of united kindom inflation. *Econometrica*, 50, 987-1008.

Gel, Y. R., Chen, B. (2012). Robust Lagrange multiplier test for detecting AR-

CH/GARCH effect using permutation and bootstrap. *The Canadian Journal of Statistics*, 40, 405-426.

Ghani, I. M. Md., Rahim, H. A. (2019). Modeling and Forecasting of Volatility using ARMA-GARCH: Case Study on Malaysia Natural Rubber Prices. *IOP Conf. Ser.: Mater. Sci. Eng.*, 548, 1-18.

Gonzalez Rivera, G. (1998). Smooth Transition GARCH Models. *Studies in Nonlinear Dynamics and Econometrics*, 3(2), 61–78.

Granger, C.W.J., Terasvirta, T. (1993). Modeling Nonlinear Economic Relationships. 479-480

Hesterberg, T. (2011). Bootstrap. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(6),497-526.

Kim, K. H., Kim, T. (2016). Capital asset pricing model: A time-varying volatility approach. *Journal of Empirical Finance*, 37, 268-281.

Kreiss J. P., Lahiri S. N. (2012), Bootstrap Methods for Time Series. *Time Series Analysis: Methods and Applications*, 30, 3-26.

Lee, J. Degennaro, R. P. (2000). Smooth Transition ARCH Models: Estimation and Testing. *Review of Quantitative Finance and Accounting*, 15, 5-20.

Liu, R. Y. (1988). Bootstrap procedure under some non-i.i.d, models. *Annals of Statistics*, 16(4), 1696-1708.

Lumsdaine, R. L., Ng, S. (1999). Testing for ARCH in the presence of a possibly misspecified conditonal mean. *Journal of Econometrics*, 93, 257-279.

Lundbergh, S., Teräsvirta, T., Van Dijk, V. (2003). Time-varying smooth transition autoregressive models. *Journal of Business  Economic Statistics*, 21, 104-121.

Luukkonen, R., Saikkonen, P., Teräsvirta, T. (1988a). Testing linearity against smooth transition autoregressive models. *Biometrika*, 70, 491-499.

Luukkonen, R., Saikkonen, P., Teräsvirta, T. (1988b). Testing linearity in univariate time series models. *Scandinavian Journal of Statistics*, 15, 161-175.

Maki, D., Ota, Y. (2021). Testing for Time-Varying Properties Under Misspecified Conditional Mean and Variance. *Computational Economics*, 57, 1167-1182.

Perron, P., Yamamoto, Y. (2019). Pitfalls of two-step testing for changes in the error variance and coefficients of a linear regression model. *Econometrics*, 7(2), 22, 1-11.

Pitarakis, J. Y. (2004). Least squares estimation and tests of breaks in mean and variance under misspecification. *Econometrics Journal*, 7, 32-54.

Teräsvirta, T. (1994). Specification, estimation, and evaluation of smooth transition autoregressive models. *Journal of the American Statistical Association*, 89, 208-218.

Tong, H. (1983). Threshold models in non-linear time series analysis. *Springer-Verlag*.

Tong, H. and Lim, K. (1980) Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society*, Series B, 42, 245-292

Van Dijk, D., Franses, P.H., and Lucas, A. (1999). Testing for smooth transition nonlinearity in the presence of outliers. *Journal of Business & Economic Statistics*, 17, 217-235.

Van Dijk, D., Teräsvirta, T. and Franses, P. (2002). Smooth transition autoregressive models - a survey of recent developments, *Econometric Reviews*, 21, 1-17.

Wu C.F.J. (1986), Jackknife, Bootstrap and other resampling methods in regression analysis. *The annals of statistics*, 14(4), 1261-1295.

# Appendice

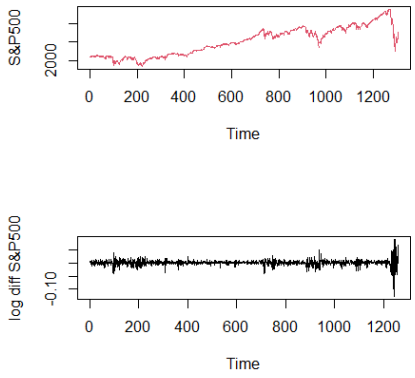


Figura 6: Grafico della serie S&P500 originale e differenziata logaritmicamente

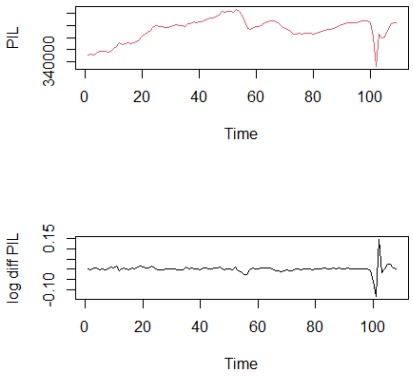


Figura 7: Grafico della serie PIL originale e differenziata logaritmicamente

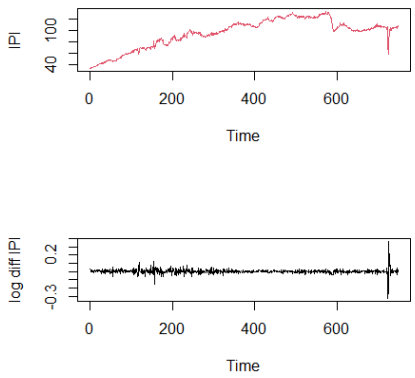


Figura 8: Grafico della serie IPI originale e differenziata logaritmicamente

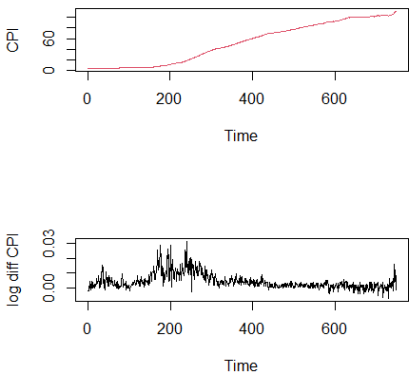


Figura 9: Grafico della serie CPI originale e differenziata logaritmicamente

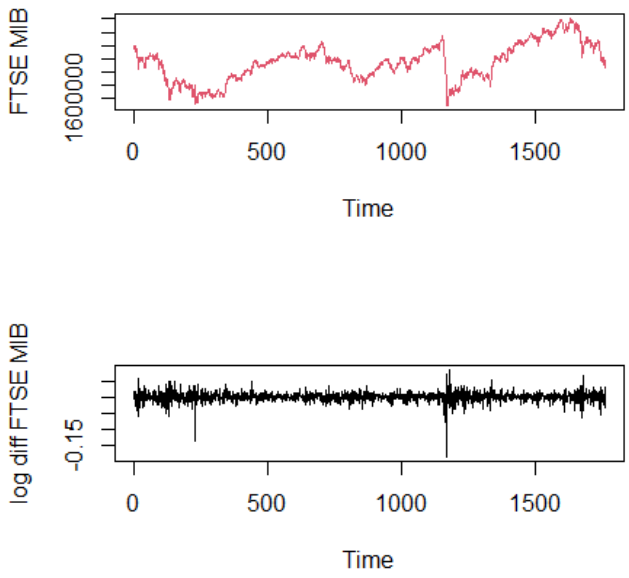


Figura 10: Grafico della serie FTSE MIB originale e differenziata logaritmicamente

Codici di R delle simulazioni:

```
library(tseries)
library(aod)
library(sarima)
library(rugarch)
source("sse.lib")
Ma tabella 1:
count=0
for (i in 1:10000){
  ts.sim20=arima.sim(n=n+1,list(order=c(1,0,0),ar=b0))
  ts.sim20=ts.sim20+(1/(1-b0))
  YY=embed(ts.sim20,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n)
  mod4=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  if(Ma$result$chi2[3]<0.05){
```

```
    count=count+1
  }
}
count/10000
Mb tabella 1
count=0
for (i in 1:1000){
  ts.sim20=arima.sim(n=n+2,list(order=c(1,0,0),ar=b0))
  ts.sim20=ts.sim20+(1/(1-b0))
  YY=embed(ts.sim20,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n)
  mod20=lm(yt~yt1)
  res=as.numeric(residuals(mod20))-mean(residuals(mod20))
  #bootstrap 1
  bpar=function(n){
    ystar0=coef(mod20)[1]+sample(res,1,replace=T)
    ystar1=coef(mod20)[1]+coef(mod20)[2]*ystar0+sample(res,1,replace=T)
    ytstar=rep(0,n+1)
    ytstar[1]=ystar0
    ytstar[2]=ystar1
    for (i in 3:(n+1)) {
      ytstar[i]=coef(mod20)[1]+coef(mod20)[2]*ytstar[i-1]+sample(res,1,replace=T)
    }
    yyst=embed(ytstar,2)
    ytstar=yyst[,1]
    ytstar1=yyst[,2]
    mod4=lm(ytstar~ytstar1+tempo+tempo:ytstar1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
```

```
  Mwb=sapply(rep(n,200),bpar)
  yt=yt[2:(n+1)]
  yt1=yt1[2:(n+1)]
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Mwb1 tabella 1
count=0
for (i in 1:1000){
  ts.sim20=arima.sim(n=n+2,list(order=c(1,0,0),ar=b0))
  ts.sim20=ts.sim20+(1/(1-b0))
  YY=embed(ts.sim20,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n)
  mod20=lm(yt~yt1)
  #bootstrap 1
  bpar=function(n){
    ystar0=coef(mod20)[1]+rnorm(1)
    ystar1=coef(mod20)[1]+coef(mod20)[2]*ystar0+
    as.numeric(residuals(mod20)[1])*rnorm(1)
    ytstar=rep(0,n+1)
    ytstar[1]=ystar0
    ytstar[2]=ystar1
    for (i in 3:(n+1)) {
      ytstar[i]=coef(mod20)[1]+coef(mod20)[2]*ytstar[i-1]+
```

```
      as.numeric(residuals(mod20)[i])*rnorm(1)
    }
    yyst=embed(ytstar,2)
    ytstar=yyst[,1]
    ytstar1=yyst[,2]
    mod4=lm(ytstar~ytstar1+tempo+tempo:ytstar1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
  Mwb=sapply(rep(n,200),bpar)
  yt=yt[2:(n+1)]
  yt1=yt1[2:(n+1)]
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
   count=count+1
  }
  }
count/1000
Mwb2 tabella 1
count=0
for (i in 1:1000){
  ts.sim20=arima.sim(n=n+2,list(order=c(1,0,0),ar=b0))
  ts.sim20=ts.sim20+(1/(1-b0))
  YY=embed(ts.sim20,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n+1)
  mod20=lm(yt~yt1)
  #bootstrap 1
```

```
  bpar=function(n){
    ytstar=coef(mod20)[1]+coef(mod20)[2]*yt1+
    as.numeric(residuals(mod20))*rnorm(n+1)
    mod4=lm(ytstar~yt1+tempo+tempo:yt1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
  Mwb=sapply(rep(n,200),bpar)
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Va tabella 1
count=0
for (i in 1:10000){
  ts.sim20=arima.sim(n=n+2,list(order=c(1,0,0),ar=b0))
  ts.sim20=ts.sim20+(1/(1-b0))
  YY=embed(ts.sim20,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n)
  yt.20=lm(yt~yt1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
  ht2<- sigma(fitsim)^2
  ut <- yt.20$residuals
```

```
  uu=embed(ut,2)
  ut1=uu[,2]^2
  ht2=ht2[1:n]
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  if(Va>qf(0.95,2,(n-4))){
    count=count+1
  }
}
count/10000
Vb tabella 1
count=0
for (i in 1:1000){
  ts.sim20=arima.sim(n=n+2,list(order=c(1,0,0),ar=b0))
  ts.sim20=ts.sim20+(1/(1-b0))
  YY=embed(ts.sim20,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n)
  mod10=lm(yt~yt1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt, solver = 'hybrid')
  htsq<- sigma(fitsim)^2
  ut <- mod10$residuals
  uu=embed(ut,2)
  ut1=uu[,2]^2
  ht2=htsq[1:n]
```

```
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)
  #bootstrap 1
  bpar=function(n){
    htstar=coef(modh0)+sample(as.numeric(modh0$residuals),n,replace=F)
    modhh0=lm(htstar~1)
    modhh1=lm(htstar~ut1+tempo+tempo:ut1)
    #statistica Va (16)
    SSRhh0=sum(modhh0$residuals^2)
    SSRhh1=sum(modhh1$residuals^2)
    Va=((SSRhh0-SSRhh1)/(2+1))/(SSRhh1/(n-(2+2)))
    Va
  }
  Vb=sapply(rep(n,200),bpar)
   #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  I=length(which(Vb>Va))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Vwb tabella 1
count=0
for (i in 1:1000){
  yt=rep(0,n+2)
  for(i in 2:(n+2)){
    yt[i]=a0+b0*yt[i-1]+rnorm(1)
  }
```

```
yy=embed(yt,2)
yt1=yy[,2]
yt=yy[,1]
tempo=seq(1,n)
mod10=lm(yt~yt1)
uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                 mean.model = list(armaOrder=c(1,0)))
fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
htsq<- sigma(fitsim)^2
ut <- mod10$residuals
uu=embed(ut,2)
ut1=uu[,2]^2
ht2=htsq[1:n]
modh0=lm(ht2~1)
modh1=lm(ht2~ut1+tempo+tempo:ut1)
#bootstrap 1
bpar=function(n){
  htstar=rep(0,n)
  for (i in 1:n) {
    htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
  }
  modh0=lm(htstar~1)
  modh1=lm(htstar~ut1+tempo+tempo:ut1)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  Va
}
Vb=sapply(rep(n,200),bpar)
#statistica Va (16)
SSRh0=sum(modh0$residuals^2)
```

```
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  I=length(which(Vb>Va))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Ma tabella 2
count=0
for (i in 1:10000){
  yt.21=rep(0,n+1)
  for(i in 2:(n+1)){
    yt.21[i]=1+0.3*yt.21[i-1]+
    (a1+b1*yt.21[i-1])*(plogis(i,n/2,1/ghamma)-1/2)+rnorm(1)
  }
  YY=embed(yt.21,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n)
  mod4=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  if(Ma$result$chi2[3]<0.05){
    count=count+1
  }
}
count/100000
Mb tabella 2
for (i in 1:1000){
  yt.21=rep(0,n+2)
  y0=1+rnorm(1)
```

```
y1=1+0.3*y0+rnorm(1)
yt.21[1]=y0
yt.21[2]=y1
for(j in 3:(n+2)){
  yt.21[j]=1+0.3*yt.21[j-1]+
  (a1+b1*yt.21[j-1])*(plogis(j,n/2,1/ghamma)-1/2)+rnorm(1)
}
YY=embed(yt.21,2)
yt=YY[,1]
yt1=YY[,2]
tempo=seq(1,n)
mod20=lm(yt~yt1)
res=as.numeric(residuals(mod20))-mean(residuals(mod20))
#bootstrap 1
bpar=function(n){
  ystar0=coef(mod20)[1]+sample(res,1,replace=T)
  ystar1=coef(mod20)[1]+coef(mod20)[2]*ystar0+
  sample(res,1,replace=T)
  ytstar=rep(0,n+1)
  ytstar[1]=ystar0
  ytstar[2]=ystar1
  for (i in 3:(n+1)) {
    ytstar[i]=coef(mod20)[1]+coef(mod20)[2]*ytstar[i-1]+
    sample(res,1,replace=T)
  }
  yyst=embed(ytstar,2)
  ytstar=yyst[,1]
  ytstar1=yyst[,2]
  mod4=lm(ytstar~ytstar1+tempo+tempo:ytstar1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  Ma$result$chi2[1]
}
```

```
  Mwb=sapply(rep(n,200),bpar)
  yt=yt[2:(n+1)]
  yt1=yt1[2:(n+1)]
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Mwb1 tabella 2
count=0
for (i in 1:1000){
  yt.21=rep(0,n+2)
  y0=1+rnorm(1)
  y1=1+0.3*y0+rnorm(1)
  yt.21[1]=y0
  yt.21[2]=y1
  for(j in 3:(n+2)){
    yt.21[j]=1+0.3*yt.21[j-1]+
    (a1+b1*yt.21[j-1])*(plogis(j,n/2,1/ghamma)-1/2)+rnorm(1)
  }
  YY=embed(yt.21,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n)
  mod20=lm(yt~yt1)
  bpar=function(n){
    ystar0=coef(mod20)[1]+rnorm(1)
    ystar1=coef(mod20)[1]+coef(mod20)[2]*ystar0+
```

```
    as.numeric(residuals(mod20)[1])*rnorm(1)
    ytstar=rep(0,n+1)
    ytstar[1]=ystar0
    ytstar[2]=ystar1
    for (i in 3:(n+1)) {
      ytstar[i]=coef(mod20)[1]+coef(mod20)[2]*ytstar[i-1]+
      as.numeric(residuals(mod20)[i])*rnorm(1)
    }
    yyst=embed(ytstar,2)
    ytstar=yyst[,1]
    ytstar1=yyst[,2]
    mod4=lm(ytstar~ytstar1+tempo+tempo:ytstar1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
  Mwb=sapply(rep(n,200),bpar)
  yt=yt[2:(n+1)]
  yt1=yt1[2:(n+1)]
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Mwb2 tabella 2
count=0
for (i in 1:1000){
  yt.21=rep(0,n+2)
  y0=1+rnorm(1)
```

```
  y1=1+0.3*y0+rnorm(1)
  yt.21[1]=y0
  yt.21[2]=y1
  for(j in 3:(n+2)){
    yt.21[j]=1+0.3*yt.21[j-1]+
    (a1+b1*yt.21[j-1])*(plogis(j,n/2,1/ghamma)-1/2)+rnorm(1)
  }
  YY=embed(yt.21,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n+1)
  mod20=lm(yt~yt1)
  #bootstrap 1
  bpar=function(n){
    ytstar=coef(mod20)[1]+coef(mod20)[2]*yt1+
    as.numeric(residuals(mod20))*rnorm(n+1)
    mod4=lm(ytstar~yt1+tempo+tempo:yt1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
  Mwb=sapply(rep(n,200),bpar)
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Va tabella 2
count=0
```

```
for (i in 1:10000){
  yt.21=rep(0,n+2)
  yt.21[1]=1+a1*plogis(1,n/2,1/ghamma)+rnorm(1)
  for(i in 2:(n+1)){
    yt.21[i]=1+0.3*yt.21[i-1]+
    (a1+b1*yt.21[i-1])*(plogis(i,n/2,1/ghamma)-1/2)+rnorm(1)
  }
  YY=embed(yt.21,2)
  yt=YY[,1]
  yt1=YY[,2]
  tempo=seq(1,n)
  mod10=lm(yt~yt1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
  ht2<- sigma(fitsim)^2
  ut <- mod10$residuals
  uu=embed(ut,2)
  ut1=uu[,2]^2
  ht2=ht2[1:n]
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  if(Va>qf(0.95,2,(n-4))){
    count=count+1
  }
}
count/10000
Vb tabella 2
```

```
count=0
for (i in 1:1000){
  yt.21=rep(0,n+2)
  for(i in 2:(n+2)){
    yt.21[i]=1+0.3*yt.21[i-1]+
    (a1+b1*yt.21[i-1])*(plogis(i,n/2,1/ghamma)-1/2)+rnorm(1)
  }
  yy=embed(yt.21,2)
  yt1=yy[,2]
  yt=yy[,1]
  tempo=seq(1,n)
  mod10=lm(yt~yt1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
  htsq<- sigma(fitsim)^2
  ut <- mod10$residuals
  uu=embed(ut,2)
  ut1=uu[,2]^2
  ht2=htsq[1:n]
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)
  #bootstrap
  bpar=function(n){
    htstar=coef(modh0)+sample(as.numeric(modh0$residuals),n,replace=T)
    modhh0=lm(htstar~1)
    modhh1=lm(htstar~ut1+tempo+tempo:ut1)
    #statistica Va (16)
    SSRh0=sum(modhh0$residuals^2)
    SSRh1=sum(modhh1$residuals^2)
    Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
    Va
```

```
  }
  Vb=sapply(rep(n,200),bpar)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  I=length(which(Vb>Va))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Vwb tabella 2
count=0
for (i in 1:1000){
  yt.21=rep(0,n+2)
  for(j in 2:(n+2)){
    yt.21[j]=1+0.3*yt.21[j-1]+
    (a1+b1*yt.21[j-1])*(plogis(j,n/2,1/ghamma)-1/2)+rnorm(1)
  }
  yy=embed(yt.21,2)
  yt1=yy[,2]
  yt=yy[,1]
  tempo=seq(1,n)
  mod10=lm(yt~yt1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
  htsq<- sigma(fitsim)^2
  ut <- mod10$residuals
  uu=embed(ut,2)
```

```
  ut1=uu[,2]^2
  ht2=htsq[1:n]
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)
  #bootstrap 1
  bpar=function(n){
    htstar=rep(0,n)
    for (i in 1:n) {
      htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
    }
    modhh0=lm(htstar~1)
    modhh1=lm(htstar~ut1+tempo+tempo:ut1)
    #statistica Va (16)
    SSRh0=sum(modhh0$residuals^2)
    SSRh1=sum(modhh1$residuals^2)
    Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
    Va
  }
  Vb=sapply(rep(n,200),bpar)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  I=length(which(Vb>Va))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Ma tabella 3
count=0
```

```
for (i in 1:10000){
  epsilon=rnorm(n+2)
  u=rep(0,n+2)
  yt.23=rep(0,n+2)
  hsquared=rep(0,n+2)
  for(i in 2:(n+2)){
    hsquared[i]=a0+b0*(u[i-1]^2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+2)]
  yy=embed(yt.23,2)
  yt1=yy[,2]
  yt=yy[,1]
  tempo=seq(1,n)
  mod4=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  if(Ma$result$chi2[3]<0.05){
    count=count+1
  }
}
count/10000
Mb tabella 3
count=0
for (i in 1:1000){
  epsilon=rnorm(n+2)
  u=rep(0,n+2)
  yt.23=rep(0,n+2)
  hsquared=rep(0,n+2)
  for(i in 2:(n+2)){
    hsquared[i]=a0+b0*(u[i-1]^2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
```

```
  yt.23[i]=1+0.3*yt.23[i-1]+u[i]
}
yt.23=yt.23[2:(n+2)]
yy=embed(yt.23,2)
yt=yy[,1]
yt1=yy[,2]
tempo=seq(1,n)
mod20=lm(yt~yt1)
res=as.numeric(residuals(mod20))-mean(residuals(mod20))
#bootstrap 1
bpar=function(n){
  ystar0=coef(mod20)[1]+sample(res,1,replace=T)
  ystar1=coef(mod20)[1]+coef(mod20)[2]*ystar0+sample(res,1,replace=T)
  ytstar=rep(0,n+1)
  ytstar[1]=ystar0
  ytstar[2]=ystar1
  for (i in 3:(n+1)) {
    ytstar[i]=coef(mod20)[1]+coef(mod20)[2]*ytstar[i-1]+
    sample(res,1,replace=T)
  }
  yyst=embed(ytstar,2)
  ytstar=yyst[,1]
  ytstar1=yyst[,2]
  mod4=lm(ytstar~ytstar1+tempo+tempo:ytstar1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  Ma$result$chi2[1]
}
Mwb=sapply(rep(n,200),bpar)
yt=yt[2:(n+1)]
yt1=yt1[2:(n+1)]
mod44=lm(yt~yt1+tempo+tempo:yt1)
Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
```

```
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Mwb1 tabella 3
count=0
for (i in 1:1000){
  epsilon=rnorm(n+2)
  u=rep(0,n+2)
  yt.23=rep(0,n+2)
  hsquared=rep(0,n+2)
  for(j in 2:(n+2)){
    hsquared[j]=a0+b0*(u[j-1]^2)
    u[j]=epsilon[j]*sqrt(hsquared[j])
    yt.23[j]=1+0.3*yt.23[j-1]+u[j]
  }
  yt.23=yt.23[2:(n+2)]
  yy=embed(yt.23,2)
  yt=yy[,1]
  yt1=yy[,2]
  tempo=seq(1,n)
  mod20=lm(yt~yt1)
  bpar=function(n){
    ystar0=coef(mod20)[1]+rnorm(1)
    ystar1=coef(mod20)[1]+coef(mod20)[2]*ystar0+
    as.numeric(residuals(mod20)[1])*rnorm(1)
    ytstar=rep(0,n+1)
    ytstar[1]=ystar0
    ytstar[2]=ystar1
```

```
    for (i in 3:(n+1)) {
      ytstar[i]=coef(mod20)[1]+coef(mod20)[2]*ytstar[i-1]+
      as.numeric(residuals(mod20)[i])*rnorm(1)
    }
    yyst=embed(ytstar,2)
    ytstar=yyst[,1]
    ytstar1=yyst[,2]
    mod4=lm(ytstar~ytstar1+tempo+tempo:ytstar1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
  Mwb=sapply(rep(n,200),bpar)
  yt=yt[2:(n+1)]
  yt1=yt1[2:(n+1)]
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Mwb2 tabella 3
count=0
for (i in 1:1000){
  epsilon=rnorm(n+2)
  u=rep(0,n+2)
  yt.23=rep(0,n+2)
  hsquared=rep(0,n+2)
  for(i in 2:(n+2)){
    hsquared[i]=a0+b0*(u[i-1]^2)
```

```
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+2)]
  yy=embed(yt.23,2)
  yt=yy[,1]
  yt1=yy[,2]
  tempo=seq(1,n+1)
  mod20=lm(yt~yt1)
  #bootstrap 1
  bpar=function(n){
    ytstar=coef(mod20)[1]+coef(mod20)[2]*yt1+
    as.numeric(residuals(mod20))*rnorm(n+1)
    mod4=lm(ytstar~yt1+tempo+tempo:yt1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
  Mwb=sapply(rep(n,200),bpar)
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Va tabella 3
count=0
for (i in 1:10000){
  epsilon=rnorm(n+3)
  u=rep(0,n+3)
```

```
  yt.23=rep(0,n+3)
  hsquared=rep(0,n+3)
  for(i in 2:(n+3)){
    hsquared[i]=a0+b0*(u[i-1]^2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+3)]
  yy=embed(yt.23,2)
  yt.23.1=yy[,2]
  yt.23=yy[,1]
  tempo=seq(1,n)
  mod10=lm(yt.23~yt.23.1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt.23,solver = 'hybrid')
  ht2<- sigma(fitsim)^2
  ut <- mod10$residuals
  uu=embed(ut,2)
  ut1=uu[,2]^2
  ht2=ht2[1:n]
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  if(Va>qf(0.95,2,(n-4))){
    count=count+1
  }
}
count/10000
```

```
Vb tabella 3
count=0
for (i in 1:1000){
  epsilon=rnorm(n+3)
  u=rep(0,n+3)
  yt.23=rep(0,n+3)
  hsquared=rep(0,n+3)
  for(i in 2:(n+3)){
    hsquared[i]=a0+b0*(u[i-1]^2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+3)]
  yy=embed(yt.23,2)
  yt1=yy[,2]
  yt=yy[,1]
  tempo=seq(1,n)
  mod10=lm(yt~yt1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
  htsq<- sigma(fitsim)^2
  ut <- mod10$residuals
  uu=embed(ut,2)
  ut1=uu[,2]^2
  ht2=htsq[1:n]
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)
  #bootstrap
  bpar=function(n){
    htstar=coef(modh0)+sample(as.numeric(modh0$residuals),n,replace=T)
    modh0=lm(htstar~1)
```

```
    modh1=lm(htstar~ut1+tempo+tempo:ut1)
    #statistica Va (16)
    SSRh0=sum(modh0$residuals^2)
    SSRh1=sum(modh1$residuals^2)
    Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
    Va
  }
  Vb=sapply(rep(n,200),bpar)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  I=length(which(Vb>Va))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Vwb tabella 3
count=0
for (i in 1:1000){
  epsilon=rnorm(n+3)
  u=rep(0,n+3)
  yt.23=rep(0,n+3)
  hsquared=rep(0,n+3)
  for(i in 2:(n+3)){
    hsquared[i]=a0+b0*(u[i-1]^2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+3)]
```

```
yy=embed(yt.23,2)
yt1=yy[,2]
yt=yy[,1]
tempo=seq(1,n)
mod10=lm(yt~yt1)
uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                 mean.model = list(armaOrder=c(1,0)))
fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
htsq<- sigma(fitsim)^2
ut <- mod10$residuals
uu=embed(ut,2)
ut1=uu[,2]^2
ht2=htsq[1:n]
modh0=lm(ht2~1)
modh1=lm(ht2~ut1+tempo+tempo:ut1)
#bootstrap 1
bpar=function(n){
  htstar=rep(0,n)
  for (i in 1:n) {
    htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
  }
  modh0=lm(htstar~1)
  modh1=lm(htstar~ut1+tempo+tempo:ut1)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  Va
}
Vb=sapply(rep(n,200),bpar)
#statistica Va (16)
SSRh0=sum(modh0$residuals^2)
```

```
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  I=length(which(Vb>Va))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Ma tabella 4
count=0
for (i in 1:10000){
  epsilon=rnorm(n+2)
  u=rep(0,n+2)
  yt.23=rep(0,n+2)
  hsquared=rep(0,n+2)
  for(i in 2:(n+2)){
    hsquared[i]=1+0.3*(u[i-1]^2)+
    (a1+b1*(u[i-1]^2))*(plogis(i,n/2,1/ghamma)-1/2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+2)]
  yy=embed(yt.23,2)
  yt1=yy[,2]
  yt=yy[,1]
  tempo=seq(1,n)
  mod4=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  if(Ma$result$chi2[3]<0.05){
    count=count+1
  }
```

```
}
count/10000
Mb tabella 4
count=0
for (i in 1:1000){
  epsilon=rnorm(n+2)
  u=rep(0,n+2)
  yt.23=rep(0,n+2)
  hsquared=rep(0,n+2)
  for(i in 2:(n+2)){
    hsquared[i]=1+0.3*(u[i-1]^2)+
    (a1+b1*(u[i-1]^2))*(plogis(i,n/2,1/ghamma)-1/2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+2)]
  yy=embed(yt.23,2)
  yt=yy[,1]
  yt1=yy[,2]
  tempo=seq(1,n)
  mod20=lm(yt~yt1)
  res=as.numeric(residuals(mod20))-mean(residuals(mod20))
  #bootstrap 1
  bpar=function(n){
    ystar0=coef(mod20)[1]+sample(res,1,replace=T)
    ystar1=coef(mod20)[1]+coef(mod20)[2]*ystar0+sample(res,1,replace=T)
    ytstar=rep(0,n+1)
    ytstar[1]=ystar0
    ytstar[2]=ystar1
    for (i in 3:(n+1)) {
      ytstar[i]=coef(mod20)[1]+coef(mod20)[2]*ytstar[i-1]+
      sample(res,1,replace=T)
```

```
    }
    yyst=embed(ytstar,2)
    ytstar=yyst[,1]
    ytstar1=yyst[,2]
    mod4=lm(ytstar~ytstar1+tempo+tempo:ytstar1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
  Mwb=sapply(rep(n,200),bpar)
  yt=yt[2:(n+1)]
  yt1=yt1[2:(n+1)]
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Mwb1 tabella 4
count=0
for (i in 1:1000){
  epsilon=rnorm(n+2)
  u=rep(0,n+2)
  yt.23=rep(0,n+2)
  hsquared=rep(0,n+2)
  for(i in 2:(n+2)){
    hsquared[i]=1+0.3*(u[i-1]^2)+
    (a1+b1*(u[i-1]^2))*(plogis(i,n/2,1/ghamma)-1/2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
```

```
}
yt.23=yt.23[2:(n+2)]
yy=embed(yt.23,2)
yt=yy[,1]
yt1=yy[,2]
tempo=seq(1,n)
mod20=lm(yt~yt1)
bpar=function(n){
  ystar0=coef(mod20)[1]+rnorm(1)
  ystar1=coef(mod20)[1]+coef(mod20)[2]*ystar0+
  as.numeric(residuals(mod20)[1])*rnorm(1)
  ytstar=rep(0,n+1)
  ytstar[1]=ystar0
  ytstar[2]=ystar1
  for (i in 3:(n+1)) {
    ytstar[i]=coef(mod20)[1]+coef(mod20)[2]*ytstar[i-1]+
    as.numeric(residuals(mod20)[i])*rnorm(1)
  }
  yyst=embed(ytstar,2)
  ytstar=yyst[,1]
  ytstar1=yyst[,2]
  mod4=lm(ytstar~ytstar1+tempo+tempo:ytstar1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  Ma$result$chi2[1]
}
Mwb=sapply(rep(n,200),bpar)
yt=yt[2:(n+1)]
yt1=yt1[2:(n+1)]
mod44=lm(yt~yt1+tempo+tempo:yt1)
Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
I=length(which(Mwb>Ma))
pval=I/200
```

```
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Mwb2 tabella 4
count=0
for (i in 1:1000){
  epsilon=rnorm(n+2)
  u=rep(0,n+2)
  yt.23=rep(0,n+2)
  hsquared=rep(0,n+2)
  for(i in 2:(n+2)){
    hsquared[i]=1+0.3*(u[i-1]^2)+
    (a1+b1*(u[i-1]^2))*(plogis(i,n/2,1/ghamma)-1/2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+2)]
  yy=embed(yt.23,2)
  yt=yy[,1]
  yt1=yy[,2]
  tempo=seq(1,n)
  mod20=lm(yt~yt1)
  #bootstrap 1
  bpar=function(n){
    ytstar=coef(mod20)[1]+coef(mod20)[2]*yt1+
    as.numeric(residuals(mod20))*rnorm(n)
    mod4=lm(ytstar~yt1+tempo+tempo:yt1)
    Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
    Ma$result$chi2[1]
  }
```

```
  Mwb=sapply(rep(n,200),bpar)
  mod44=lm(yt~yt1+tempo+tempo:yt1)
  Ma=wald.test(Sigma=vcov(mod44),b=coef(mod44),Terms=3:4)$result$chi2[1]
  I=length(which(Mwb>Ma))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Va tabella 4
count=0
for (i in 1:10000){
  epsilon=rnorm(n+3)
  u=rep(0,n+3)
  yt.23=rep(0,n+3)
  hsquared=rep(0,n+3)
  for(i in 2:(n+3)){
    hsquared[i]=1+0.3*(u[i-1]^2)+
    (a1+b1*(u[i-1]^2))*(plogis(i,n/2,1/ghamma)-1/2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+3)]
  yy=embed(yt.23,2)
  yt.23.1=yy[,2]
  yt.23=yy[,1]
  tempo=seq(1,n)
  mod10=lm(yt.23~yt.23.1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt.23,solver='hybrid')
```

```
  ht2<- sigma(fitsim)^2
  ut <- mod10$residuals
  uu=embed(ut,2)
  ut1=uu[,2]^2
  ht2=ht2[1:n]
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  if(Va>qf(0.95,2,(n-4))){
    count=count+1
  }
}
count/10000
Vb tabella 4
count=0
for (i in 1:1000){
  epsilon=rnorm(n+3)
  u=rep(0,n+3)
  yt.23=rep(0,n+3)
  hsquared=rep(0,n+3)
  for(i in 2:(n+3)){
    hsquared[i]=1+0.3*(u[i-1]^2)+
    (a1+b1*(u[i-1]^2))*(plogis(i,n/2,1/ghamma)-1/2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+3)]
  yy=embed(yt.23,2)
  yt1=yy[,2]
```

```
yt=yy[,1]
tempo=seq(1,n)
mod10=lm(yt~yt1)
uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                 mean.model = list(armaOrder=c(1,0)))
fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
htsq<- sigma(fitsim)^2
ut <- mod10$residuals
uu=embed(ut,2)
ut1=uu[,2]^2
ht2=htsq[1:n]
modh0=lm(ht2~1)
modh1=lm(ht2~ut1+tempo+tempo:ut1)

#bootstrap
bpar=function(n){
  htstar=coef(modh0)+sample(as.numeric(modh0$residuals),n,replace=T)
  modh0=lm(htstar~1)
  modh1=lm(htstar~ut1+tempo+tempo:ut1)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  Va
}
Vb=sapply(rep(n,200),bpar)
#statistica Va (16)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
I=length(which(Vb>Va))
pval=I/200
```

```
  if (pval<0.05){
    count=count+1
  }
}
count/1000
Vwb tabella 4
count=0
for (i in 1:1000){
  epsilon=rnorm(n+3)
  u=rep(0,n+3)
  yt.23=rep(0,n+3)
  hsquared=rep(0,n+3)
  for(i in 2:(n+3)){
    hsquared[i]=1+0.3*(u[i-1]^2)+
    (a1+b1*(u[i-1]^2))*(plogis(i,n/2,1/ghamma)-1/2)
    u[i]=epsilon[i]*sqrt(hsquared[i])
    yt.23[i]=1+0.3*yt.23[i-1]+u[i]
  }
  yt.23=yt.23[2:(n+3)]
  yy=embed(yt.23,2)
  yt1=yy[,2]
  yt=yy[,1]
  tempo=seq(1,n)
  mod10=lm(yt~yt1)
  uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                   mean.model = list(armaOrder=c(1,0)))
  fitsim=ugarchfit(spec=uspec,data=yt,solver='hybrid')
  htsq<- sigma(fitsim)^2
  ut <- mod10$residuals
  uu=embed(ut,2)
  ut1=uu[,2]^2
  ht2=htsq[1:n]
```

```
  modh0=lm(ht2~1)
  modh1=lm(ht2~ut1+tempo+tempo:ut1)

  #bootstrap 1
  bpar=function(n){
    htstar=rep(0,n)
    for (i in 1:n) {
      htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
    }
    modh0=lm(htstar~1)
    modh1=lm(htstar~ut1+tempo+tempo:ut1)
    #statistica Va (16)
    SSRh0=sum(modh0$residuals^2)
    SSRh1=sum(modh1$residuals^2)
    Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
    Va
  }
  Vb=sapply(rep(n,200),bpar)
    #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2+1))/(SSRh1/(n-(2+2)))
  I=length(which(Vb>Va))
  pval=I/200
  if (pval<0.05){
    count=count+1
  }
}
count/1000
```

Codice R delle applicazioni

S&P500:

```
    sp=read.csv("SP500.csv")
```

```
s=as.numeric(sp[1:1306,2])
serie=ts(diff(log(na.omit(s))))
par(mfrow=c(2,1))
plot(ser,ylab="S&P500",col=2)
plot(serie, ylab="log diff S&P500")

library(vars)
VARselect(serie)
emb=embed(serie,10)
y=emb[,1]
y1=emb[,2]
y2=emb[,3]
y3=emb[,4]
y4=emb[,5]
y5=emb[,6]
y6=emb[,7]
y7=emb[,8]
y8=emb[,9]
y9=emb[,10]
tempo=seq(1,length(serie)-9)
fit9=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+y9+tempo+tempo:y1+tempo:y2+tempo:y3+
tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo+y9:tempo)
#ma
Ma=wald.test(Sigma=vcov(fit9),b=coef(fit9),Terms=11:20)
Ma
fit20=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+y9)

t=length(y)
#Mb
res=as.numeric(residuals(fit20)-mean(residuals(fit20)))
bpar3=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
```

```
ystar1=coef(fit20)[1]+coef(fit20)[2]*ystar0+sample(res,1,replace=T)
ystar2=coef(fit20)[1]+coef(fit20)[2]*ystar1+coef(fit20)[3]*ystar0+
sample(res,1,replace=T)
ystar3=coef(fit20)[1]+coef(fit20)[2]*ystar2+coef(fit20)[3]*ystar1+
  coef(fit20)[4]*ystar0+sample(res,1,replace=T)
ystar4=coef(fit20)[1]+coef(fit20)[2]*ystar3+coef(fit20)[3]*ystar2+
  coef(fit20)[4]*ystar1+coef(fit20)[5]*ystar0+sample(res,1,replace=T)
ystar5=coef(fit20)[1]+coef(fit20)[2]*ystar4+coef(fit20)[3]*ystar3+
coef(fit20)[4]*ystar2+
  coef(fit20)[5]*ystar1+coef(fit20)[6]*ystar0+sample(res,1,replace=T)
ystar6=coef(fit20)[1]+coef(fit20)[2]*ystar5+coef(fit20)[3]*ystar4+
coef(fit20)[4]*ystar3+coef(fit20)[5]*ystar2+
  coef(fit20)[6]*ystar1+coef(fit20)[7]*ystar0+sample(res,1,replace=T)
ystar7=coef(fit20)[1]+coef(fit20)[2]*ystar6+coef(fit20)[3]*ystar5+
coef(fit20)[4]*ystar4+coef(fit20)[5]*ystar3+coef(fit20)[6]*ystar2+
coef(fit20)[7]*ystar1+coef(fit20)[8]*ystar0+
  sample(res,1,replace=T)
ystar8=coef(fit20)[1]+coef(fit20)[2]*ystar7+coef(fit20)[3]*ystar6+
coef(fit20)[4]*ystar5+coef(fit20)[5]*ystar4+
  coef(fit20)[6]*ystar3+coef(fit20)[7]*ystar2+coef(fit20)[8]*ystar1+
  coef(fit20)[9]*ystar0+sample(res,1,replace=T)
ytstar=rep(0,t)
ytstar[1]=ystar0
ytstar[2]=ystar1
ytstar[3]=ystar2
ytstar[4]=ystar3
ytstar[5]=ystar4
ytstar[6]=ystar5
ytstar[7]=ystar6
ytstar[8]=ystar7
ytstar[9]=ystar8
for (i in 10:(t)) {
```

```
    ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
    coef(fit20)[3]*ytstar[i-2]+coef(fit20)[4]*ytstar[i-3]+
    coef(fit20)[5]*ytstar[i-4]+coef(fit20)[6]*ytstar[i-5]+
    coef(fit20)[7]*ytstar[i-6]+coef(fit20)[8]*ytstar[i-7]+
    coef(fit20)[9]*ytstar[i-8]+coef(fit20)[10]*ytstar[i-9]+
    sample(res,1,replace=T)
  }
  emb=embed(ytstar,10)
  y=emb[,1]
  y1=emb[,2]
  y2=emb[,3]
  y3=emb[,4]
  y4=emb[,5]
  y5=emb[,6]
  y6=emb[,7]
  y7=emb[,8]
  y8=emb[,9]
  y9=emb[,10]
  tempo=seq(1,t-9)
  mod4=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+y9+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo+y9:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=11:20)
  Ma$result$chi2[1]
}
Mwbcen=sapply(rep(t,1000),bpar3)
Ma=wald.test(Sigma=vcov(fit9),b=coef(fit9),Terms=11:20)
Ma=Ma$result$chi2[1]
I=length(which(Mwbcen>Ma))
pval=I/1000
pval
#Mwb1
bpar2=function(t){
```

```
ystar0=coef(fit20)[1]+rnorm(1)
ystar1=coef(fit20)[1]+coef(fit20)[2]*ystar0+
as.numeric(residuals(fit20)[1])*rnorm(1)
ystar2=coef(fit20)[1]+coef(fit20)[2]*ystar1+coef(fit20)[3]*ystar0+
as.numeric(residuals(fit20)[2])*rnorm(1)
ystar3=coef(fit20)[1]+coef(fit20)[2]*ystar2+coef(fit20)[3]*ystar1+
  coef(fit20)[4]*ystar0+as.numeric(residuals(fit20)[3])*rnorm(1)
ystar4=coef(fit20)[1]+coef(fit20)[2]*ystar3+coef(fit20)[3]*ystar2+
  coef(fit20)[4]*ystar1+coef(fit20)[5]*ystar0+
  as.numeric(residuals(fit20)[3])*rnorm(1)
ystar5=coef(fit20)[1]+coef(fit20)[2]*ystar4+coef(fit20)[3]*ystar3+
coef(fit20)[4]*ystar2+
  coef(fit20)[5]*ystar1+coef(fit20)[6]*ystar0+
  as.numeric(residuals(fit20)[4])*rnorm(1)
ystar6=coef(fit20)[1]+coef(fit20)[2]*ystar5+coef(fit20)[3]*ystar4+
coef(fit20)[4]*ystar3+coef(fit20)[5]*ystar2+coef(fit20)[6]*ystar1+
coef(fit20)[7]*ystar0+
as.numeric(residuals(fit20)[5])*rnorm(1)
ystar7=coef(fit20)[1]+coef(fit20)[2]*ystar6+coef(fit20)[3]*ystar5+
coef(fit20)[4]*ystar4+
  coef(fit20)[5]*ystar3+coef(fit20)[6]*ystar2+coef(fit20)[7]*ystar1+
  coef(fit20)[8]*ystar0+
  as.numeric(residuals(fit20)[6])*rnorm(1)
ystar8=coef(fit20)[1]+coef(fit20)[2]*ystar7+coef(fit20)[3]*ystar6+
coef(fit20)[4]*ystar5+coef(fit20)[5]*ystar4+
  coef(fit20)[6]*ystar3+coef(fit20)[7]*ystar2+coef(fit20)[8]*ystar1+
  coef(fit20)[9]*ystar0+
  as.numeric(residuals(fit20)[7])*rnorm(1)
ystar9=coef(fit20)[1]+coef(fit20)[2]*ystar7+coef(fit20)[3]*ystar6+
coef(fit20)[4]*ystar5+
  coef(fit20)[5]*ystar4+coef(fit20)[6]*ystar3+coef(fit20)[7]*ystar2+
  coef(fit20)[8]*ystar1+
```

```
  coef(fit20)[9]*ystar0+as.numeric(residuals(fit20)[7])*rnorm(1)
ytstar=rep(0,t)
ytstar[1]=ystar0
ytstar[2]=ystar1
ytstar[3]=ystar2
ytstar[4]=ystar3
ytstar[5]=ystar4
ytstar[6]=ystar5
ytstar[7]=ystar6
ytstar[8]=ystar7
ytstar[9]=ystar8
for (i in 10:(t)) {
  ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
  coef(fit20)[3]*ytstar[i-2]+
    coef(fit20)[4]*ytstar[i-3]+coef(fit20)[5]*ytstar[i-4]+
    coef(fit20)[6]*ytstar[i-5]+
    coef(fit20)[7]*ytstar[i-6]+coef(fit20)[8]*ytstar[i-7]+
    coef(fit20)[9]*ytstar[i-8]+
    coef(fit20)[10]*ytstar[i-9]+as.numeric(residuals(fit20)[i])*rnorm(1)
}
emb=embed(ytstar,10)
y=emb[,1]
y1=emb[,2]
y2=emb[,3]
y3=emb[,4]
y4=emb[,5]
y5=emb[,6]
y6=emb[,7]
y7=emb[,8]
y8=emb[,9]
y9=emb[,10]
tempo=seq(1,t-9)
```

```
  mod4=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+y9+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo+y9:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=11:20)
  Ma$result$chi2[1]
}
Mb=sapply(rep(t,1000),bpar2)
Ma=wald.test(Sigma=vcov(fit9),b=coef(fit9),Terms=11:20)
Ma=Ma$result$chi2[1]
I=length(which(Mb>Ma))
pval=I/1000
pval
#mwb2
bpar=function(t){
  ytstar=coef(fit20)[1]+coef(fit20)[2]*y1+coef(fit20)[3]*y2+
  coef(fit20)[4]*y3+coef(fit20)[5]*y4+
    coef(fit20)[6]*y5+coef(fit20)[7]*y6+coef(fit20)[8]*y7+coef(fit20)[9]*y8+
    coef(fit20)[10]*y9+
    as.numeric(residuals(fit20))*rnorm(t)
  mod4=lm(ytstar~y1+y2+y3+y4+y5+y6+y7+y8+y9+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo+y9:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=11:20)
  Ma$result$chi2[1]
}
Mwb=sapply(rep(t,1000),bpar)
Ma=wald.test(Sigma=vcov(fit9),b=coef(fit9),Terms=11:20)
Ma=Ma$result$chi2[1]
I=length(which(Mwb>Ma))
pval=I/1000
pval
#VA
uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,0)),
                 mean.model = list(armaOrder=c(9,0)))
```

```
fitsim=ugarchfit(spec=uspec,data=serie, solver = 'hybrid')
ht2<- sigma(fitsim)^2
ut <- fit20$residuals
ut=ut^2
VARselect(ut)
uu=embed(ut,10)
u=uu[,1]
u1=uu[,2]
u2=uu[,3]
u3=uu[,4]
u4=uu[,5]
u5=uu[,6]
u6=uu[,7]
u7=uu[,8]
u8=uu[,9]
u9=uu[,10]
u10=uu[,11]
tempo=seq(1,length(u))
t=length(u)
ht2=ht2[1:t]
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+tempo:u3+
tempo:u4+tempo:u5+tempo:u6+tempo:u7+u8:tempo+u9:tempo+u10:tempo)
#statistica Va (16)
t=length(u1)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*10+1))/(SSRh1/(t-(2*10+2)))
Va
anova(modh0,modh1)
#Vb
t=length(u)
```

```
#bootstrap var
bpar=function(t){
  htstar=coef(modh0)+sample(as.numeric(modh0$residuals),t,replace=T)
  modh0=lm(htstar~1)
  modh1=lm(htstar~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+
  tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+u8:tempo+u9:tempo+u10:tempo)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2*10+1))/(SSRh1/(t-(2*10+2)))
  Va
}
Vb=sapply(rep(t,1000),bpar)
#statistica Va (16)
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+
tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+u8:tempo+u9:tempo+u10:tempo)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*10+1))/(SSRh1/(t-(2*10+2)))
I=length(which(Vb>Va))
pval=I/1000
pval

#Vwb
#bootstrap var
bparr=function(t){
  htstar=rep(0,t)
  for (i in 1:t) {
    htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
  }
  modhh0=lm(htstar~1)
```

```
  modhh1=lm(htstar~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+
  tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+u8:tempo+u9:tempo+u10:tempo)
  #statistica Va (16)
  SSRhh0=sum(modhh0$residuals^2)
  SSRhh1=sum(modhh1$residuals^2)
  Va=((SSRhh0-SSRhh1)/(2*10+1))/(SSRhh1/(t-(2*10+2)))
  Va
}
Vwb=sapply(rep(t,1000),bparr)
#statistica Va (16)
ht2=ht2[1:t]
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+tempo:u3+
tempo:u4+tempo:u5+tempo:u6+tempo:u7+u8:tempo+u9:tempo+u10:tempo)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*10+1))/(SSRh1/(t-(2*10+2)))
I=length(which(Vwb>Va))
pval=I/1000
pval
PIL:
dati.pil=read.csv("CLVMNACSCAB1GQIT.csv",header=T)
s=as.numeric(dati.pil[1:109,2])
pil=ts(diff(log(na.omit(s))))
library(vars)
VARselect(pil)
emb=embed(pil,2)
y=emb[,1]
y1=emb[,2]
tempo=seq(1,length(pil)-1)
fit1=lm(y~y1+tempo+tempo:y1)
#ma
```

```
Ma=wald.test(Sigma=vcov(fit1),b=coef(fit1),Terms=3:4)
fit20=lm(y~y1)
t=length(y)
#Mb
res=as.numeric(residuals(fit20)-mean(residuals(fit20)))
bpar3=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
  ytstar=rep(0,t)
  ytstar[1]=ystar0
  for (i in 2:t) {
    ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+sample(res,1,replace=T)
  }
  emb=embed(ytstar,2)
  y=emb[,1]
  y1=emb[,2]
  tempo=seq(1,t-1)
  mod4=lm(y~y1+tempo+tempo:y1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  Ma$result$chi2[1]
}
Mwbcen=sapply(rep(t,1000),bpar3)
Ma=wald.test(Sigma=vcov(fit1),b=coef(fit1),Terms=3:4)
Ma=Ma$result$chi2[1]
I=length(which(Mwbcen>Ma))
pval=I/1000
pval
#Mwb1
bpar2=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
  ytstar=rep(0,t)
  for (i in 2:(t)) {
    ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
```

```
    as.numeric(residuals(fit20)[i])*rnorm(1)
  }
  emb=embed(ytstar,2)
  y=emb[,1]
  y1=emb[,2]
  tempo=seq(1,t-1)
  mod4=lm(y~y1+tempo+tempo:y1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  Ma$result$chi2[1]
}
Mb=sapply(rep(t,200),bpar2)
Ma=wald.test(Sigma=vcov(fit1),b=coef(fit1),Terms=3:4)
Ma=Ma$result$chi2[1]
I=length(which(Mb>Ma))
pval=I/200
pval
#Mwb2
bpar=function(t){
  ytstar=coef(fit20)[1]+coef(fit20)[2]*y1+
    as.numeric(residuals(fit20))*rnorm(t)
  mod4=lm(ytstar~y1+tempo+tempo:y1)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=3:4)
  Ma$result$chi2[1]
}
Mwb=sapply(rep(t,1000),bpar)
Ma=wald.test(Sigma=vcov(fit1),b=coef(fit1),Terms=3:4)
Ma=Ma$result$chi2[1]
I=length(which(Mwb>Ma))
pval=I/1000
pval
#VA
uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(4,0)),
```

```
                mean.model = list(armaOrder=c(1,0)))
fitsim=ugarchfit(spec=uspec,data=pil, solver = 'hybrid')
ht2<- sigma(fitsim)^2
ut <- fit20$residuals
ut=ut^2
VARselect(ut)
uu=embed(ut,5)
u=uu[,1]
u1=uu[,2]
u2=uu[,3]
u3=uu[,4]
u4=uu[,5]
tempo=seq(1,length(u))
ht2=ht2[1:(t-4)]
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+tempo+tempo:u1+tempo:u2+tempo:u3+tempo:u4)
#statistica Va (16)
t=length(u1)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*4+1))/(SSRh1/(t-(2*4+2)))
anova(modh0,modh1)

#Vb
t=length(u)
#bootstrap var
bpar=function(t){
  htstar=coef(modh0)+sample(as.numeric(modh0$residuals),t,replace=T)
  modh0=lm(htstar~1)
  modh1=lm(htstar~u1+u2+u3+u4+tempo+tempo:u1+tempo:u2+tempo:u3+tempo:u4)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
```

```
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2*4+1))/(SSRh1/(t-(2*4+2)))
  Va
}
Vb=sapply(rep(t,1000),bpar)
#statistica Va (16)
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+tempo+tempo:u1+tempo:u2+tempo:u3+tempo:u4)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*4+1))/(SSRh1/(t-(2*4+2)))
I=length(which(Vb>Va))
pval=I/1000
pval
#Vwb
#bootstrap var
bparr=function(t){
  htstar=rep(0,t)
  for (i in 1:t) {
    htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
  }
  modhh0=lm(htstar~1)
  modhh1=lm(htstar~u1+u2+u3+u4+tempo+tempo:u1+tempo:u2+tempo:u3+tempo:u4)
  #statistica Va (16)
  SSRhh0=sum(modhh0$residuals^2)
  SSRhh1=sum(modhh1$residuals^2)
  Va=((SSRhh0-SSRhh1)/(2*4+1))/(SSRhh1/(t-(2*4+2)))
  Va
}
Vwb=sapply(rep(t,1000),bparr)
#statistica Va (16)
modh0=lm(ht2~1)
```

```
modh1=lm(ht2~u1+u2+u3+u4+tempo+tempo:u1+tempo:u2+tempo:u3+tempo:u4)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*4+1))/(SSRh1/(t-(2*4+2)))
I=length(which(Vwb>Va))
pval=I/1000
pval

IPI:
dati.ipi=read.csv("ITAPROINDMISMEI.csv",header=T)
s=as.numeric(dati.ipi[1:748,2])
ipi=ts(diff(log(na.omit(s))))
library(vars)
VARselect(ipi)
emb=embed(ipi,9)
y=emb[,1]
y1=emb[,2]
y2=emb[,3]
y3=emb[,4]
y4=emb[,5]
y5=emb[,6]
y6=emb[,7]
y7=emb[,8]
y8=emb[,9]
tempo=seq(1,length(ipi)-8)
fit8=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+tempo+tempo:y1+tempo:y2+tempo:y3+tempo:y4+
tempo:y5+tempo:y6+tempo:y7+y8:tempo)
#Ma
Ma=wald.test(Sigma=vcov(fit8),b=coef(fit8),Terms=10:18)
fit20=lm(y~y1+y2+y3+y4+y5+y6+y7+y8)
t=length(y)
#Mb
```

```
res=as.numeric(residuals(fit20)-mean(residuals(fit20)))
t=length(y)
bpar3=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
  ystar1=coef(fit20)[1]+coef(fit20)[2]*ystar0+sample(res,1,replace=T)
  ystar2=coef(fit20)[1]+coef(fit20)[2]*ystar1+coef(fit20)[3]*ystar0+
  sample(res,1,replace=T)
  ystar3=coef(fit20)[1]+coef(fit20)[2]*ystar2+coef(fit20)[3]*ystar1+
    coef(fit20)[4]*ystar0+sample(res,1,replace=T)
  ystar4=coef(fit20)[1]+coef(fit20)[2]*ystar3+coef(fit20)[3]*ystar2+
    coef(fit20)[4]*ystar1+coef(fit20)[5]*ystar0+sample(res,1,replace=T)
  ystar5=coef(fit20)[1]+coef(fit20)[2]*ystar4+coef(fit20)[3]*ystar3+
  coef(fit20)[4]*ystar2+
    coef(fit20)[5]*ystar1+coef(fit20)[6]*ystar0+sample(res,1,replace=T)
  ystar6=coef(fit20)[1]+coef(fit20)[2]*ystar5+coef(fit20)[3]*ystar4+
  coef(fit20)[4]*ystar3+coef(fit20)[5]*ystar2+
  coef(fit20)[6]*ystar1+coef(fit20)[7]*ystar0+sample(res,1,replace=T)
  ystar7=coef(fit20)[1]+coef(fit20)[2]*ystar6+coef(fit20)[3]*ystar5+
  coef(fit20)[4]*ystar4+coef(fit20)[5]*ystar3+
  coef(fit20)[6]*ystar2+coef(fit20)[7]*ystar1+
    coef(fit20)[8]*ystar0+
    sample(res,1,replace=T)
  ytstar=rep(0,t)
  ytstar[1]=ystar0
  ytstar[2]=ystar1
  ytstar[3]=ystar2
  ytstar[4]=ystar3
  ytstar[5]=ystar4
  ytstar[6]=ystar5
  ytstar[7]=ystar6
  ytstar[8]=ystar7
  for (i in 9:(t)) {
```

```
    ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
    coef(fit20)[3]*ytstar[i-2]+coef(fit20)[4]*ytstar[i-3]+
    coef(fit20)[5]*ytstar[i-4]+coef(fit20)[6]*ytstar[i-5]+
    coef(fit20)[7]*ytstar[i-6]+coef(fit20)[8]*ytstar[i-7]+
    coef(fit20)[9]*ytstar[i-8]+sample(res,1,replace=T)
  }
  emb=embed(ytstar,9)
  y=emb[,1]
  y1=emb[,2]
  y2=emb[,3]
  y3=emb[,4]
  y4=emb[,5]
  y5=emb[,6]
  y6=emb[,7]
  y7=emb[,8]
  y8=emb[,9]
  tempo=seq(1,t-8)
  mod4=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=10:18)
  Ma$result$chi2[1]
}
Mwbcen=sapply(rep(t,1000),bpar3)
Ma=wald.test(Sigma=vcov(fit8),b=coef(fit8),Terms=10:18)
Ma=Ma$result$chi2[1]
I=length(which(Mwbcen>Ma))
pval=I/1000
pval
#Mwb1
bpar2=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
  ystar1=coef(fit20)[1]+coef(fit20)[2]*ystar0+
```

```
as.numeric(residuals(fit20)[1])*rnorm(1)
ystar2=coef(fit20)[1]+coef(fit20)[2]*ystar1+coef(fit20)[3]*ystar0+
as.numeric(residuals(fit20)[2])*rnorm(1)
ystar3=coef(fit20)[1]+coef(fit20)[2]*ystar2+coef(fit20)[3]*ystar1+
  coef(fit20)[4]*ystar0+as.numeric(residuals(fit20)[3])*rnorm(1)
ystar4=coef(fit20)[1]+coef(fit20)[2]*ystar3+coef(fit20)[3]*ystar2+
  coef(fit20)[4]*ystar1+coef(fit20)[5]*ystar0+
  as.numeric(residuals(fit20)[3])*rnorm(1)
ystar5=coef(fit20)[1]+coef(fit20)[2]*ystar4+coef(fit20)[3]*ystar3+
coef(fit20)[4]*ystar2+coef(fit20)[5]*ystar1+
coef(fit20)[6]*ystar0+as.numeric(residuals(fit20)[4])*rnorm(1)
ystar6=coef(fit20)[1]+coef(fit20)[2]*ystar5+coef(fit20)[3]*ystar4+
coef(fit20)[4]*ystar3+coef(fit20)[5]*ystar2+
  coef(fit20)[6]*ystar1+coef(fit20)[7]*ystar0+
  as.numeric(residuals(fit20)[5])*rnorm(1)
ystar7=coef(fit20)[1]+coef(fit20)[2]*ystar6+coef(fit20)[3]*ystar5+
coef(fit20)[4]*ystar4+coef(fit20)[5]*ystar3+
coef(fit20)[6]*ystar2+coef(fit20)[7]*ystar1+coef(fit20)[8]*ystar0+
  as.numeric(residuals(fit20)[6])*rnorm(1)
ytstar=rep(0,t)
ytstar[1]=ystar0
ytstar[2]=ystar1
ytstar[3]=ystar2
ytstar[4]=ystar3
ytstar[5]=ystar4
ytstar[6]=ystar5
ytstar[7]=ystar6
ytstar[8]=ystar7
for (i in 9:(t)) {
  ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
  coef(fit20)[3]*ytstar[i-2]+coef(fit20)[4]*ytstar[i-3]+
  coef(fit20)[5]*ytstar[i-4]+coef(fit20)[6]*ytstar[i-5]+
```

```
      coef(fit20)[7]*ytstar[i-6]+coef(fit20)[8]*ytstar[i-7]+
      coef(fit20)[9]*ytstar[i-8]+
      as.numeric(residuals(fit20)[i])*rnorm(1)
  }
  emb=embed(ytstar,9)
  y=emb[,1]
  y1=emb[,2]
  y2=emb[,3]
  y3=emb[,4]
  y4=emb[,5]
  y5=emb[,6]
  y6=emb[,7]
  y7=emb[,8]
  y8=emb[,9]
  tempo=seq(1,t-8)
  mod4=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=10:18)
  Ma$result$chi2[1]
}
Mb=sapply(rep(t,1000),bpar2)
Ma=wald.test(Sigma=vcov(fit8),b=coef(fit8),Terms=10:18)
Ma=Ma$result$chi2[1]
I=length(which(Mb>Ma))
pval=I/1000
pval
#Mwb2
bpar=function(t){
  ytstar=coef(fit20)[1]+coef(fit20)[2]*y1+coef(fit20)[3]*y2+
  coef(fit20)[4]*y3+coef(fit20)[5]*y4+
    coef(fit20)[6]*y5+coef(fit20)[7]*y6+coef(fit20)[8]*y7+coef(fit20)[9]*y8+
    as.numeric(residuals(fit20))*rnorm(t)
```

```
  mod4=lm(ytstar~y1+y2+y3+y4+y5+y6+y7+y8+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=10:18)
  Ma$result$chi2[1]
}
Mwb=sapply(rep(t,1000),bpar)
Ma=wald.test(Sigma=vcov(fit8),b=coef(fit8),Terms=10:18)
Ma=Ma$result$chi2[1]
I=length(which(Mwb>Ma))
pval=I/1000
pval
#VA
uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(7,0)),
                 mean.model = list(armaOrder=c(8,0)))
fitsim=ugarchfit(spec=uspec,data=ipi, solver = 'hybrid')
ht2<- sigma(fitsim)^2
ut <- fit20$residuals
ut=ut^2
VARselect(ut)
uu=embed(ut,8)
u=uu[,1]
u1=uu[,2]
u2=uu[,3]
u3=uu[,4]
u4=uu[,5]
u5=uu[,6]
u6=uu[,7]
u7=uu[,8]
tempo=seq(1,length(u))
ht2=ht2[1:(t-7)]
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+tempo+tempo:u1+tempo:u2+tempo:u3+
```

```
tempo:u4+tempo:u5+tempo:u6+tempo:u7)
#statistica Va (16)
t=length(u1)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*7+1))/(SSRh1/(t-(2*7+2)))
anova(modh0,modh1)
#Vb
t=length(u)
#bootstrap var
bpar=function(t){
  htstar=coef(modh0)+sample(as.numeric(modh0$residuals),t,replace=T)
  modh0=lm(htstar~1)
  modh1=lm(htstar~u1+u2+u3+u4+u5+u6+u7+tempo+tempo:u1+tempo:u2+tempo:u3+
  tempo:u4+tempo:u5+tempo:u6+tempo:u7)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2*7+1))/(SSRh1/(t-(2*7+2)))
  Va
}
Vb=sapply(rep(t,1000),bpar)
#statistica Va (16)
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+tempo+tempo:u1+tempo:u2+tempo:u3+
tempo:u4+tempo:u5+tempo:u6+tempo:u7)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*7+1))/(SSRh1/(t-(2*7+2)))
I=length(which(Vb>Va))
pval=I/1000
pval
```

```
#Vwb
#bootstrap var
bparr=function(t){
  htstar=rep(0,t)
  for (i in 1:t) {
    htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
  }
  modhh0=lm(htstar~1)
  modhh1=lm(htstar~u1+u2+u3+u4+u5+u6+u7+tempo+tempo:u1+tempo:u2+tempo:u3+
  tempo:u4+tempo:u5+tempo:u6+tempo:u7)
  #statistica Va (16)
  SSRhh0=sum(modhh0$residuals^2)
  SSRhh1=sum(modhh1$residuals^2)
  Va=((SSRhh0-SSRhh1)/(2*7+1))/(SSRhh1/(t-(2*7+2)))
  Va
}
Vwb=sapply(rep(t,1000),bparr)
#statistica Va (16)
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+tempo+tempo:u1+tempo:u2+tempo:u3+
tempo:u4+tempo:u5+tempo:u6+tempo:u7)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*7+1))/(SSRh1/(t-(2*7+2)))
I=length(which(Vwb>Va))
pval=I/1000
pval

CPI:
dati.cpi=read.csv("ITACPIALLMINMEI.csv",header=T)
s=as.numeric(dati.cpi[1:749,2])
```

```
cpi=ts(diff(log(na.omit(s))))
library(vars)
VARselect(cpi)
emb=embed(cpi,9)
y=emb[,1]
y1=emb[,2]
y2=emb[,3]
y3=emb[,4]
y4=emb[,5]
y5=emb[,6]
y6=emb[,7]
y7=emb[,8]
y8=emb[,9]
tempo=seq(1,length(cpi)-8)
fit8=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+tempo+tempo:y1+tempo:y2+tempo:y3+
tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo)
#Ma
Ma=wald.test(Sigma=vcov(fit8),b=coef(fit8),Terms=10:18)
Ma
fit20=lm(y~y1+y2+y3+y4+y5+y6+y7+y8)
t=length(y)
#Mb
#mwb prof
res=as.numeric(residuals(fit20)-mean(residuals(fit20)))
t=length(y)
bpar3=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
  ystar1=coef(fit20)[1]+coef(fit20)[2]*ystar0+sample(res,1,replace=T)
  ystar2=coef(fit20)[1]+coef(fit20)[2]*ystar1+coef(fit20)[3]*ystar0+
  sample(res,1,replace=T)
  ystar3=coef(fit20)[1]+coef(fit20)[2]*ystar2+coef(fit20)[3]*ystar1+
    coef(fit20)[4]*ystar0+sample(res,1,replace=T)
```

```
ystar4=coef(fit20)[1]+coef(fit20)[2]*ystar3+coef(fit20)[3]*ystar2+
  coef(fit20)[4]*ystar1+coef(fit20)[5]*ystar0+sample(res,1,replace=T)
ystar5=coef(fit20)[1]+coef(fit20)[2]*ystar4+coef(fit20)[3]*ystar3+
coef(fit20)[4]*ystar2+coef(fit20)[5]*ystar1+
coef(fit20)[6]*ystar0+sample(res,1,replace=T)
ystar6=coef(fit20)[1]+coef(fit20)[2]*ystar5+coef(fit20)[3]*ystar4+
coef(fit20)[4]*ystar3+coef(fit20)[5]*ystar2+
  coef(fit20)[6]*ystar1+coef(fit20)[7]*ystar0+sample(res,1,replace=T)
ystar7=coef(fit20)[1]+coef(fit20)[2]*ystar6+coef(fit20)[3]*ystar5+
coef(fit20)[4]*ystar4+coef(fit20)[5]*ystar3+
coef(fit20)[6]*ystar2+coef(fit20)[7]*ystar1+coef(fit20)[8]*ystar0+
  sample(res,1,replace=T)
ytstar=rep(0,t)
ytstar[1]=ystar0
ytstar[2]=ystar1
ytstar[3]=ystar2
ytstar[4]=ystar3
ytstar[5]=ystar4
ytstar[6]=ystar5
ytstar[7]=ystar6
ytstar[8]=ystar7
for (i in 9:(t)) {
  ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
  coef(fit20)[3]*ytstar[i-2]+coef(fit20)[4]*ytstar[i-3]+
  coef(fit20)[5]*ytstar[i-4]+coef(fit20)[6]*ytstar[i-5]+
    coef(fit20)[7]*ytstar[i-6]+coef(fit20)[8]*ytstar[i-7]+
    coef(fit20)[9]*ytstar[i-8]+sample(res,1,replace=T)
}
emb=embed(ytstar,9)
y=emb[,1]
y1=emb[,2]
y2=emb[,3]
```

```
  y3=emb[,4]
  y4=emb[,5]
  y5=emb[,6]
  y6=emb[,7]
  y7=emb[,8]
  y8=emb[,9]
  tempo=seq(1,t-8)
  mod4=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=10:18)
  Ma$result$chi2[1]
}
Mwbcen=sapply(rep(t,1000),bpar3)
Ma=wald.test(Sigma=vcov(fit8),b=coef(fit8),Terms=10:18)
Ma=Ma$result$chi2[1]
I=length(which(Mwbcen>Ma))
pval=I/1000
pval
#Mwb1
bpar2=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
  ystar1=coef(fit20)[1]+coef(fit20)[2]*ystar0+
  as.numeric(residuals(fit20)[1])*rnorm(1)
  ystar2=coef(fit20)[1]+coef(fit20)[2]*ystar1+coef(fit20)[3]*ystar0+
  as.numeric(residuals(fit20)[2])*rnorm(1)
  ystar3=coef(fit20)[1]+coef(fit20)[2]*ystar2+coef(fit20)[3]*ystar1+
    coef(fit20)[4]*ystar0+as.numeric(residuals(fit20)[3])*rnorm(1)
  ystar4=coef(fit20)[1]+coef(fit20)[2]*ystar3+coef(fit20)[3]*ystar2+
    coef(fit20)[4]*ystar1+coef(fit20)[5]*ystar0+
    as.numeric(residuals(fit20)[3])*rnorm(1)
  ystar5=coef(fit20)[1]+coef(fit20)[2]*ystar4+coef(fit20)[3]*ystar3+
  coef(fit20)[4]*ystar2+coef(fit20)[5]*ystar1+
```

```
coef(fit20)[6]*ystar0+as.numeric(residuals(fit20)[4])*rnorm(1)
ystar6=coef(fit20)[1]+coef(fit20)[2]*ystar5+coef(fit20)[3]*ystar4+
coef(fit20)[4]*ystar3+coef(fit20)[5]*ystar2+
  coef(fit20)[6]*ystar1+coef(fit20)[7]*ystar0+
  as.numeric(residuals(fit20)[5])*rnorm(1)
ystar7=coef(fit20)[1]+coef(fit20)[2]*ystar6+coef(fit20)[3]*ystar5+
coef(fit20)[4]*ystar4+coef(fit20)[5]*ystar3+
coef(fit20)[6]*ystar2+coef(fit20)[7]*ystar1+coef(fit20)[8]*ystar0+
  as.numeric(residuals(fit20)[6])*rnorm(1)
ytstar=rep(0,t)
ytstar[1]=ystar0
ytstar[2]=ystar1
ytstar[3]=ystar2
ytstar[4]=ystar3
ytstar[5]=ystar4
ytstar[6]=ystar5
ytstar[7]=ystar6
ytstar[8]=ystar7
for (i in 9:(t)) {
  ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
  coef(fit20)[3]*ytstar[i-2]+coef(fit20)[4]*ytstar[i-3]+
  coef(fit20)[5]*ytstar[i-4]+coef(fit20)[6]*ytstar[i-5]+
    coef(fit20)[7]*ytstar[i-6]+coef(fit20)[8]*ytstar[i-7]+
    coef(fit20)[9]*ytstar[i-8]+
    as.numeric(residuals(fit20)[i])*rnorm(1)
}
emb=embed(ytstar,9)
y=emb[,1]
y1=emb[,2]
y2=emb[,3]
y3=emb[,4]
y4=emb[,5]
```

```
  y5=emb[,6]
  y6=emb[,7]
  y7=emb[,8]
  y8=emb[,9]
  tempo=seq(1,t-8)
  mod4=lm(y~y1+y2+y3+y4+y5+y6+y7+y8+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=10:18)
  Ma$result$chi2[1]
}
Mb=sapply(rep(t,1000),bpar2)
Ma=wald.test(Sigma=vcov(fit8),b=coef(fit8),Terms=10:18)
Ma=Ma$result$chi2[1]
I=length(which(Mb>Ma))
pval=I/1000
pval
#Mwb2
bpar=function(t){
  ytstar=coef(fit20)[1]+coef(fit20)[2]*y1+coef(fit20)[3]*y2+
  coef(fit20)[4]*y3+coef(fit20)[5]*y4+
    coef(fit20)[6]*y5+coef(fit20)[7]*y6+coef(fit20)[8]*y7+coef(fit20)[9]*y8+
    as.numeric(residuals(fit20))*rnorm(t)
  mod4=lm(ytstar~y1+y2+y3+y4+y5+y6+y7+y8+tempo+tempo:y1+tempo:y2+tempo:y3+
  tempo:y4+tempo:y5+tempo:y6+tempo:y7+y8:tempo)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=10:18)
  Ma$result$chi2[1]
}
Mwb=sapply(rep(t,1000),bpar)
Ma=wald.test(Sigma=vcov(fit8),b=coef(fit8),Terms=10:18)
Ma=Ma$result$chi2[1]
I=length(which(Mwb>Ma))
pval=I/1000
```

```
pval
#VA
uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(8,0)),
                 mean.model = list(armaOrder=c(8,0)))
fitsim=ugarchfit(spec=uspec,data=cpi, solver = 'hybrid')
ht2<- sigma(fitsim)^2
ut <- fit20$residuals
ut=ut^2
VARselect(ut) #8 lag
uu=embed(ut,9)
u=uu[,1]
u1=uu[,2]
u2=uu[,3]
u3=uu[,4]
u4=uu[,5]
u5=uu[,6]
u6=uu[,7]
u7=uu[,8]
u8=uu[,9]
tempo=seq(1,length(u))
ht2=ht2[1:(t-8)]
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+tempo+tempo:u1+tempo:u2+tempo:u3+
tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8)
#statistica Va (16)
t=length(u1)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*8+1))/(SSRh1/(t-(2*8+2)))
anova(modh0,modh1)

#Vb
```

```
t=length(u)
#bootstrap var
bpar=function(t){
  htstar=coef(modh0)+sample(as.numeric(modh0$residuals),t,replace=T)
  modh0=lm(htstar~1)
  modh1=lm(htstar~u1+u2+u3+u4+u5+u6+u7+u8+tempo+tempo:u1+tempo:u2+tempo:u3+
  tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8)
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2*8+1))/(SSRh1/(t-(2*8+2)))
  Va
}
Vb=sapply(rep(t,1000),bpar)
#statistica Va (16)
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+tempo+tempo:u1+tempo:u2+tempo:u3+
tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*8+1))/(SSRh1/(t-(2*8+2)))
I=length(which(Vb>Va))
pval=I/1000
pval

#Vwb
#bootstrap var
bparr=function(t){
  htstar=rep(0,t)
  for (i in 1:t) {
    htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
  }
```

```
  modhh0=lm(htstar~1)
  modhh1=lm(htstar~u1+u2+u3+u4+u5+u6+u7+u8+tempo+tempo:u1+tempo:u2+
  tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8)
  #statistica Va (16)
  SSRhh0=sum(modhh0$residuals^2)
  SSRhh1=sum(modhh1$residuals^2)
  Va=((SSRhh0-SSRhh1)/(2*8+1))/(SSRhh1/(t-(2*8+2)))
  Va
}
Vwb=sapply(rep(t,1000),bparr)
#statistica Va (16)
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+tempo+tempo:u1+tempo:u2+tempo:u3+
tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*8+1))/(SSRh1/(t-(2*8+2)))
I=length(which(Vwb>Va))
pval=I/1000
pval

FTSE MIB:

dati.mib=read.table("ftsemib.txt",header=T)
set.seed(1324)
s=as.numeric(dati.mib[,2])
mib=ts(diff(log(na.omit(s))))
library(vars)
VARselect(mib)
emb=embed(mib,4)
y=emb[,1]
y1=emb[,2]
```

```
y2=emb[,3]
y3=emb[,4]
tempo=seq(1,length(mib)-3)
fit3=lm(y~y1+y2+y3+tempo+tempo:y1+tempo:y2+tempo:y3)
#Ma
Ma=wald.test(Sigma=vcov(fit3),b=coef(fit3),Terms=5:8)
fit20=lm(y~y1+y2+y3)
t=length(y)
#Mb
res=as.numeric(residuals(fit20)-mean(residuals(fit20)))
t=length(y)
bpar3=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
  ystar1=coef(fit20)[1]+coef(fit20)[2]*ystar0+sample(res,1,replace=T)
  ystar2=coef(fit20)[1]+coef(fit20)[2]*ystar1+coef(fit20)[3]*ystar0+
  sample(res,1,replace=T)
  ytstar=rep(0,t)
  ytstar[1]=ystar0
  ytstar[2]=ystar1
  ytstar[3]=ystar2
  for (i in 4:(t)) {
    ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
    coef(fit20)[3]*ytstar[i-2]+
      coef(fit20)[4]*ytstar[i-3]+sample(res,1,replace=T)
  }
  emb=embed(ytstar,4)
  y=emb[,1]
  y1=emb[,2]
  y2=emb[,3]
  y3=emb[,4]
  tempo=seq(1,t-3)
  mod4=lm(y~y1+y2+y3+tempo+tempo:y1+tempo:y2+tempo:y3)
```

```
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=5:8)
  Ma$result$chi2[1]
}
Mwbcen=sapply(rep(t,1000),bpar3)
Ma=wald.test(Sigma=vcov(fit3),b=coef(fit3),Terms=5:8)
Ma=Ma$result$chi2[1]
I=length(which(Mwbcen>Ma))
pval=I/1000
pval
#Mwb1
bpar2=function(t){
  ystar0=coef(fit20)[1]+rnorm(1)
  ystar1=coef(fit20)[1]+coef(fit20)[2]*ystar0+
  as.numeric(residuals(fit20)[1])*rnorm(1)
  ystar2=coef(fit20)[1]+coef(fit20)[2]*ystar1+coef(fit20)[3]*ystar0+
  as.numeric(residuals(fit20)[2])*rnorm(1)
  ytstar=rep(0,t)
  ytstar[1]=ystar0
  ytstar[2]=ystar1
  ytstar[3]=ystar2
  for (i in 4:(t)) {
    ytstar[i]=coef(fit20)[1]+coef(fit20)[2]*ytstar[i-1]+
    coef(fit20)[3]*ytstar[i-2]+
      coef(fit20)[4]*ytstar[i-3]+as.numeric(residuals(fit20)[i])*rnorm(1)
  }
  emb=embed(ytstar,4)
  y=emb[,1]
  y1=emb[,2]
  y2=emb[,3]
  y3=emb[,4]
  tempo=seq(1,t-3)
  mod4=lm(y~y1+y2+y3+tempo+tempo:y1+tempo:y2+tempo:y3)
```

```
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=5:8)
  Ma$result$chi2[1]
}
Mb=sapply(rep(t,1000),bpar2)
Ma=wald.test(Sigma=vcov(fit3),b=coef(fit3),Terms=5:8)
Ma=Ma$result$chi2[1]
I=length(which(Mb>Ma))
pval=I/1000
pval
#Mwb2
bpar=function(t){
  ytstar=coef(fit20)[1]+coef(fit20)[2]*y1+coef(fit20)[3]*y2+
  coef(fit20)[4]*y3+as.numeric(residuals(fit20))*rnorm(t)
  mod4=lm(ytstar~y1+y2+y3+tempo+tempo:y1+tempo:y2+tempo:y3)
  Ma=wald.test(Sigma=vcov(mod4),b=coef(mod4),Terms=5:8)
  Ma$result$chi2[1]
}
Mwb=sapply(rep(t,1000),bpar)
Ma=wald.test(Sigma=vcov(fit3),b=coef(fit3),Terms=5:8)
Ma=Ma$result$chi2[1]
I=length(which(Mwb>Ma))
pval=I/1000
pval
#VA
uspec=ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(10,0)),
                 mean.model = list(armaOrder=c(3,0)))
fitsim=ugarchfit(spec=uspec,data=mib, solver = 'hybrid')
ht2<- sigma(fitsim)^2
ut <- fit20$residuals
ut=ut^2
VARselect(ut) #10 lag
uu=embed(ut,11)
```

```
u=uu[,1]
u1=uu[,2]
u2=uu[,3]
u3=uu[,4]
u4=uu[,5]
u5=uu[,6]
u6=uu[,7]
u7=uu[,8]
u8=uu[,9]
u9=uu[,10]
u10=uu[,11]
tempo=seq(1,length(u))
t=length(u)
ht2=ht2[1:t]
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+
tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8+tempo:u9+tempo:u10)
#statistica Va (16)
t=length(u1)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*10+1))/(SSRh1/(t-(2*10+2)))
anova(modh0,modh1)

#Vb
t=length(u)
#bootstrap var
bpar=function(t){
  htstar=coef(modh0)+sample(as.numeric(modh0$residuals),t,replace=T)
  modh0=lm(htstar~1)
  modh1=lm(htstar~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+
  tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8+tempo:u9+tempo:u10)
```

```
  #statistica Va (16)
  SSRh0=sum(modh0$residuals^2)
  SSRh1=sum(modh1$residuals^2)
  Va=((SSRh0-SSRh1)/(2*10+1))/(SSRh1/(t-(2*10+2)))
  Va
}
Vb=sapply(rep(t,1000),bpar)
#statistica Va (16)
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+
tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8+tempo:u9+tempo:u10)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*10+1))/(SSRh1/(t-(2*10+2)))
I=length(which(Vb>Va))
pval=I/1000
pval

#Vwb
#bootstrap var
bparr=function(t){
  htstar=rep(0,t)
  for (i in 1:t) {
    htstar[i]=coef(modh0)+as.numeric(residuals(modh0)[i])*rnorm(1)
  }
  modhh0=lm(htstar~1)
  modhh1=lm(htstar~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+
  tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8+tempo:u9+tempo:u10)
  #statistica Va (16)
  SSRhh0=sum(modhh0$residuals^2)
  SSRhh1=sum(modhh1$residuals^2)
  Va=((SSRhh0-SSRhh1)/(2*10+1))/(SSRhh1/(t-(2*10+2)))
```

```
  Va
}
Vwb=sapply(rep(t,1000),bparr)
#statistica Va (16)
modh0=lm(ht2~1)
modh1=lm(ht2~u1+u2+u3+u4+u5+u6+u7+u8+u9+u10+tempo+tempo:u1+tempo:u2+
tempo:u3+tempo:u4+tempo:u5+tempo:u6+tempo:u7+tempo:u8+tempo:u9+tempo:u10)
SSRh0=sum(modh0$residuals^2)
SSRh1=sum(modh1$residuals^2)
Va=((SSRh0-SSRh1)/(2*10+1))/(SSRh1/(t-(2*10+2)))
I=length(which(Vwb>Va))
pval=I/1000
pval
```