UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI ECONOMIA "GIORGIO FUÀ"

Corso di Laurea Magistrale in Scienze Economiche e Finanziarie,
Analista Finanziario

FILTERED HISTORICAL SIMULATION (FHS): UN'ANALISI EMPIRICA CON PYTHON

Filtered Historical Simulation (FHS): an empirical analysis with Python

Relatore: Chiar.mo
Prof. Valerio Sullo

Tesi di Laurea di:
Nicola Crescimbeni

Anno Accademico 2021 – 2022

SOMMARIO

INTRODUZIONE 5

CAPITOLO 1 7

FILTERED HISTORICAL SIMULATION (FHS): ANALISI DEL METODO SEMIPARAMETRICO 7

1.1 OVERVIEW SULLA TECNICA FHS 7

1.2. CENNI STORICI SUL MODELLO 10

1.3. IMPLEMENTAZIONE 13

1.3.1 Simulazione per un solo asset 14

1.3.2 Simulazione per più assets 18

CAPITOLO 2 21

ANALISI EMPIRICA 21

2.1 OBIETTIVO DELL'ANALISI 21

2.2 RACCOLTA ED ANALISI ESPLORATIVA DEI DATI 22

2.3 ANALISI PARAMETRICA 35

2.3.1 Modello ARMA 35

2.3.2 Modello GARCH 45

2.3.3 Scaling dei residui 47

2.4 SIMULAZIONE 49

2.4.1 Caratteristiche generali 49

2.4.2 Implementazione e risultati ottenuti 50

CAPITOLO 3 56

VaR E BACKTESTING 56

3.1 VALUE AT RISK (VaR) 56

3.2 BACKTESTING 58

3.2.1 Backtesting sul VaR 58
3.2.2 Backtesting: analisi empirica 63

CONCLUSIONE **72**

APPENDICE **75**

BIBLIOGRAFIA **93**

INTRODUZIONE

Il presente elaborato ha l'obiettivo di analizzare la *Filtered Historical Simulation* (FHS), metodologia di simulazione semiparametrica che combina la simulazione storica pura ai metodi di simulazione strettamente parametrici.

Nello specifico, nel Capitolo 1 verrà illustrato il metodo in questione da un punto di vista statistico, evidenziando in modo dettagliato le differenze tra simulazione FHS univariata, resa possibile dalla tecnica del *portfolio freezing*, e multivariata, basata piuttosto sulla matrice varianza-covarianza. Inoltre, verranno spiegate le ragioni e le motivazioni che hanno portato allo sviluppo della metodologia FHS da parte di Barone-Adesi e Giannopoulos. Questa breve digressione fornisce al lettore una visione più ampia dell'argomento, fondamentale affinché possano essere valutati i pregi e i limiti della simulazione FHS.

Dopo aver esposto secondo un approccio teorico il metodo in questione, la trattazione si sposta verso l'analisi empirica effettuata in relazione ad un portafoglio composto da sei ETF europei. In particolare, nel Capitolo 2 sarà condotta i*n primis* l'analisi esplorativa dei dati, focalizzando l'attenzione sulla gestione delle osservazioni mancanti e sulle caratteristiche degli *assets* e del portafoglio. Successivamente, verranno mostrati i risultati relativi ai processi di stima, con l'indagine inferenziale associata, e di simulazione. In relazione a quest'ultima, saranno valutate le differenze tra i valori di simulazione ottenuti mediante il metodo FHS e il modello parametrico ARMA-GARCH.

Infine, nel Capitolo 3 inizialmente saranno illustrati i concetti di *VaR* e di *backtesting*, mentre in seguito verrà valutata in modo retrospettivo la solidità del modello FHS, effettuando anche una comparazione con i risultati ottenuti dal modello parametrico ARMA-GARCH e dal metodo di simulazione storica basato su distribuzione gaussiana dei rendimenti.

In sintesi, l'elaborato mira a fornire al lettore un quadro completo del metodo di simulazione FHS, precisando i punti di forza e di debolezza della tecnica di Barone-Adesi e Giannopoulos. Le valutazioni e le considerazioni effettuate tengono conto delle ragioni alla base dell'ideazione del modello e sono al contempo avvalorate dai risultati ottenuti grazie all'analisi empirica, effettuata mediante l'ausilio del linguaggio di programmazione *Python.*

CAPITOLO 1

FILTERED HISTORICAL SIMULATION (FHS): ANALISI DEL METODO SEMIPARAMETRICO

1.1 OVERVIEW SULLA TECNICA FHS

La *Filtered Historical Simulation* (FHS) è un metodo di simulazione semiparametrico utilizzato per la generazione di scenari relativi al valore di un portafoglio composto da *assets* finanziari, che agiscono come *risk factors* per il rendimento futuro del portafoglio stesso. Sulla base dei vari scenari creati mediante la simulazione, è possibile costruire la distribuzione dei rendimenti attesi del portafoglio in un dato momento futuro. Questa tecnica si avvale dell'utilizzo contestuale dei rendimenti storici dei *risk factors* come input del processo di simulazione e di alcuni modelli econometrici non lineari, ragion per cui può essere considerato un metodo di simulazione semiparametrico. La metodologia FHS viene ampiamente utilizzata nell'ambito del Risk Management poiché, combinando i vantaggi dei metodi di simulazione tradizionale con quelli dei modelli parametrici, risulta agevole calcolare misure di rischio quali il *Value at Risk* (VaR) o l'*Expected Shortfall* (ES).

Il principale vantaggio di questo modello è la presenza di ipotesi minimali sulla distribuzione dei rendimenti, essenziali invece nel caso delle metodologie parametriche. La mancanza di vincoli particolarmente stringenti sulla distribuzione

condizionale dei rendimenti consente di evitare che eventi più rari e dunque meno probabili (ma non per questo impossibili) vengano sottostimati dal modello. Infatti, le distribuzioni teoriche, quali ad esempio la distribuzione Gaussiana, attribuiscono bassa probabilità a rendimenti estremi degli *assets* che dal punto di vista empirico risultano invece più frequenti. Per questo motivo le distribuzioni empiriche dei rendimenti sono caratterizzate tipicamente da code più spesse[1].

Ciò che rende più efficace il presente metodo a paragone dei classici modelli di simulazione storica è l'utilizzo di una tecnica di *scaling* applicata ai rendimenti storici degli *assets*: questi ultimi vengono infatti standardizzati rispetto alla volatilità giornaliera per poi essere moltiplicati per la volatilità futura prevista da uno specifico modello econometrico eteroschedastico (*re-scaling*). Così facendo, tali rendimenti vengono resi serialmente indipendenti[2] e adatti ad essere utilizzati come input nel processo di simulazione. Pur essendo basata su dati rappresentativi del passato come nel caso delle simulazioni storiche tradizionali, la tecnica FHS effettua la simulazione a partire dai valori correnti di rendimento, volatilità e correlazione, utilizzando i modelli ARMA e GARCH. Dunque, la grande differenza rispetto ai classici modelli di simulazione storica è proprio il modo di proiettare nel futuro i dati storici, visto l'utilizzo in parallelo dei modelli parametrici sopracitati.

[1] Da questo fenomeno deriva la tipica denominazione "*fat tails distributions*".

[2] Non è detto che i rendimenti siano sempre caratterizzati da correlazione seriale. Lo scaling consente in ogni caso di ovviare alla problematica in questione.

Gli scenari generati dalla tecnica FHS si basano su una tecnica di ricampionamento con reimmissione chiamata *bootstrap*[3] che garantisce la convergenza della distribuzione simulata a quella del *Data Generating Process (DGP)*[4]. Per metodi di ricampionamento si intendono quelle procedure inferenziali che mirano ad accrescere il potenziale informativo di un campione attraverso l'estrazione di sottocampioni dello stesso. Il *bootstrap* consente inoltre di ottenere vantaggi anche dal punto di vista computazionale: mediante la sua applicazione infatti la crescita del numero dei parametri e del tempo di elaborazione dei dati risulta essere lineare rispetto al numero di *risk factors*, a differenza di quanto avviene nei modelli che si basano sulla matrice varianza-covarianza.

I vantaggi derivanti dall'efficienza computazionale consentono anche di poter implementare piuttosto agevolmente attività di *stress testing* ed analisi di sensitività, essenziali affinché un portafoglio possa essere valutato anche dal punto di vista della sua resilienza a scenari più o meno avversi.

[3] Si veda Efron e Tibshirani (1993), "An Introduction to the Bootstrap", dove il metodo del bootstrapping viene introdotto e descritto accuratamente.

[4] Per *Data Generating Process (DGP)* si intende il meccanismo mediante il quale vengono generati i dati, indipendentemente dal fatto che siano già osservati o ancora non disponibili.

1.2. CENNI STORICI SUL MODELLO

La tecnica FHS nasce dall'esigenza di superare il problema della stima della matrice varianza-covarianza tra i vari *risk factors*, che risulta complicata soprattutto nel caso in cui il numero di questi ultimi sia elevato. Infatti, una stima poco robusta delle correlazioni degli *assets* impatta negativamente sull'affidabilità delle misure di rischio calcolabili su un predeterminato portafoglio. Un grande passo avanti per superare questo scoglio viene compiuto da Giovanni Barone-Adesi e Kostas Giannopoulos nel 1996, quando insieme pubblicano un articolo chiamato "*VaR without using correlations*" nel *Journal of Futures Markets*. L'idea principale di questo contributo risiede nel fatto che la matrice varianza-covarianza sia necessaria solamente in caso di ottimizzazioni di portafoglio, allo scopo dunque di determinare i pesi dei vari *assets* all'interno del portafoglio stesso. Nel mondo del Risk Management, per contro, la composizione di quest'ultimo risulta spesso già predeterminata, poiché le decisioni di investimento vengono prese a monte dagli *asset managers*. Dunque, secondo Barone-Adesi e Giannopoulos, l'uso della matrice di correlazione allo scopo di determinare il VaR di un prestabilito portafoglio risulta essere inutile e al contempo espone la misurazione del rischio ad errori dovuti all'eventuale scarsa precisione della stima.

Le ragioni alla base della precedente affermazione sono strettamente legate all'intrinseca necessità da parte dei metodi parametrici di approssimare la distribuzione reale dei rendimenti con una distribuzione teorica. Come accennato

nel precedente paragrafo, questa ipotesi di base sembra essere piuttosto forte e irrealistica e dal punto di vista empirico. Come analizzato nell'articolo, ciò risulta evidente soprattutto nel caso di approssimazione mediante distribuzione normale. In sintesi, la stima della matrice di correlazione richiede di approssimare la distribuzione empirica con una teorica di fatto ignota che, non garantendo a priori di trattare in modo efficiente il problema delle *fat tails distributions,* rende dunque la stima del VaR inaffidabile. Questa criticità assume un rilievo particolare soprattutto quando il numero degli *assets* presenti all'interno del portafoglio risulta essere elevato, poiché ipotizzare che ognuno di essi sia rappresentato dalla stessa distribuzione teorica è molto spesso irrealistico. Inoltre, partendo anche dall'assunto che la stima della matrice varianza-covarianza sia accurata, non esiste nessuna garanzia che la matrice stessa risulti invertibile, a causa di dati insufficienti, mancanti o di bassa qualità[5].

Dopo aver illustrato la problematica della matrice di correlazione nel paper del 1996, Barone-Adesi e Giannopoulos implementano la tecnica di simulazione FHS in due successivi articoli, pubblicati rispettivamente nel *Risk magazine* nel

[5] È opportuno considerare che nel caso di dataset multivariati è la lunghezza della serie storica più breve a determinare l'ampiezza del dataset stesso: tutte le serie devono dunque essere di lunghezza sufficiente.

1998[6] e nel *Journal of Futures Markets* nel 1999[7]. Il primo contributo verte su un'applicazione di un ipotetico *synthetic portfolio*, costruito in modo tale da essere diversificato tra tredici mercati azionari nazionali. Questi ultimi sono pesati proporzionalmente in base alla loro capitalizzazione di mercato all'interno dell'indice mondiale nel dicembre 1995. Dunque, i pesi vengono mantenuti costanti per dieci anni e moltiplicati per i rispettivi rendimenti degli indici nazionali azionari (cd, *constant-weights portfolio*). È in questo contributo che viene introdotta per la prima volta l'azione di *scaling*, citata nel paragrafo precedente, che caratterizza il metodo FHS. L'articolo del 1999 considera come *risk factors* futures e opzioni su futures negoziati in vari mercati londinesi e *"plain vanilla"* interest rate swaps. Nonostante quest'ultimo articolo venga pubblicato soltanto nel 1999, una prima bozza viene resa disponibile già dal 1997, quando la *London Clearing House (LCH)* inizia uno studio di fattibilità sulle operazioni di *clearing* tra prodotti OTC. La tecnica FHS viene ritenuta assai promettente dalla Cassa di Compensazione inglese e utile soprattutto a modellare la curva dei rendimenti giornaliera delle principali valute, necessaria nel *clearing* tra contratti SWAPS. In aggiunta, in questo periodo la LCH, con l'ausilio di Barone-Adesi e Giannopoulos, implementa un estensivo

[6] Giovanni Barone-Adesi, Frederick Bourgoin and Kostas Giannopoulos (1998), "Don't look back" , Risk magazine, **11**, August, 100-104

[7] Giovanni Barone-Adesi, Kostas Giannopoulos and Les Vosper (1999), "VaR without correlations for non-linear portfolios", *Journal of Futures Markets*, **19**, August, 583-602.

backtesting dell'FHS, il quale dopo due anni viene sintetizzato in un paper pubblicato nell'*European Financial Management* nel 2002[8].

Nei successivi anni la tecnica di simulazione FHS si diffonde nel settore finanziario e nell'ambito accademico, soprattutto allo scopo di generare scenari di prezzo e di volatilità. Ulteriori importanti contributi sono stati rilasciati nella prima decade del 2000, tra cui un paper di Barone-Adesi, Engle e Mancini (2008) denominato *"GARCH options in incomplete markets"*[9] dove viene analizzata la divergenza tra volatilità implicita e volatilità derivante dal modello GARCH.

1.3. IMPLEMENTAZIONE

L'idea di base di Barone-Adesi e Giannopoulos consiste nell'ancorare i cambiamenti dei prezzi degli *assets* ai livelli correnti di volatilità. La variabilità degli *assets* viene simulata tenendo conto della volatilità degli stessi e dei rendimenti estratti dal campione. Dunque, l'obiettivo del modello è quello di catturare la relazione vigente tra i vari *assets* mediante un'estrazione randomica sul dataset dei rendimenti passati: sono i dati storici che contengono l'informazione che lega gli *assets* e questo è il principio fondamentale che consente di evitare la stima

[8] Giovanni Barone-Adesi, Kostas Giannopoulos and Les Vosper (2002), "Backtesting derivative portfolios with filtered historical simulation", *European Financial Management*, **8**, 1, 31-58

[9] Giovanni Barone-Adesi, Robert Engle1 and Loriano Mancini (2008), "GARCH options in incomplete markets", Review of Financial Studies, 21, 1223 – 1258

della matrice varianza-covarianza[10]. Per simulare il valore del portafoglio per i successivi *H* giorni, è necessario estrarre *S* rendimenti passati dal dataset. Ogni valore corrisponde perciò ad un campione di rendimenti degli *assets* in uno specifico giorno ed ognuno di essi verrà poi utilizzato in modo iterativo per costruire la volatilità simulata del portafoglio associata ad ogni giorno mediante il modello GARCH.

1.3.1 Simulazione per un solo *asset*

La tecnica FHS risulta essere una generalizzazione della simulazione storica, ma a differenza di quest'ultima impone dei vincoli distributivi ai dati a causa dell'utilizzo dei modelli ARMA e GARCH[11]. I rendimenti storici però non possono essere utilizzati come input nella simulazione a meno che non siano i.i.d. (indipendenti e identicamente distribuiti). Per far ciò, è strettamente necessario rimuovere qualsiasi forma di correlazione seriale, tenendo in esplicita considerazione gli eventuali clusters di volatilità presenti nel dataset. Per perseguire il primo obiettivo è spesso sufficiente aggiungere all'equazione della media condizionale un ritardo ai residui, utilizzando dunque un semplice modello

[10] L'affermazione è vera nel caso di FHS multivariato. L'utilizzo del modello FHS nella versione univariata (*portfolio freezing*) fa scomparire le informazioni relative alla correlazione tra i vari *assets* poiché sono inglobate nella serie storica univariata dei rendimenti del portafoglio.

[11] È opportuno precisare che i vincoli distributivi imposti dai modelli ARMA e GARCH derivano dall'implementazione della stima e per questo motivo sono assai meno stringenti rispetto a quelli dettati dai modelli parametrici in senso stretto.

ARMA(1,1) ed implementare un modello GARCH(1,1)[12] (Bollerslev[13], 1986), che rappresenta la varianza condizionale come un processo eteroschedastico autoregressivo. Quest'ultimo consente allo stesso tempo di poter investigare sulle aggregazioni di volatilità presenti nelle serie storiche. Il modello ARMA-GARCH(1,1)[14] può essere scritto come:

$$(1)\ r_t = \mu + \varphi r_{t-1} + \theta\varepsilon_{t-1} + \varepsilon_t \qquad \varepsilon_t \sim \mathrm{N}(0, \mathrm{h}_t)$$

$$(2)\ \mathrm{e}_t = u_t\sqrt{h_t} \qquad e_t \sim \mathrm{i.i.d.}(0,1)$$

$$(3)\ h_t = \omega + \alpha\varepsilon^2{}_{t-1} + \beta\, h_{t-1}$$

dove r_t *e* r_{t-1} indicano rispettivamente il rendimento dell'*asset* al tempo *t* e *t-1*, φ rappresenta il coefficiente di autoregressione, mentre θ quello di media mobile. Inoltre, il termine ε_t è la componente idiosincratica definita come un *white-noise,* mentre μ *e* ω sono costanti. L'equazione (2) definisce la relazione tra innovazione e_t e il processo standardizzato u_t, mentre l'equazione (3) considera la varianza di ε_t come una funzione lineare dei valori di varianza passati h_{t-1} e dei termini di errore al quadrato passati $\varepsilon^2{}_{t-1}$, dove $\alpha \geq 0$ e $\beta \geq 0$.

[12] Il modello GARCH consente di rendere pressoché nulla la correlazione seriale, ottenendo una situazione tale per cui $Cov\left(\frac{\varepsilon_{t+1}}{\sigma_{t+1}}, \frac{\varepsilon_t}{\sigma_t}\right) \cong 0$.

[13] Bollerslev T (1986), "Generalised Autoregressive Conditional Heteroskedasticity", Journal of Econometrics, 31, 307-27.

[14] L'appropriata forma dei processi ARMA e GARCH viene determinata mediante test statistici. Il modello ARMA-GARCH (1,1) viene considerato solo a scopo illustrativo.

Come affermato nel paragrafo 1.1., affinché i rendimenti possano essere depurati dall'effetto della correlazione seriale, è necessario effettuare un'operazione di *scaling*, illustrata nell'equazione (4):

(4) $\hat{e}_t = \frac{\hat{\varepsilon}_t}{\sqrt{\hat{h}_t}}$

dove i residui $\hat{\varepsilon}_t$ vengono standardizzati per la stima della volatilità giornaliera $\sqrt{\hat{h}_t}$. Mediante questa operazione i residui standardizzati sono resi i.i.d. e quindi adatti ad essere accolti come input del processo di simulazione.

Per poter generare degli scenari di prezzo e di volatilità futura, vengono estratti con reimmissione un elevato numero di residui standardizzati dalla serie storica per poi essere moltiplicati per la volatilità corrente (*re-scaling*). I valori ottenuti sono immessi nei modelli di media condizionale (ARMA) e di varianza condizionale (GARCH) consentendo la stima di prezzi e volatilità futuri mediante simulazione.

I passi necessari all'implementazione della simulazione storica filtrata possono dunque essere riassunti nel seguente modo:

a) dal vettore dei residui standardizzati v avviene un ricampionamento con reimmisione secondo la metodologia *bootstrap*:

   (5) $e^* = \{e^*{}_i = v\}$, $v = \left[e_{1,} e_{2,} \dots, e_{T,}\right]$, dove *i=1,…,n* indica il tempo di simulazione

b) i residui standardizzati del periodo vengono moltiplicati per la volatilità giornaliera stimata tramite il modello GARCH, ottenendo l'innovazione simulata:

(6) $z^*_{t+i} = e^*_i \cdot \sqrt{\hat{h}_{t+i}}$

c) l'innovazione simulata viene immessa nel modello ARMA (1,1) per poter ottenere la stima del rendimento futuro dell'*asset*: [15]

(7) $r^*_{t+i} = \mu + \varphi r^*_{t+i-1} + \theta z^*_{t+i-1} + z^*_{t+i}$, per $i \geq 2$

d) il rendimento futuro concorre alla determinazione del prezzo futuro mediante la relazione seguente[16]:

(8) $p^*_{t+i} = p^*_{t+i-1} + p^*_{t+i-1} . r^*_{t+i}$, per $i \geq 2$

e) l'innovazione standardizzata ottenuta via simulazione viene immessa nel modello GARCH (1,1) per ottenere una stima della volatilità futura dell'*asset*[17]:

(9) $\sqrt{h^*_{t+i}} = \sqrt{\omega + \alpha(z^*_{t+i-1})^2 + \beta\, h^*_{t+i-1}}$, per $i \geq 2$

[15] Nel caso in cui i=1, il rendimento viene calcolato tenendo conto del valore osservato al momento *t*:

$$r^*_{t+1} = \mu + \varphi r_t + \theta z^*_t + z^*_{t+1}$$

[16] Nel caso in cui i=1, il prezzo futuro viene calcolato tenendo conto del valore osservato al momento *t*:

$$p^*_{t+1} = p_t + p_t . r^*_{t+1}$$

[17] Nel caso in cui i = 1, la varianza condizionale viene calcolata tenendo conto del valore osservato del residuo:

$$h^*_{t+1} = \omega + \alpha \varepsilon^2_t + \beta\, h_t .$$

Replicando la procedura appena illustrata per un elevato numero di volte, è possibile determinare la distribuzione simulata del rendimento futuro del portafoglio, necessaria per poter stimare il VaR. Quest'ultimo viene semplicemente calcolato come quantile della distribuzione dei rendimenti simulati, avendo precedentemente fissato uno specifico livello di confidenza (ad es., 99%).

1.3.2 Simulazione per più *assets*

Come già detto in precedenza, l'approccio FHS di Barone-Adesi e Giannopoulos evita l'utilizzo della matrice varianza-covarianza nella stima delle misure di rischio del portafoglio. Questa tecnica, infatti, è basata sull'estrazione di un campione di rendimenti storici dal dataset in possesso, unendo la metodologia del *bootstrapping* con i modelli econometrici ARMA e GARCH. In principio, avviene l'estrazione di *S* date casuali dal dataset che indicizzano sulla scala temporale i rendimenti giornalieri i.i.d. di ogni *asset*.

Detto ciò, per ogni *asset* viene costruito un campione da cui verranno estratti i residui necessari per la simulazione della volatilità e del prezzo futuro:

$$(10) \quad e^*{}_i = \{e^*{}_{i,j} = v_i\}, \; v_i = [e_{1,} e_{2,} \dots, e_{T,}]_i$$

dove *i* indica l'i-esimo *asset* all'interno del portafoglio e *j* il giorno di simulazione futuro.

Per ogni valore *j=1,..,T* e per ogni *asset i=1,...,N* verrà estratta una data dal dataset v_i e i valori dei residui associati $e^*{}_i$ saranno conseguentemente selezionati. I prezzi e le volatilità future di ogni *asset* verranno costruiti mediante le seguenti relazioni:

$$(11) \quad \sqrt{h^*_{i,t+j}} = \sqrt{\omega_i + \alpha_i (z^*{}_{i,t+j-1})^2 + \beta_i \, h^*_{i,t+j-1}}$$

$$(12) \quad p^*{}_{i,t+j} = p^*{}_{i,t+j-1} + p^*{}_{i,t+j-1}(\mu_i + \varphi_i r^*{}_{i,t+j-1} + \theta_i z^*{}_{i,t+j-1} + z^*{}_{i,t+j})$$

dove $z^*{}_{i,t+j}$ viene stimato mediante l'equazione (5).

Dati $w = [w_1, \omega_{2,\dots}, \omega_n]$ e $p^*_{t+j} = [p^*{}_{1,t+j}, p^*{}_{2,t+j}, \dots, p^*{}_{n,t+j}]$, dove *w* rappresenta il vettore dei pesi che i vari *assets* hanno all'interno del portafoglio di riferimento e p^*_{t+j} indica il vettore dei prezzi degli *N assets* nel periodo *t + j*, il rendimento di portafoglio simulato π^*_{t+j} al tempo *t + j* è determinabile mediante la seguente operazione matriciale:

$$(13) \quad \pi^*_{t+j} = w' p^*_{t+j} .$$

Affinché possa essere ottenuta la distribuzione dei valori di portafoglio simulati, è necessario replicare la procedura sopra illustrata *S* volte. Così facendo sarà costruita la distribuzione di probabilità del portafoglio dato un orizzonte di riferimento *H*, fondamentale per poter calcolare misure di rischio, tra cui il VaR. A differenza della classica simulazione storica, il metodo FHS consente di aumentare il *range* dei valori ottenibili grazie alla procedura di *filtraggio* iniziale. Infatti, il

processo di scaling e il successivo riadattamento ai livelli di volatilità futuri previsti, consente di produrre anche rendimenti di portafoglio non necessariamente osservati nel passato. In altre parole, FHS supera il problema della frequenza potenzialmente bassa di valori estremi che caratterizza la simulazione storica pura, andando dunque a definire in modo più realistico le code della distribuzione.

L'approccio FHS può essere anche esteso all'ambito *stress testing* visto che permette di disporre della distribuzione dei rendimenti degli *assets*. Per ottenere maggiori valori estremi nelle code è necessario semplicemente aumentare il numero di simulazioni effettuate[18].

[18] Allo stesso tempo è opportuno considerare che nel caso di serie storiche brevi le code verranno popolate in modo poco variabile, dato che il campionamento avverrà a partire da un dataset ristretto.

CAPITOLO 2

ANALISI EMPIRICA

2.1 OBIETTIVO DELL'ANALISI

Dopo aver illustrato dal punto di vista teorico la metodologia *FHS*, viene ora condotta un'analisi empirica allo scopo di effettuare una valutazione critica sui risultati ottenuti. Nello specifico, la tecnica *FHS* sarà utilizzata come metodo di simulazione dei rendimenti futuri di un *equally weighted portfolio* composto da sei ETF europei. Sulla base degli scenari generati, verrà calcolato il VaR a 10 giorni e successivamente tale stima verrà comparata con il reale valore di rendimento osservato nel mercato.

È opportuno precisare che l'analisi empirica verrà condotta sulla base del metodo *FHS* univariato, il quale effettua la simulazione e le stime econometriche sui valori relativi al *portfolio frozen* e non sui singoli *assets*. Non sarà dunque necessario stimare le correlazioni tra gli *assets* attraverso la matrice varianza-covarianza poiché tali legami verranno inglobati all'interno dell'unica serie dei rendimenti del portafoglio[19].

Inoltre, la simulazione effettuata nel presente elaborato presuppone l'invarianza della composizione del portafoglio durante l'orizzonte di analisi. Infatti, il metodo *FHS* univariato si basa su un portafoglio statico lungo l'arco temporale di simulazione poiché le decisioni degli *asset* manager vengono prese a monte del processo di generazione degli scenari futuri.

[19] Ciò che rende particolarmente efficace la tecnica *FHS* è proprio l'assenza di stima della matrice varianza-covarianza, particolarmente problematica nei metodi parametrici.

2.2 RACCOLTA ED ANALISI ESPLORATIVA DEI DATI

L'analisi empirica condotta nel presente elaborato si basa su un *equally weighted portfolio*[20] composto da sei ETF azionari europei. I dati storici giornalieri relativi agli *assets* appena menzionati sono stati estratti dalla piattaforma *Refinitiv*[21] e successivamente sono stati gestiti ed analizzati utilizzando il linguaggio di programmazione *Python 3.9*. Nello specifico, per ogni ETF è stata importata la serie storica giornaliera dei prezzi di chiusura, in seguito gestita mediante la libreria *pandas*.

La composizione del dataset viene riassunta nella Tabella 1:

Tabella 1 – Descrizione del dataset

ETF	INDICE DI RIFERIMENTO	SOCIETÀ EMITTENTE	PERIODO DI RIFERIMENTO
iShares Core MSCI Europe UCITS ETF EUR (Acc)	MSCI Europe Index	iShares III plc	10/10/2012 –10/10/2022
Invesco EURO STOXX 50 UCITS ETF Acc	EURO STOXX 50 Index	Invesco Markets plc	30/06/2014 -10/10/2022
iShares MSCI Europe Financials ETF	MSCI Europe Financials Index	iShares III plc	10/10/2012 –10/10/2022
SPDR MSCI Europe Industrials UCITS ETF	MSCI Europe Industrials 35/20 Capped Index	SsgA SPDR ETFs Europe II plc	09/12/2014 – 10/10/2022

[20] Per *equally weighted portfolio* si intende il portafoglio caratterizzato dall'uguaglianza dei fattori di ponderazione per tutti gli *assets* che compongono il portafoglio.

[21] *Refinitiv* è un provider di dati finanziari americano-britannico fondato a Londra nel 2018. Attualmente è controllato dalla *London Stock Exchange Group*.

Vanguard FTSE Developed Europe UCITS ETF EUR	FTSE Developed Europe Index	Vanguard Funds PLC	22/05/2013 – 10/10/2022
Xtrackers Stoxx Europe 600 UCITS ETF 1C	STOXX Europe 600 Index	Xtrackers	10/10/2012 – 10/10/2022

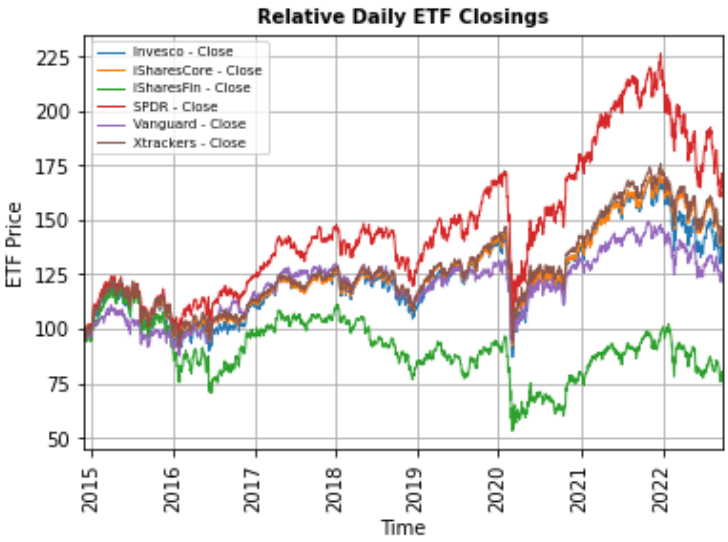
Fonte: elaborazione propria

Come visibile dalla tabella, le serie estratte da *Refinitiv* sono caratterizzate da differenti profondità storiche, imputabili alla disomogeneità delle date di lancio e ad aspetti tecnici della piattaforma gestita dal provider. Affinché gli *assets* possano essere utilizzati per il metodo FHS univariato, è necessario uniformare la lunghezza delle serie storiche. Per far ciò, è stata effettuata un'intersezione tra i vari datasets sulla base della variabile *"Exchange Date"* ed è stata applicato il metodo "*backfill"*[22] nel caso di osservazioni mancanti. Il dataset finale è composto da 1953 osservazioni per il periodo di riferimento 09/12/2014 – 10/10/2022. Nella Figura 1 viene mostrata l'andamento storico dei prezzi normalizzati[23] degli *assets* del portafoglio.

[22] Il metodo *"backfill"* consiste nell'utilizzo dell'osservazione precedente nel caso di indisponibilità di un dato (c.d. NaN).

[23] Nel presente caso la normalizzazione dei prezzi degli *assets* considera come prima osservazione (09/12/2014) il valore di prezzo 100.

Figura 1 – Prezzi di chiusura storici degli ETF



Fonte: elaborazione propria

È possibile notare come l'*asset "iShares MSCI Europe Financials"* abbia fornito la peggior performance storica tra i vari ETF. La ragione è imputabile principalmente al forte crollo dei titoli finanziari durante il 2016 e 2020, periodo in cui è aumentato il clima di incertezza nei mercati, prima a causa della fine del *Quantitative Easing*[24] e in seguito a causa dello scoppio della pandemia. In contrasto, il titolo *"SPDR MSCI Europe Industrials UCITS ETF"* mostra una costante e sostenuta tendenza al rialzo, frenata dapprima nel 2020, a causa dello shock da Covid-19, e poi nel 2022, dove gli scenari macroeconomici e geopolitici[25] hanno causato le peggiori performance nel nuovo millennio nella gran parte delle Borse mondiali.

[24] Il *Quantitative Easing* è una politica monetaria espansiva effettuata dalla BCE e consistente nell'acquisto di titoli di stato e altri titoli finanziari allo scopo di stimolare l'attività economica e contrastare i fenomeni deflattivi.

[25] I mercati hanno fatto registrare significativi ribassi a causa dello spettro della recessione, della guerra in Ucraina, della crisi energetica, del *rally* del petrolio e dell'impennata dell'inflazione.

Dopo aver importato e uniformato sul piano temporale le serie storiche dei prezzi di chiusura degli ETF, sono stati calcolati i rendimenti logaritmici giornalieri di ogni *asset*[26], secondo la seguente relazione:

$$(14) \quad r_t = \ln\left(\frac{p_t}{p_{t-1}}\right) = \ln(1 + R_t) \,,$$

dove r_t indica il rendimento logaritmico al tempo *t*, p_t *e* p_{t-1}rappresentano i prezzi dell'*asset* al tempo *t* e *t-1* e R_t denota il rendimento semplice[27] al tempo *t*.

La Tabella 2 mostra gli ultimi cinque rendimenti giornalieri di ogni ETF, calcolati mediante l'equazione (14). Il titolo *"SPDR MSCI Europe Industrials UCITS ETF"* in quattro delle cinque osservazioni presenta rendimento logaritmico in valore assoluto maggiore rispetto agli altri assets, mentre il titolo "Vanguard FTSE Developed Europe UCITS ETF EUR" è caratterizzato da variazioni di prezzo più contenute.

Tabella 2 – Head del dataframe dei rendimenti logaritmici

Data	Invesco	iSharesCore	iSharesFin	SPDR	Vanguard	Xtrackers
07/10/2022	0.598%	0.379%	0.148%	-0.616%	0.451%	0.377%
06/10/2022	1.752%	1.242%	0.859%	2.276%	1.148%	1.303%
05/10/2022	0.476%	0.775%	1.877%	0.361%	0.232%	0.773%
04/10/2022	0.935%	0.762%	1.006%	1.125%	0.646%	0.829%
03/10/2022	-4.043%	-3.126%	-3.812%	-4.059%	-3.471%	-3.056%

Fonte: elaborazione propria

La Tabella 3 fornisce delle statistiche descrittive di base relative ad ogni serie storica, tra cui la deviazione standard. Il titolo *"iShares MSCI Europe Financials"* presenta volatilità storica annuale pari a 0.2603 e al contempo si

[26] Il calcolo dei rendimenti a partire dalla serie dei prezzi riduce la dimensione del campione di un'unità.

[27] Il rendimento semplice R_t viene calcolato mediante la seguente formula:

$$R_t = \frac{p_t - p_{t-1}}{p_{t-1}}$$

distingue per il rendimento minimo assoluto. Nonostante ciò, l'asset in questione è l'unico ETF a presentare rendimento medio aritmetico positivo. È opportuno considerare inoltre che il titolo *"SPDR MSCI Europe Industrials UCITS ETF"* ha registrato rendimento massimo giornaliero durante l'orizzonte di analisi, pari a 19.688%. Tale performance è stata ottenuta il 16 marzo 2020, durante il *rally* dei mercati a causa del diffondersi della pandemia da Covid-19.

Tabella 3- Statistiche di base relative al dataset dei rendimenti storici

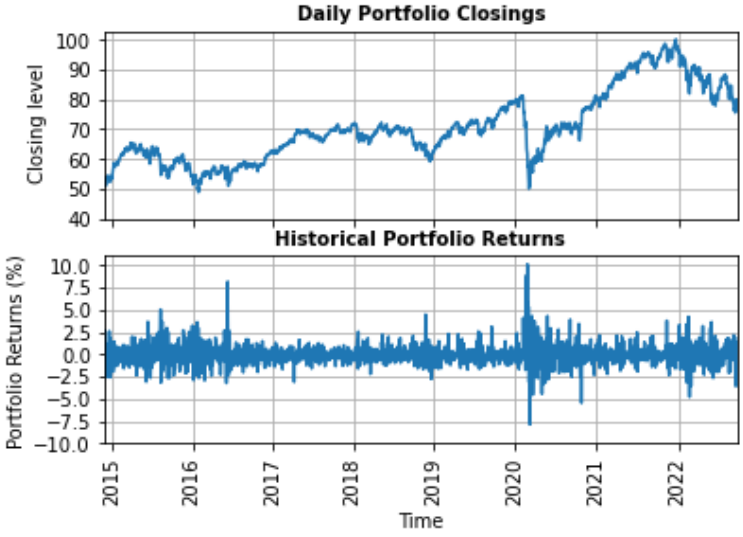
Statistica	Invesco	iSharesCore	iSharesFin	SPDR	Vanguard	Xtrackers
Media aritmetica annuale	-3.6792%	-4.4352%	3.3516%	-6.5520%	-2.6712%	-4.5360%
Deviazione Standard annuale	0.2064	0.1794	0.2603	0.2111	0.1667	0.1794
Minimo	-7.9889%	-6.982%	-10.686%	-8.259%	-7.060%	-6.354%
Percentile: 25%	-0.6534%	-0.593%	-0.762%	-0.675%	-0.611%	-0.585%
Percentile: 50%	-0.0536%	-0.073%	-0.035%	-0.070%	-0.035%	-0.068%
Percentile: 75%	0.5757%	0.488%	0.713%	0.547%	0.512%	0.486%
Massimo	11.7458%	11.815%	16.895%	19.688%	10.485%	11.659%

Fonte: elaborazione propria

Dopo aver analizzato gli assets singolarmente, viene ora costruito l'*equally weighted portfolio* secondo l'equazione (13). È necessario precisare che il vettore dei pesi w in questo caso è pari a $w = \left[\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right]$ poiché ogni asset contribuisce in maniera eguale al prezzo e al rendimento di portafoglio. Il risultato ottenuto viene illustrato nella Figura 2, la quale mostra in alto il trend storico del valore del

portafoglio, mentre in basso i rendimenti logaritmici giornalieri durante il periodo di riferimento.

Figura 2 - Prezzi e rendimenti storici del portafoglio



Fonte: Elaborazione propria

Il valore di chiusura del portafoglio oscilla durante il periodo di analisi tra €48.89 e €100.22, attestandosi ad un valore pari a €77.50 il 10/10/2020. Dal punto di vista visivo, i rendimenti storici sembrano essere caratterizzati da stazionarietà in senso debole: infatti, la gran parte delle osservazioni sono comprese nell'intervallo (-2.5%, +2.5%) e i momenti fino al secondo ordine appaiono essere costanti nel tempo. Affinché possa essere analizzata nel dettaglio la persistenza del processo stocastico in questione, vengono effettuati mediante le librerie *statmodels* e *archmodel* tre test di radice unitaria:

- Test Augmented Dickey-Fuller (*ADF*)[28]

[28] Dickey, D.A. and Fuller, W. A. (1979) "Distribution of the Estimators for Autoregressive Time Series With a Unit Root", Journal of the American Statistical Association

- Test Phillips-Perron (*PP*)[29]
- Test Kwiatkowski-Phillips-Schmidt-Shin (*KPSS*)[30].

Il test Augmented Dickey-Fuller (*ADF*) è un test di radice unitaria che si configura come una generalizzazione del test d'integrazione Dickey-Fuller (*DF*). Quest'ultimo consiste in un semplice test t di azzeramento del parametro ρ del seguente modello:

$$\Delta y_t = \rho y_{t-1} + u_t \quad (15)$$ [31]

dove u_t è un *white noise* e ρ= φ-1. Il processo è stazionario soltanto se $\rho < 0$. Affinchè il test di radice unitaria sia generalizzabile ed applicabile anche a modelli con caratteristiche diverse di persistenza, viene considerato il seguente processo AR(p):

$$y_t = \varphi_1 y_{t-1} + \cdots + \varphi_p y_{t-p} + u_t \quad (16)$$

dove u_t è un white noise e $\varphi_1, \dots, \varphi_p$ sono i parametri che legano la variabile dipendente y_t con ogni corrispondente ritardo. L'equazione (16) viene riparametrizzata mediante la scomposizione di Beveridge-Nelson[32], ottenendo la seguente formulazione:

---

[29] Phillips, P. C. B. and Perron, P (1988) "Testing for a Unit Root in Time Series Regression", Biometrika, 75: 335-346

[30] Kwiatkowski, D., Phillips, P.C.B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. Journal of Econometrics, 54: 159-178.

[31] Il modello presente nell'equazione (15) corrisponde ad un processo autoregressivo di primo ordine, espresso dalla seguente relazione:

$$y_t = \varphi y_{t-1} + u_t$$

dove φ è il parametro che lega la variabile dipendente y_t con il suo ritardo di ordine 1.

[32] Beveridge S. and Nelson C.R. (1981) "A new approach to decomposition of economic time series into permanent and transitory components with particular attention to measurement of the 'business cycle", Journal of Monetary Economics, Volume 7, Issue 2, 1981, Pages 151-174

$$\Delta y_t = \rho y_{t-1} + \gamma_1 \Delta y_{t-1} + \cdots + \gamma_p \Delta y_{t-p} + u_t \quad (17)$$

dove $\rho = (\varphi_1 + \cdots + \varphi_p) - 1$ e $\gamma_i = -(\varphi_{i+1} + \cdots + \varphi_p)$. Il test *ADF* consiste nell'azzeramento del parametro ρ presente nell'equazione (17), basato dunque sulla statistica

$$t_\rho = \frac{\hat{\rho}}{\sqrt{\widehat{var(\hat{\rho})}}}. \quad (18)$$

Il test, dunque, ha come ipotesi nulla la non stazionarietà del processo e la stazionarietà come ipotesi alternativa.

Il test di Phillips-Perron (*PP*) consiste sostanzialmente in una versione della statistica test di Dickey-Fuller, dove la varianza $Var(\hat{\rho})$ viene sostituita dalla varianza asintotica dello stimatore ottenuta mediante la seguente relazione:

$$AVar(\hat{\rho}) = \sum_{i=-m}^{m} k_i \widehat{\tau_i} \quad (19)$$

dove $\widehat{\tau_i}$ è l'i-esima autocovarianza stimata del residuo u_t, k_i è il peso assegnato a ciascuna covarianza e *m* è il parametro di troncamento. La determinazione dei pesi segue lo schema della "*Tendina di Bartlett*" espresso nella seguente espressione:

$$k_i = \begin{cases} 1 - \frac{|i|}{m+1} & se\ |i| \le m \\ 0 & altrove \end{cases}. \quad (20)$$

Il test *PP* consente di testare la presenza di una radica unitaria anche in presenza di dinamiche più generali rispetto al processo AR(p), utilizzato nel test *ADF*. Inoltre, a differenza di quest'ultimo, il test *PP* è robusto all'eteroschedasticità dell'errore u_t e non richiede alcuna specificazione del numero dei ritardi della variabile dipendente y_t.

Il test Kwiatkowski-Phillips-Schmidt-Shin (*KPSS*) è un test di radice unitaria non parametrico che considera la seguente equazione:

$$y_t = bt + \mu_t + \epsilon_t \quad (21)$$

dove μ_t si distribuisce come un *Random Walk*[33], t indica il trend lineare e ϵ_t è un *white noise* con media nulla e varianza σ_ε^2. Il test *KPSS* si configura come un test di azzeramento della varianza σ_ε^2, dove la struttura delle ipotesi, inversa a quella dei test *ADF* e *PP*, viene indicata dalla seguente formulazione:

$$(22) \quad \begin{cases} H_0: \sigma_\varepsilon^2 = 0 \ \rightarrow \ \mathrm{y}_t \text{ è } un\ processo\ stazionario \\ H_1: \sigma_\varepsilon^2 = 0 \ \rightarrow \ \mathrm{y}_t \text{ è } un\ processo\ non\ stazionario \end{cases}$$

Gli output dei test di radice unitaria condotti vengono mostrati nelle Figure 3-4-5. È possibile notare come tutti i test rifiutino l'ipotesi di stazionarietà della serie storica dei rendimenti del portafoglio per ogni diverso nucleo deterministico.

Figura 3- Test Augmented Dickey-Fuller (ADF): output

Results of Augmented Dickey-Fuller Test:			
	Constant	Constant and Linear Trend	
Test Statistic	-42.357522	-42.350251	
p-value	0.0	0.0	
#Lags Used	0	0	
Number of Observations Used	1951	1951	
Critical Value (1%)	-3.433706	-3.963418	
Critical Value (5%)	-2.863023	-3.412743	
Critical Value (10%)	-2.567559	-3.128376	
Result	No unit root	No unit root	
	Constant and Quadratic Trend		No Trend
Test Statistic	-42.342302		-42.363753
p-value	0.0		0.0
#Lags Used	0		0
Number of Observations Used	1951		1951
Critical Value (1%)	-4.377079		-2.566887
Critical Value (5%)	-3.83542		-1.941139
Critical Value (10%)	-3.555137		-1.616685
Result	No unit root		No unit root

Fonte: Elaborazione propria

[33] Nello specifico, μ_t si basa sulla seguente espressione:

$$\mu_t \ = \ \mu_{t-1} + u_t$$

Figura 4 - Test KPSS: output

```
Results of KPSS Test:
                        Constant Constant and Linear Trend
Test Statistic              0.040657                0.032946
p-value                          0.1                     0.1
#Lags Used                         3                       3
Critical Value (10%)           0.347                   0.119
Critical Value (5%)            0.463                   0.146
Critical Value (2,5%)          0.574                   0.176
Critical Value (1%)            0.739                   0.216
Result                  No unit root            No unit root
```

Fonte: Elaborazione propria

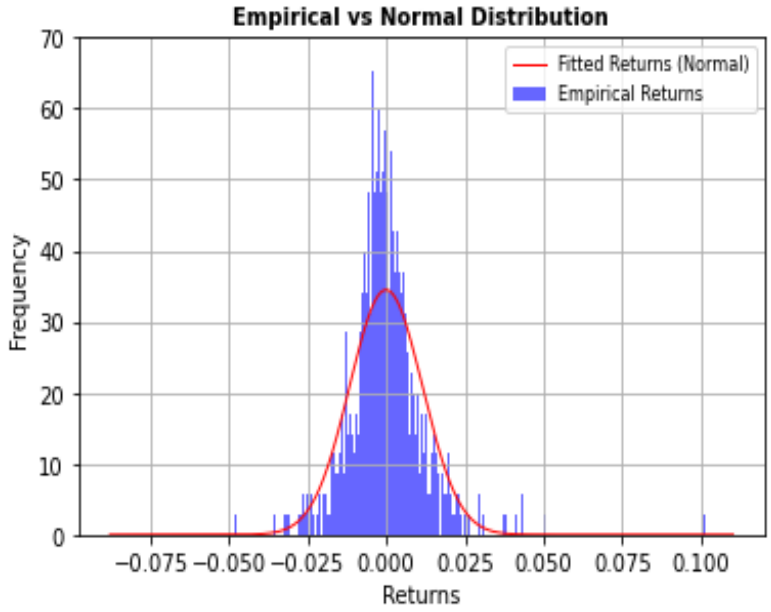
Figura 5- Test Phillips-Perron: output

```
     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -42.331
P-value                         0.000
Lags                               26
-------------------------------------

Trend: No Trend
Critical Values: -2.57 (1%), -1.94 (5%), -1.62 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.

     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -42.323
P-value                         0.000
Lags                               26
-------------------------------------

Trend: Constant
Critical Values: -3.43 (1%), -2.86 (5%), -2.57 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.

     Phillips-Perron Test (Z-tau)
=====================================
Test Statistic                -42.315
P-value                         0.000
Lags                               26
-------------------------------------

Trend: Constant and Linear Time Trend
Critical Values: -3.96 (1%), -3.41 (5%), -3.13 (10%)
Null Hypothesis: The process contains a unit root.
Alternative Hypothesis: The process is weakly stationary.
```

Fonte: Elaborazione propria

Dopo aver effettuato un'analisi relativa alla stazionarietà in senso debole, viene ora condotta un'indagine relativa all'ipotesi di normalità della serie dei rendimenti logaritmici. La Figura 6 confronta la frequenza della distribuzione empirica dei rendimenti del portafoglio, raffigurata mediante un istogramma di 100 barre, con quella della normale, avente come parametri la media dei rendimenti μ pari a -0.0122% e la varianza σ^2 pari a 0.0116.

Figura 6 - Fitting della distribuzione normale sulla frequenza dei rendimenti

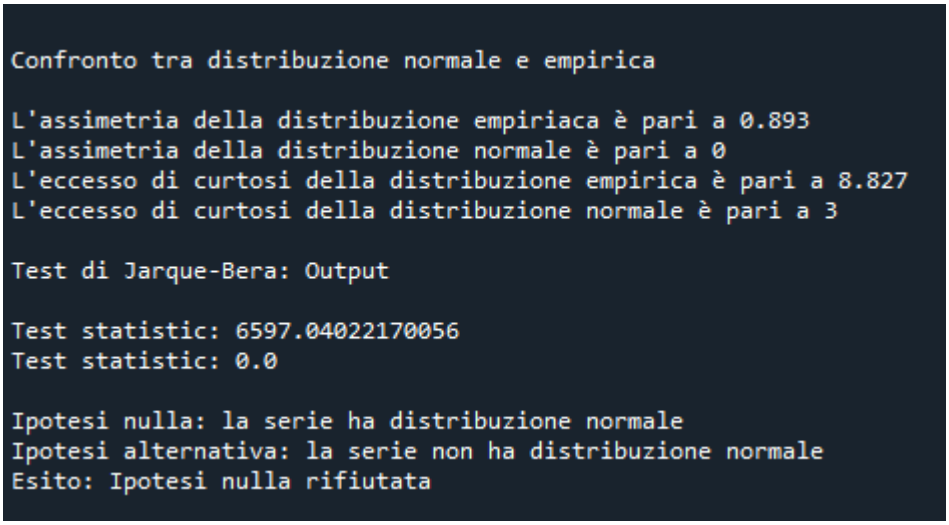


Fonte: Elaborazione propria

L'osservazione della Figura 6 consente già visivamente di concludere che la distribuzione normale non riesce ad approssimare in modo corretto quella dei rendimenti empirici. Questa affermazione viene confermata dapprima dalle statistiche di curtosi e di asimmetria delle due distribuzioni e in seguito dal test

statistico Jarque-Bera[34], illustrati nella Figura 7. Quest'ultima mostra l'output dell'analisi, svolta mediante la libreria *scipy*:

Figura 7 - Analisi di normalità dei rendimenti



Fonte: Elaborazione propria

Inoltre, è opportuno precisare che l'esito dell'analisi di normalità appena condotta conferma come l'ipotesi di approssimazione della distribuzione empirica con una distribuzione gaussiana sia decisamente fuorviante e conduca ad errori rilevanti. Per questo motivo la tecnica *FHS* ha migliori *chance* di adattarsi al campione empirico esaminato, non basandosi su alcuna stringente ipotesi distributiva[35].

Si conclude l'analisi esplorativa dei dati con un'indagine sulla correlazione seriale, effettuata mediante l'*AutoCorrelation Function (AFC)* e la *Partial AutoCorrelation Function (PAFC)*. L'*AFC* studia la correlazione tra l'osservazione

[34] Jarque, C. and Bera, A. (1980) "Efficient tests for normality, homoscedasticity and serial independence of regression residuals", 6 Econometric Letters 255-259.

[35] Anche la tecnica *FHS* contempla ipotesi distributive, necessarie all'interno della stima GARCH, ma queste sono nettamente meno stringenti rispetto a quelle presenti nei modelli parametrici.

r_t e la r_{t-k} , dove $k = 1, 2, 3, \ldots$ indica il ritardo, ed è sintetizzata mediante la seguente relazione:

$$(23) \quad \rho_k = \frac{COV(r_t, r_{t-k})}{\sqrt{Var(r_t)\, Var(r_{t-k})}}$$

dove ρ_k indica l'autocorrelazione relativa al ritardo di ordine k.

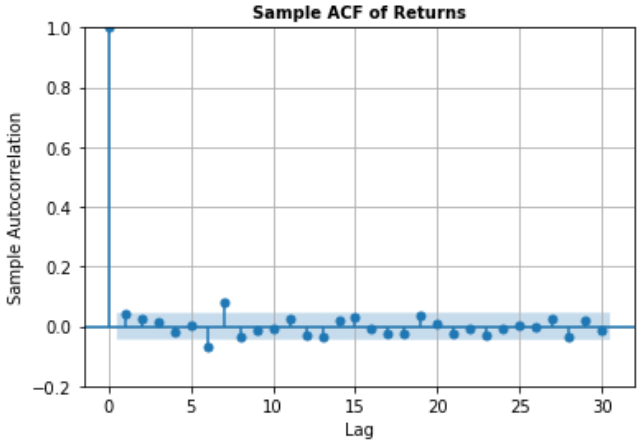
La *PAFC* piuttosto è una misura di correlazione condizionale. Essa mostra la correlazione tra due osservazioni al netto della correlazione osservata tra ritardi più brevi. In sintesi, la correlazione parziale per ogni ritardo è la correlazione tra due osservazioni dopo aver depurato l'effetto delle correlazioni pregresse. La *PACF* viene mostrata nella seguente relazione:

$$(24) \quad \gamma_k = \frac{COV(r_t, r_{t-k} \mid r_{t-1}, r_{t-2}, .., r_{t-k+1})}{\sqrt{Var(r_t \mid r_{t-1}, r_{t-2}, .., r_{t-k+1})\, Var(r_{t-k} \mid r_{t-1}, r_{t-2}, .., r_{t-k+1})}}$$

dove $\gamma_k$indica l'autocorrelazione parziale di ordine k.

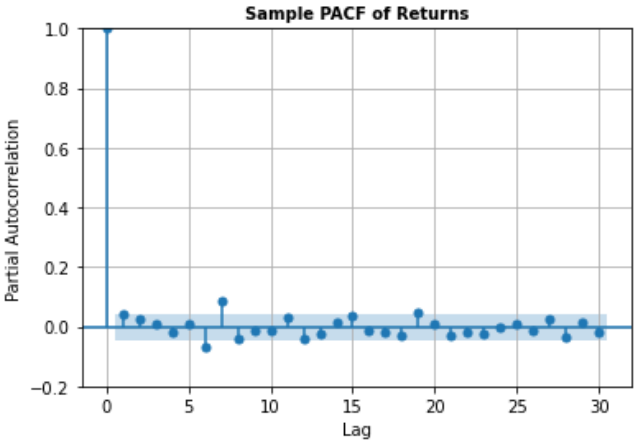
Le Figure 8-9 mostrano l'*ACF* e la *PACF* calcolate sulla serie storica dei rendimenti di portafoglio, considerando il ritardo *k = 1,2,…,30* nell'asse delle ascisse. È possibile concludere che entrambe le tipologie di autocorrelazioni risultino contenute e dunque non sembrerebbe essere presente il fenomeno della correlazione seriale.

Figura 8 - Funzione di autocorrelazione dei rendimenti di portafoglio



Fonte: Elaborazione propria

Figura 9 - Funzione di autocorrelazione dei rendimenti di portafoglio



Fonte: Elaborazione propria

È necessario precisare inoltre che gli effetti di un'eventuale elevata correlazione seriale tra i rendimenti della serie verrebbero comunque gestiti dal processo di filtraggio, effettuato all'interno della tecnica *FHS*. Infatti, come precisato nel capitolo 1, lo *scaling* della serie rende le osservazioni serialmente indipendenti e adatte ad essere utilizzate come input del processo di simulazione.

2.3 ANALISI PARAMETRICA

Dopo aver effettuato l'analisi esplorativa del dataset, vengono ora condotte le stime parametriche, rispettivamente mediante i modelli ARMA(p,q)[36] e GARCH(m,s).

2.3.1 Modello ARMA

Il modello ARMA(p,q) studia la media condizionale della variabile dipendente (nel caso del presente elaborato r_t) rispetto al set informativo I_{t-1}. Affinché ciò sia possibile, l'ARMA(p,q) comprende sia i processi AutoRegressivi

[36] Il modello ARMA(p,q) viene anche denominato modello di *Box-Jenkins* dal nome dei suoi inventori George Box e Gwilym Jenkins.

(AR) che i processi *Moving Average* (MA). Dunque, nel modello ARMA(*p,q*) le caratteristiche di persistenza del processo stocastico vengono studiate mediante i momenti primi dello stesso. In generale, il processo ARMA(*p,q*) può essere definito dalla seguente equazione:

(25) $ARMA(p,q)$: $A(L)y_t = \mu + C(L)\varepsilon_t$

dove p rappresenta l'ordine del polinomio A(L) (componente autoregressiva) mentre q indica l'ordine del polinomio C(L) (componente a media mobile). Allo stesso tempo y_t è la variabile indipendente, μ indica l'intercetta e ε_t rappresenta un processo *white noise*.

Come illustrato nel Capitolo 1, ai fini della simulazione FHS il modello di *Box-Jenkins* è necessario affinché possano essere simulati i valori di rendimento e prezzo futuri. Nel paper originale di Barone-Adesi e Giannopoulos, gli ordini *p* e *q* del modello in questione sono entrambi pari ad 1. Infatti, l'obiettivo principale del contributo era quello di mostrare come fosse possibile simulare le traiettorie dei prezzi di portafoglio senza imporre vincoli distributivi alla serie storica dei rendimenti, evitando la stima della matrice varianza-covarianza. La scelta ottimale degli ordini del modello ARMA(*p,q*) era dunque secondaria e per questo motivo, allo scopo anche di facilitare la trattazione e l'implementazione della simulazione, *p* e *q* sono stati fissati ad un valore pari a 1.

Nel presente elaborato è stato deciso invece di determinare gli ordini ottimali di *p* e *q* allo scopo di consentire una stima parametrica efficace, con dei correlati vantaggi anche nel processo di simulazione successivo. Affinché ciò sia possibile, è necessario osservare le *Figure 8-9*, mostranti rispettivamente le funzioni di autocorrelazione (ACF) e autocorrelazione parziale (PACF). Queste ultime sono utili per la determinazione dell'ordine massimo del polinomio autoregressivo e del polinomio relativo alla componente a media mobile. Infatti, il valore massimo di *p* è di norma uguale all'ordine dell'ultimo valore della funzione

di autocorrelazione che supera la banda di confidenza costruita intorno al valore 0. Quanto appena espresso vale anche per il valore massimo di q, ma osservando piuttosto la funzione di autocorrelazione parziale. Seppur ciò garantisca una delimitazione efficiente delle possibili combinazioni tra p e q, è opportuno tenere conto di come l'elevatezza dei parametri in questione conduca ad una maggiore complessità di stima e a tempi di elaborazione più lunghi. L'analista deve dunque considerare il *trade off* tra efficacia del modello e parsimonia nella scelta degli ordini.

Nell'analisi del presente elaborato, è evidente come le *Figure 8-9* indichino dei valori massimi di p e q pari a 7. Infatti, L'ACF e la PACF fuoriescono dalla banda di oscillazione in presenza degli ordini 5, 6 e 7. Sembra dunque appropriato associare a p e q l'ordine massimo pari a 7 di modo che i principi di parsimonia e di efficacia siano bilanciati correttamente.

Dopo aver determinato i valori massimi degli ordini del modello ARMA(p,q), è necessario selezionare la combinazione ottima tra p e q tra le 49 possibili, considerando che il modello ottimo è quello che garantisce la minimizzazione del maggior numero di Criteri Informativi (IC).

I Criteri Informativi sono strumenti utili in fase di specificazione del modello poiché consentono una misurazione del sopracitato *trade off* tra capacità esplicativa della stima e parsimonia nel numero di variabili. A tal fine è opportuno considerare come l'aumento di variabili comporti una non diminuzione della funzione di log-verosimiglianza associata al modello. Sulla base di ciò si basano i Criteri Informativi: essi, infatti, sono funzioni direttamente proporzionali al numero delle variabili k e inversamente proporzionali al valore della log-verosimiglianza stimata $l(\hat{\theta})$. In particolare:

(26) $$IC = f(\, l(\hat{\theta}), k\,)$$

I Criteri Informativi utilizzati come criteri decisionale nel presente elaborato sono i seguenti:

- Criterio di Akaike (AIC): $AIC = -2l(\hat{\theta}) + 2k$;
- Criterio di Schwarz (BIC): $BIC = -2l(\hat{\theta}) + k\ logT$;
- Criterio di Hannah e Quinn (HQC): $HQC = -2l(\hat{\theta}) + 2k\ log\ log\ T$.

Dal punto di vista empirico, può accadere che i Criteri Informativi producano risultati contrastanti: è comune, infatti, la tendenza del criterio *AIC* a preferire modelli con maggiore numero di parametri.

Nelle *Tabelle 4-5-6* vengono mostrati i valori dei Criteri Informativi ottenuti in relazione ad ogni stima condotta.

Tabella 4 - Criterio Informativo AIC per le 49 stime: in grassetto il valore minimo

p\q	1	2	3	4	5	6	7
1	-11864.6	-11862.9	-11877.1	-11876.0	-11875.5	-11880.9	-11880.6
2	-11862.9	-11863.2	-11878.0	-11880.9	-11879.4	-11879.5	-11878.6
3	-11877.3	-11865.1	**-11891.2**	-11881.5	-11876.0	-11889.3	-11888.6
4	-11876.7	-11881.2	-11881.3	-11881.2	-11870.3	-11889.1	-11887.8
5	-11875.5	-11880.1	-11876.2	-11877.4	-11875.5	-11890.9	-11886.9
6	-11879.5	-11878.1	-11889.3	-11881.4	-11887.0	-11882.7	-11882.1
7	-11879.6	-11877.7	-11888.3	-11889.6	-11881.5	-11882.2	-11881.8

Fonte: Elaborazione Propria

Tabella 5 - Criterio Informativo BIC per le 49 stime: in grassetto il valore minimo

p\q	1	2	3	4	5	6	7
1	-11842.3	-11835.0	-11843.7	-11836.9	-11830.9	-11830.7	-11824.9
2	-11835.0	-11829.8	-11839.0	-11836.3	-11829.2	-11823.8	-11817.3
3	-11843.8	-11826.0	**-11846.5**	-11831.3	-11820.2	-11828.0	-11821.7
4	-11837.7	-11836.6	-11831.1	-11825.4	-11808.9	-11822.2	-11815.4
5	-11830.9	-11829.9	-11820.4	-11816.1	-11808.6	-11818.4	-11808.9

6	-11829.3	-11822.3	-11828.0	-11814.4	-11814.5	-11804.7	-11798.4
7	-11823.9	-11816.3	-11821.3	-11817.1	-11803.5	-11798.5	-11792.6

Fonte: Elaborazione Propria

Tabella 6 - Criterio Informativo HQC per le 49 stime: in grassetto il valore minimo

p\q	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	-11856.4	-11852.7	-11864.8	-11861.6	-11859.1	-11862.5	-11860.1
2	-11852.6	-11850.9	-11863.7	-11864.5	-11861.0	-11859.0	-11856.1
3	-11865.0	-11850.7	**-11874.8**	-11863.1	-11855.5	-11866.8	-11864.0
4	-11862.3	-11864.8	-11862.9	-11860.7	-11847.7	-11864.5	-11861.2
5	-11859.1	-11861.7	-11855.7	-11854.9	-11850.9	-11864.3	-11858.2
6	-11861.0	-11857.6	-11866.8	-11856.7	-11860.4	-11854.0	-11851.3
7	-11859.1	-11855.1	-11863.7	-11863.0	-11852.8	-11851.4	-11849.0

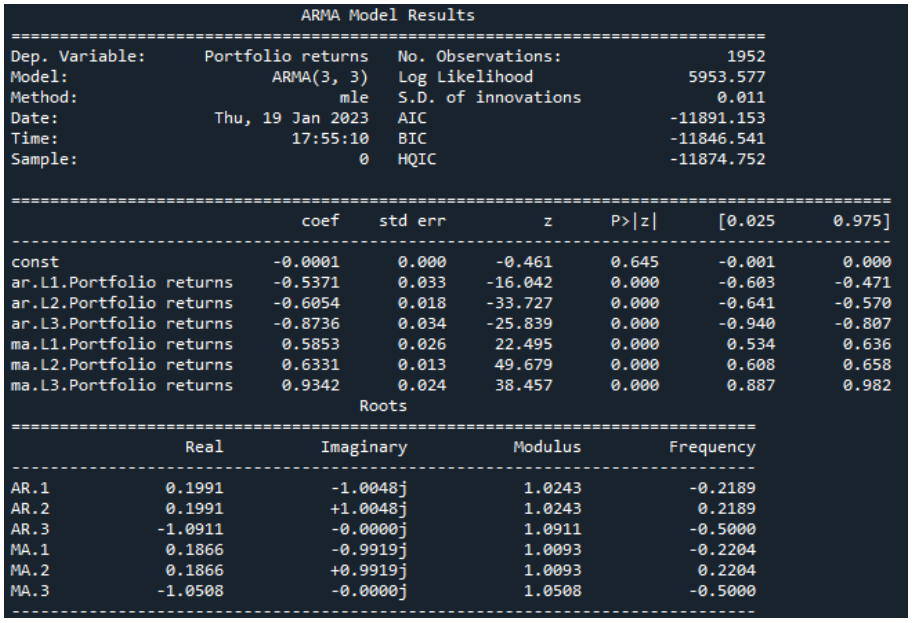
Fonte: Elaborazione Propria

Risulta evidente dagli output mostrati come il modello ARMA(3,3) minimizzi tutti i Criteri Informativi e garantisca al contempo un limitato numero di variabili.

La *Figura 10* mostra l'output relativo alla stima dell'ARMA(3,3), effettuata mediante la libreria *statmodels.* È opportuno precisare che il metodo di stima utilizzato coincide con la massimizzazione della funzione di verosimiglianza esatta mediante il filtro di Kalman[37] (nella *Figura 10* indicata con la sigla "*mle*"). È possibile notare come tutti i coefficienti associati alla componente autoregressiva e a media mobile siano statisticamente significativi, visti i relativi *p-value* prossimi allo zero. Allo stesso tempo risulta evidente la relazione negativa tra la variabile dipendente r_t ed i propri ritardi; al contrario, la componente a media mobile contribuisce in modo positivo al rendimento al tempo t.

[37] Si veda Hamilton, J.D. "Time Series Analysis". Princeton, 1994, Cap 13

Figura 10 - Output del modello ARMA(3,3)



ARMA Model Results

Dep. Variable:	Portfolio returns	No. Observations:	1952
Model:	ARMA(3, 3)	Log Likelihood	5953.577
Method:	mle	S.D. of innovations	0.011
Date:	Thu, 19 Jan 2023	AIC	-11891.153
Time:	17:55:10	BIC	-11846.541
Sample:	0	HQIC	-11874.752

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0001	0.000	-0.461	0.645	-0.001	0.000
ar.L1.Portfolio returns	-0.5371	0.033	-16.042	0.000	-0.603	-0.471
ar.L2.Portfolio returns	-0.6054	0.018	-33.727	0.000	-0.641	-0.570
ar.L3.Portfolio returns	-0.8736	0.034	-25.839	0.000	-0.940	-0.807
ma.L1.Portfolio returns	0.5853	0.026	22.495	0.000	0.534	0.636
ma.L2.Portfolio returns	0.6331	0.013	49.679	0.000	0.608	0.658
ma.L3.Portfolio returns	0.9342	0.024	38.457	0.000	0.887	0.982

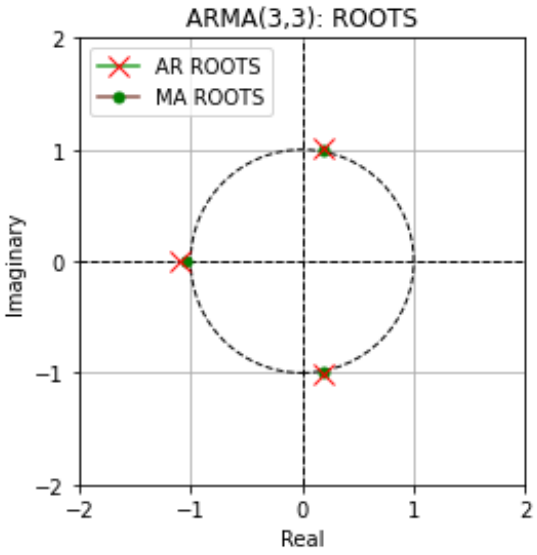
Roots

	Real	Imaginary	Modulus	Frequency
AR.1	0.1991	-1.0048j	1.0243	-0.2189
AR.2	0.1991	+1.0048j	1.0243	0.2189
AR.3	-1.0911	-0.0000j	1.0911	-0.5000
MA.1	0.1866	-0.9919j	1.0093	-0.2204
MA.2	0.1866	+0.9919j	1.0093	0.2204
MA.3	-1.0508	-0.0000j	1.0508	-0.5000

Fonte: Elaborazione Propria

Nella parte inferiore della *Figura 10* vengono inoltre mostrate le radici associate ai polinomi A(L) e C(L), scindendo i valori ottenuti tra componente reale e immaginaria. Una rappresentazione grafica delle stesse viene mostrata nella *Figura 11*, la quale colloca all'interno di un piano cartesiano i corrispondenti valori.

Figura 11 - Radici del modello ARMA(3,3)



Fonte: Elaborazione Propria

È possibile notare come tutte le radici cadano, seppur non così nettamente, al di fuori del cerchio unitario. Ciò conferma i risultati dei test di stazionarietà condotti nel paragrafo precedente, i quali rigettano l'ipotesi che il processo analizzato sia *I(1)* (integrato di ordine 1)[38].

Terminata la fase di stima, viene ora condotta l'analisi diagnostica[39] relativamente al modello di Box-Jenkins. Nello specifico, il controllo effettuato sul modello ARMA(3,3) verte sui seguenti fenomeni:

---

[38] Un processo integrato di ordine 1 è un processo che va differenziato una volta affinché sia reso stazionario. Un esempio di processo I(1) è il *random walk*, rappresentato dalla seguente equazione:

$$y_t = y_{t-1} + \varepsilon_t$$

Il processo *random walk* se differenziato una volta si trasforma in un processo *white noise*, per definizione stazionario. Infatti:

$$\Delta y_t = y_t - y_{t-1} = \varepsilon_t$$

[39] Un'analisi diagnostica completa presupporrebbe anche lo studio della normalità dei residui, ma ai fini del metodo di simulazione *FHS* non risulta necessario investigare sulla distribuzione degli stessi. Per questo motivo, l'analisi diagnostica verte esclusivamente sull'autocorrelazione dei residui e sull'eteroschedasticità condizionale.

- *autocorrelazione dei residui*: consiste nella correlazione osservata tra i residui al tempo *t* e quelli al tempo *t-k* (con *k>0*). Un modello caratterizzato da autocorrelazione dei residui conserva memoria nei residui e dunque risulta essere *mispecificato*, non adatto cioè a rappresentare in modo corretto le relazioni contenute nella variabile dipendente;
- *eteroschedasticità condizionale*: si configura come fenomeno di variabilità della varianza dei residui condizionale al set informativo I_{t-1}. In sintesi, la varianza condizionale dei residui $Var(\varepsilon_t|I_{t-1})$ non risulta essere costante (in tal caso si parla di *omoschedasticità condizionale*), ma piuttosto assume valori diversi nel tempo. Nel caso in cui venga appurata la variabilità della varianza degli errori, è necessario specificare una legge di moto per la varianza condizionale[40], allo scopo di evitare una perdita di efficienza della stima.

Per poter indagare sull'autocorrelazione dei residui, è stato effettuato un test di Ljung-Box (LB), il quale stabilisce la presenza di autocorrelazione fino ad un ordine massimo *k*, stabilito dall'analista. Il test in questione viene applicato frequentemente ai modelli ARMA(*p,q*) ed è basato sulla seguente statistica test:

$$(27) \quad LB = T(T+2)\sum_{i=1}^{k}\frac{\hat{\rho}_i^2}{T-i} \sim \chi^2_{k-(p+q)}$$

dove *T* indica la lunghezza temporale della serie storica, *k* l'ordine massimo di autocorrelazione testato, $\widehat{\rho_i}$ l'i-esima autocorrelazione stimata e $\chi^2_{k-(p+q)}$[41] una distribuzione chi-quadrato con *k-(p+q)* gradi di libertà.

[40] La legge di moto utilizzata in questo elaborato per modellare la varianza condizionale degli errori è rappresentata dal processo GARCH(1,1).

[41] Tenendo conto che il parametro "gradi di libertà" risulta essere sempre positivo, è evidente come l'ordine minimo *k* del test LB debba essere pari a *p+q+1*.

Al fine di testare l'autocorrelazione dei residui del modello ARMA(3,3), sono stati condotti vari test LB, ognuno dei quali considera un ordine di autocorrelazione $k = 7, \dots, 30$. L'output dei test viene mostrato nella *Tabella 7*, la quale evidenzia per ogni ordine *k* la relativa statistica test, accompagnata dal *p-value* associato e dall'esito finale.

Tabella 7 - Output del Test di Ljung-Box

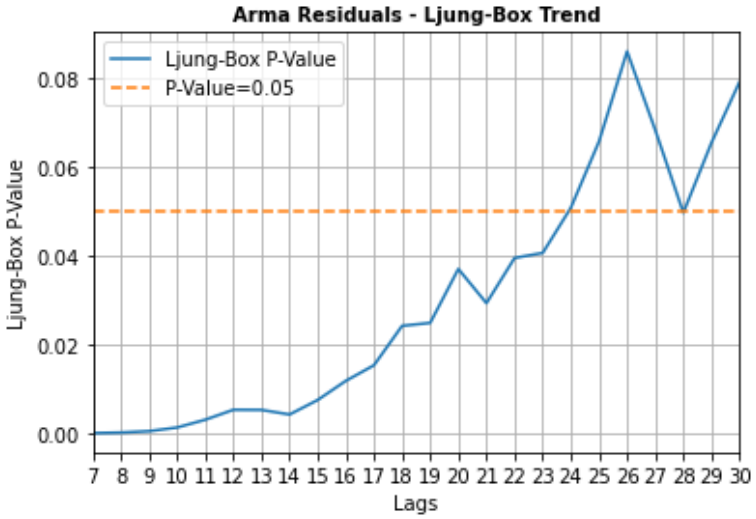
k	Statistica test	*P-value* del test	Esito del test
7	17.34913	3.11E-05	Autocorrelazione
8	17.39827	0.000167	Autocorrelazione
9	17.84039	0.000474	Autocorrelazione
10	17.89103	0.001296	Autocorrelazione
11	17.92545	0.003041	Autocorrelazione
12	18.40986	0.005286	Autocorrelazione
13	20.14849	0.005258	Autocorrelazione
14	22.3977	0.00423	Autocorrelazione
15	22.47295	0.007495	Autocorrelazione
16	22.72723	0.011799	Autocorrelazione
17	23.4407	0.015309	Autocorrelazione
18	23.44451	0.024181	Autocorrelazione
19	24.75685	0.024841	Autocorrelazione
20	24.76285	0.036982	Autocorrelazione
21	26.93278	0.029289	Autocorrelazione
22	27.18623	0.039459	Autocorrelazione
23	28.38984	0.040584	Autocorrelazione
24	28.8101	0.05075	Assenza di autocorrelazione
25	29.02188	0.065642	Assenza di autocorrelazione
26	29.09298	0.085945	Assenza di autocorrelazione

27	31.33406	0.068275	Assenza di autocorrelazione
28	33.9553	0.049641	Autocorrelazione
29	33.98452	0.065382	Assenza di autocorrelazione
30	34.32293	0.079091	Assenza di autocorrelazione

Fonte: Elaborazione Propria

Dall'output è possibile sostenere la presenza di autocorrelazione seriale fino all'ordine 24, sintomo di un forte fenomeno di persistenza tra i residui del modello stimato. Quanto appena affermato, è ancora più evidente dalla *Figura 12*, la quale mostra il *p-value* del test di Ljung-Box al variare dell'ordine *k*.

Figura 12 - Trend del Test di Ljung-Box: residui del modello ARMA(3,3)



Fonte: Elaborazione Propria

È opportuno precisare come l'effetto negativo del fenomeno di autocorrelazione seriale venga ridotto drasticamente dal processo di *scaling* dei residui. Lo scopo della standardizzazione è infatti quello di rendere i residui indipendenti e identicamente distribuiti, prevenendo una *mispecificazione* del modello ARMA(*p,q*) che impatterebbe negativamente sul processo di simulazione successivo.

2.3.2 Modello GARCH

Dopo aver definito il modello di media condizionale della variabile dipendente r_t, viene ora specificata la legge di moto della varianza condizionale mediante il processo GARCH(*m,s*), il quale studia la persistenza attraverso i momenti secondi.

I processi GARCH(*m,s*) (*Generalized AutoRegressive Conditional Heteroscedasticity*) si configurano come generalizzazione degli ARCH(*m*), dove la varianza h_t è funzione lineare dei valori passati degli errori passati al quadrato ε_{t-i}^2:

$$h_t = \omega + \sum_{i=1}^{m} \alpha_i \varepsilon_{t-i}^2 \quad (28)$$

dove ω è una costante, α_i indica il coefficiente associato ad ogni ε_{t-i}^2 e m rappresenta l'ordine della componente a media mobile del modello.

Il processo GARCH(*m,s*) generalizza l'ARCH(*m*) aggiungendo al modello in questione anche i valori passati della variabile dipendente. Dunque, l'equazione (28) viene modificata nella seguente maniera:

$$h_t = \omega + \sum_{i=1}^{m} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=1}^{s} \beta_i h_{t-1} \quad (29)$$

dove β_i indica il coefficiente associato ad ogni ritardo della variabile dipendente e s rappresenta l'ordine della componente autoregressiva del modello.

A differenza del modello ARMA(*p,q*), è stato deciso di non variare la trattazione rispetto al contributo iniziale di Barone-Adesi e Giannopoulos, fissando dunque gli ordini del modello GARCH(*m,s*) entrambi pari a 1.

Inoltre, è opportuno precisare che sul modello GARCH viene applicata l'unica ipotesi distributiva prevista all'interno del metodo FHS. Infatti, la struttura del processo in questione prevede l'ipotesi di normalità degli errori ε_t e di conseguenza le stime vengono inficiate da ciò[42]. Al contempo, bisogna tener conto

[42] Non sempre l'ipotesi di normalità è adatta a descrivere la distribuzione degli errori, soprattutto in presenza di molti *outliers.* Per questo motivo in alcuni casi è preferibile adoperare distribuzioni con valori di curtosi diversi da 3, come la *t* di Student e la *Generalised Error Distribution (GED).*

come l'ipotesi distributiva appena menzionata sia fortemente meno stringente rispetto a quelle presenti nei metodi di simulazione parametrici in senso stretto, i quali si basano sull'approssimazione della distribuzione empirica dei rendimenti storici mediante funzioni di distribuzione teoriche.

Giunge ora il momento della stima del modello GARCH(1,1), effettuata mediante la libreria *arch_model* di Kevin Sheppard[43]. L'output della stima viene mostrato nella *Figura 13*, la quale evidenzia come il metodo di ottimizzazione sia quello della massima verosimiglianza e specifica l'utilizzo degli errori standard robusti.

Figura 13 - Output del modello GARCH(1,1)

```
                     Zero Mean - GARCH Model Results
==============================================================================
Dep. Variable:                    None   R-squared:                       0.000
Mean Model:                 Zero Mean   Adj. R-squared:                  0.001
Vol Model:                      GARCH   Log-Likelihood:                6234.90
Distribution:                  Normal   AIC:                           -12463.8
Method:            Maximum Likelihood   BIC:                           -12447.1
                                        No. Observations:                 1952
Date:                Sun, Jan 22 2023   Df Residuals:                     1952
Time:                        15:17:07   Df Model:                            0
                            Volatility Model
==============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
------------------------------------------------------------------------------
omega      2.5393e-06  2.619e-09    969.621      0.000 [2.534e-06,2.544e-06]
alpha[1]       0.1284  3.732e-02      3.441  5.786e-04   [5.529e-02,  0.202]
beta[1]        0.8589  2.631e-02     32.641 1.083e-233     [  0.807,  0.910]
==============================================================================

Covariance estimator: robust
```

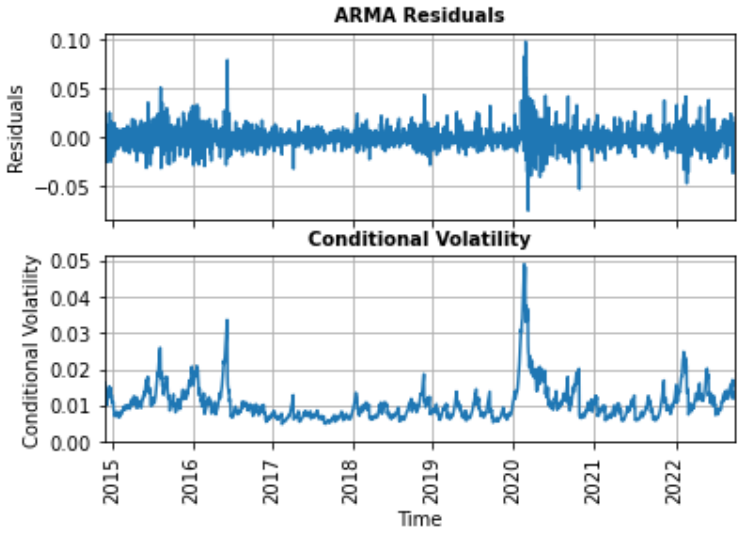
Fonte: Elaborazione propria

Risulta immediato constatare come tutte le variabili siano fortemente significative, soprattutto a causa degli errori standard infinitesimali: il modello, dunque, sembra avere adeguato potere esplicativo, confermando l'inadeguatezza dell'ipotesi di omoschedasticità. Quanto appena affermato sembra essere avvalorato dalla *Figura 14*, la quale mostra in alto il trend storico dei residui del

[43] Kevin Sheppard è professore associato di Econometria Finanziaria presso l'università di Oxford.

modello ARMA, mentre in basso presenta i valori di volatilità condizionale nel tempo.

Figura 14 - Residui del modello ARMA e volatilità condizionale nel tempo



Fonte: Elaborazione propria

La *Figura 14* evidenzia chiaramente la presenza di cluster di volatilità, marcata soprattutto nel 2020 in relazione alla forte incertezza presente nei mercati a causa della pandemia. Anche nel 2022 è possibile notare un incremento del livello medio di volatilità storico: la ragione alla base di ciò sembra poter essere imputabile allo scenario macroeconomico e geopolitico vigente, caratterizzato da forte instabilità.

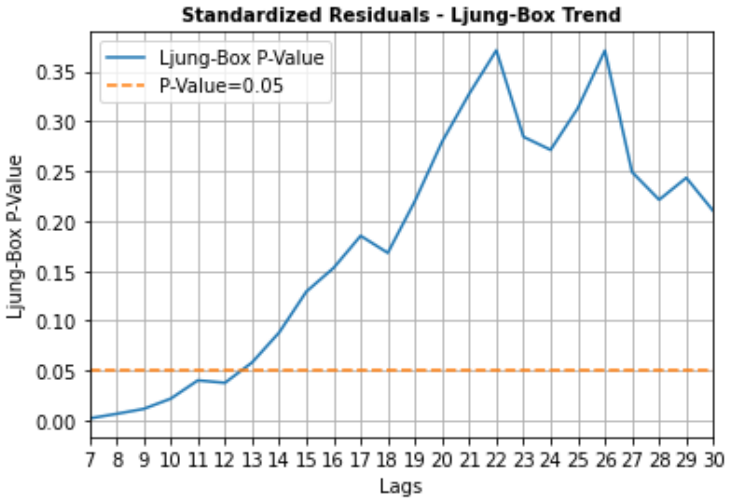
2.3.3 Scaling dei residui

L'ultima attività rimanente in vista del processo di simulazione coincide con la standardizzazione dei residui. Nello specifico, i residui del modello ARMA(3,3) vengono divisi per la volatilità condizionata stimata mediante il processo GARCH(1,1), come illustrato nell'equazione (4). Così facendo, i residui vengono depurati dall'effetto della correlazione seriale e perciò ogni fenomeno rilevante di

persistenza tra gli stessi viene rimosso. Ciò giustifica l'utilizzo dei residui standardizzati e^*_i all'interno del processo di simulazione ed il successivo processo di *re-scaling*, il quale adatta ogni e^*_i al valore di volatilità previsto $\sqrt{\hat{h}_{t+i}}$.

L'abbattimento dell'effetto di correlazione seriale mediante lo *scaling* viene dimostrato dalla *Figura 15,* la quale mostra l'esito dei test di Ljung-Box condotti sulla serie dei residui standardizzati al variare dell'ordine *k.*

Figura 15 - Trend del Test di Ljung-Box: residui standardizzati



Fonte: Elaborazione Propria

Infatti, in seguito al processo di *scaling* i residui sembrano essere correlati fino all'ordine 12, a differenza di quanto osservato precedentemente nella *Figura 12,* dove l'autocorrelazione seriale era visibile fino all'ordine 24.

2.4 SIMULAZIONE

In seguito alla determinazione dei modelli parametrici e all'implementazione del processo di *scaling*, viene ora effettuata la simulazione dei prezzi e dei rendimenti futuri di portafoglio, parte fondamentale del metodo FHS.

2.4.1 Caratteristiche generali

Come illustrato nel Capitolo 1, la simulazione FHS si basa su un ricampionamento *bootstrap* sulla serie dei residui standardizzati, che alimenta in seguito i modelli parametrici ARMA e GARCH al fine di simulare rispettivamente rendimenti e volatilità futuri. Dal punto di vista pratico, il ricampionamento in oggetto è stato effettuato mediante due diverse metodologie:

1. libreria *arch.bootstrap,* la quale attraverso il metodo *IndependentSamplesBootstrap* consente di effettuare un ricampionamento con reimmissione in relazione a serie storiche indipendenti;
2. metodo del posizionale[44], il quale seleziona il residuo standardizzato la cui funzione di ripartizione associata ha distanza minima rispetto al valore estratto da una distribuzione uniforme continua *U(0,1).*

È opportuno precisare come la scelta tra i due metodi alternativi non influenzi significativamente l'esito della simulazione e per tale motivo in sede di illustrazione dei risultati non verrà effettuato alcun distinguo tra essi. Nonostante ciò, è stato preferito mantenere la duplice metodologia di ricampionamento all'interno del codice, al fine di fornire all'utente la possibilità di ottenere una simulazione più versatile.

Allo stesso modo, risulta importante sottolineare come il codice associato alla simulazione *FHS* non sia limitato soltanto ai modelli ARMA(3,3) e GARCH(1,1),

[44] Il codice relativo al metodo del posizionale è stato elaborato completamente *from scratch*, senza l'utilizzo di alcuna libreria di supporto.

selezionati all'interno del presente elaborato, ma piuttosto risulti adattabile a qualsiasi ordine *p, q, m, s* dei modelli summenzionati.

Inoltre, ai fini di una corretta analisi dei risultati e di un'efficace valutazione delle misure VaR ottenute, il numero delle simulazioni *S* è stato fissato ad un valore pari a 10000.

2.4.2 Implementazione e risultati ottenuti

Come già precisato precedentemente, l'applicazione empirica del presente elaborato varia rispetto alla trattazione originale di Barone-Adesi e Giannopoulos poiché non si limita ad usare un modello ARMA(1,1), ma piuttosto vengono selezionati gli ordini *p* e *q* indicati come ottimali dalle procedure inferenziali utilizzate. Detto ciò, risulta necessario evidenziare come l'equazione (7) venga modificata nella seguente maniera:

$$(30) \quad r^*_{t+i} = \mu + \varphi_1 r^*_{t+i-1} + \varphi_2 r^*_{t+i-2} + \varphi_3 r^*_{t+i-3} + \theta_1 z^*_{t+i-1} + \\ + \theta_2 z^*_{t+i-2} + \theta_3 z^*_{t+i-3} + z^*_{t+i} \qquad \text{con}^{45}\ i \geq 4.$$

Il vettore dei parametri relativo all'equazione appena mostrata deriva dalla stima ARMA e viene illustrato nella seguente equazione:

$$(31) \quad \begin{bmatrix} \mu \\ \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} -0.000125 \\ -0.537059 \\ -0.605378 \\ -0.873574 \\ 0.585330 \\ 0.633110 \\ 0.934218 \end{bmatrix}$$

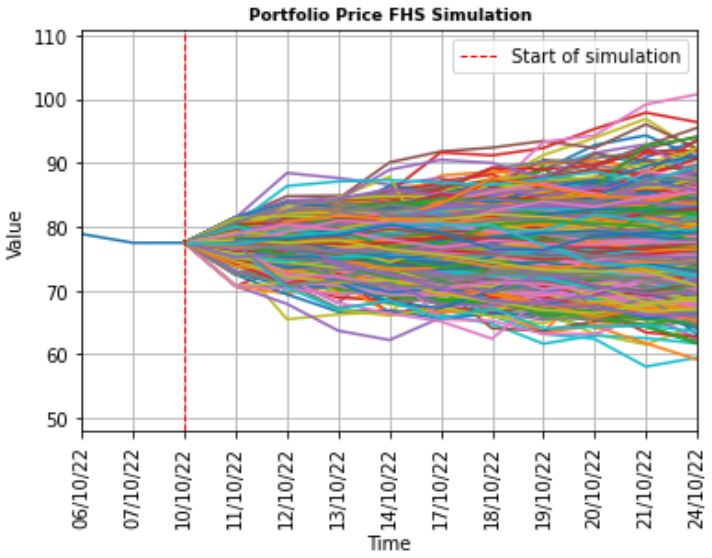
Allo stesso tempo i parametri del modello GARCH(1,1) relativi all'equazione (9) sono:

[45] Quando $1 \leq i \leq 3$, le variabili simulate (connotate dall'asterisco) presenti nell'equazione (30) vengono sostituite dai valori osservati disponibili.

$$(32) \quad \begin{bmatrix} \omega \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 2.539303e^{-6} \\ 0.128448 \\ 0.858919 \end{bmatrix}$$

Tenuto conto di quanto appena menzionato e della trattazione presente nel Capitolo 1, vengono ora mostrati i risultati ottenuti mediante la simulazione FHS. Nello specifico, la Figura 16 illustra le traiettorie di prezzo del portafoglio in un orizzonte temporale di simulazione *H* pari a 10 giorni[46]. È opportuno precisare che l'ultimo valore osservato di portafoglio, pari a €77.50, è relativo al 10 ottobre 2022, come segnalato dalla linea tratteggiata rossa visibile in Figura 16.

Figura 16 - Simulazione FHS: valori di portafoglio



Fonte: Elaborazione Propria

Risulta evidente constatare come le traiettorie simulate di prezzo assumano la forma di una campana rovesciata, data la forte concentrazione dei valori di portafoglio attorno al centro della distribuzione ottenuta. Quanto appena constatato viene confermato dalla Tabella 8, la quale mostra le statistiche descrittive delle distribuzioni associate ai rendimenti e ai prezzi di portafoglio e relative all'ultimo

[46] L'orizzonte di simulazione comprende soltanto le giornate di apertura delle principali borse europee: sono perciò escluse dal conteggio le festività.

giorno di simulazione (24 ottobre 2022). È possibile notare che il 50% delle osservazioni è compreso tra i valori pari a €75.04 e €78.87, segnale di una forte polarizzazione dei prezzi simulati attorno alla media aritmetica, pari a €76.95.

Tabella 8 - Statistiche descrittive relative alla simulazione FHS

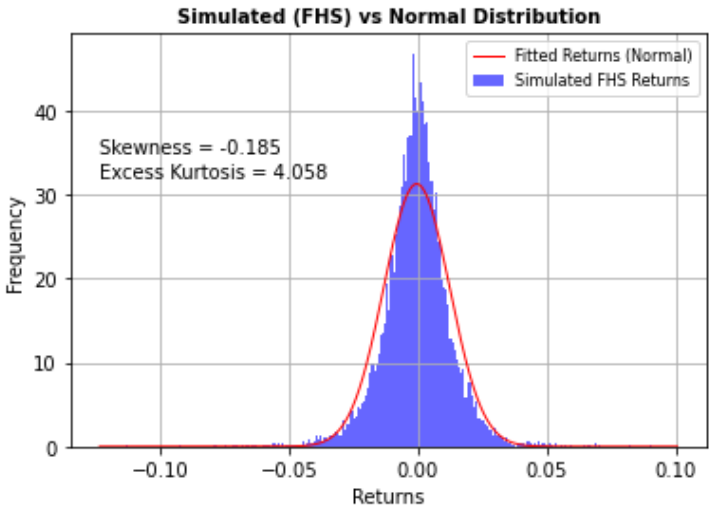
Statistica descrittiva	Prezzo	Rendimento
Media aritmetica	€76.95	-0.04%
Deviazione Standard	3.2654	0.0128
Minimo	€59.21	-11.31%
Percentile: 25%	€75.04	-0.71%
Percentile: 50%	€76.98	-0.02%
Percentile: 75%	€78.87	0.64%
Massimo	€100.75	9.04%

Fonte: Elaborazione propria

Inoltre, è possibile osservare come sia la media aritmetica che la mediana dei rendimenti simulati si attestino entrambe su valori negativi, ma prossimi allo zero. Ciò conferma il fenomeno di concentrazione delle simulazioni rispetto al valore di partenza, ma allo stesso tempo indica una lieve tendenza negativa del portafoglio. Quest'ultima era presente anche nei dati storici di portafoglio, visto il rendimento medio osservato pari a -0.01%.

Nonostante l'addensamento delle osservazioni attorno al valore centrale, dalla Tabella 8 si constata la presenza di rendimenti estremi, dati i valori massimi e minimi simulati rispettivamente pari a 9.04% e -11.31%. Quanto appena affermato viene mostrato nel dettaglio nella Figura 17, la quale confronta la distribuzione dei rendimenti simulati con una distribuzione normale, avente media μ=-0.04% e deviazione standard σ=0.0128.

Figura 17- Distribuzione simulata dei rendimenti e distribuzione normale



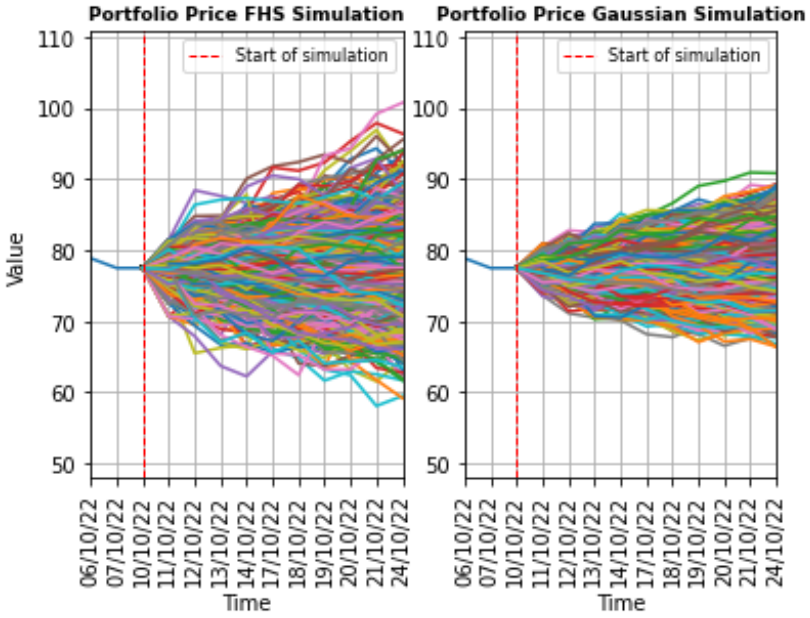
Fonte: Elaborazione propria

La presenza di valori estremi nella distribuzione dei rendimenti simulati viene avallata dall'eccesso di curtosi, pari a 4.058 e nettamente maggiore rispetto a quello della distribuzione normale, per definizione pari a 0. Al contempo, l'indice di asimmetria pari a -0.185 suggerisce una leggera preponderanza dei rendimenti negativi e ciò viene confermato anche dalla mediana pari a -0.02%.

Oltre a mostrare la presenza di rendimenti estremi, la Figura 17 consente di analizzare la bontà di adattamento della distribuzione normale rispetto alla serie dei rendimenti simulati. Dati l'indice di asimmetria e l'eccesso di curtosi, risulta evidente constatare come la distribuzione normale non sia in grado di rappresentare in modo accurato la frequenza dei rendimenti simulati di portafoglio. Ciò conferma la ragione fondamentale alla base dell'ideazione della tecnica FHS: l'applicazione di vincoli distributivi alla serie dei rendimenti storici impatta in modo rilevante sull'accuratezza della simulazione, rendendo i risultati spesso non affidabili, data la mancanza di previsione di valori estremi. Al contrario, la tecnica FHS consente la generazione anche di rendimenti più distanti dalla media, gestendo in modo più efficace il problema delle *fat tails,* a differenza della distribuzione normale.

In sintesi, ipotizzare che i rendimenti di portafoglio siano distribuiti in modo normale può inficiare la capacità previsiva della simulazione e la produzione di valori estremi. Quanto appena affermato è comprovato dalla Figura 18, la quale confronta i risultati ottenuti dalla simulazione FHS e dalla simulazione basata su distribuzione gaussiana[47].

Figura 18 - Confronto tra simulazione FHS e gaussiana



Fonte: Elaborazione propria

La simulazione basata sulla distribuzione gaussiana produce esclusivamente rendimenti polarizzati attorno alla media, mentre al contrario il metodo FHS genera anche valori più estremi. Di conseguenza, la simulazione gaussiana elabora necessariamente una stima VaR meno prudente rispetto a quella basata sul metodo di Barone-Adesi e Giannopoulos, esponendo il portafoglio a perdite superiori rispetto a quanto previsto.

[47] Entrambe le metodologie hanno considerato un numero di simulazioni *S* pari a 10000.

Al fine di fornire una trattazione più completa, viene ora illustrata la Tabella 9, la quale mostra le statistiche descrittive associate ai due diversi metodi di simulazione.

Tabella 9 - Simulazione FHS e gaussiana: confronto tra statistiche descrittive

Statistica descrittiva	Simulazione FHS		Simulazione Gaussiana	
	Prezzo	Rendimento	Prezzo	Rendimento
Media aritmetica	€76.95	-0.04%	€77.31	-0.06%
Deviazione Standard	3.2654	0.0128	3.1107	0.0128
Minimo	€59.21	-11.31%	€66.38	-4.92%
Percentile: 25%	€75.04	-0.71%	€75.17	-0.92%
Percentile: 50%	€76.98	-0.02%	€77.23	-0.05%
Percentile: 75%	€78.87	0.64%	€79.37	0.78%
Massimo	€100.75	9.04%	€90.84	4.91%

Fonte: Elaborazione propria

L'osservazione della Tabella 9 consente di verificare la capacità della simulazione FHS di produrre rendimenti più distanti dalla media, rispetto alla metodologia gaussiana. Al contempo, lo scarto interquantile[48] associato al primo metodo, pari a 1,35%, è inferiore rispetto a quello riferito alla simulazione con distribuzione normale, pari a 1,7%. Dunque, viene osservata una contestuale tendenza del metodo di simulazione FHS a concentrare maggiormente i valori attorno alla mediana e allo stesso tempo di generare valori assai più distanti rispetto al centro della distribuzione. Questo aspetto caratteristico della metodologia FHS sembra essere maggiormente in linea con l'osservazione empirica.

[48] Lo scarto interquantile è la differenza tra terzo e primo quartile e rappresenta l'ampiezza della fascia comprendnte la metà "centrale" dei valori osservati.

CAPITOLO 3

VaR E BACKTESTING

3.1 VALUE AT RISK (VaR)

Dopo aver effettuato la simulazione FHS, è necessario ora focalizzare l'attenzione sulla misurazione del rischio di portafoglio, di modo che possa essere testato il comportamento di quest'ultimo in presenza di scenari sfavorevoli. Al fine di rendere la trattazione più conforme alle pratiche comuni nel mondo del *Risk Management*, è stato deciso di selezionare tra le varie misure di rischio di portafoglio il *Value At Risk (VaR)*[49].

Il *VaR* nasce dall'esigenza del *top management* degli istituti finanziari di fruire di una misura immediata e sintetica del rischio di portafoglio, evitando l'analisi di lunghi report di difficile comprensione. La misura di rischio in questione è divenuta popolare all'interno del mondo del *risk management* nei primi anni '90, quando Dennis Weatherstone, CEO di JPMorgan, decide di adottarla all'interno dei processi di valutazione del rischio interni alla banca. Successivamente anche gli organismi di vigilanza bancaria hanno deciso di utilizzare il *VaR* come misura di rischio, inglobandola esplicitamente nel quadro regolamentare di Basilea relativo alla gestione del capitale bancario. Nello specifico, i requisiti patrimoniali di Basilea prevedono la misurazione del rischio di mercato mediante un *VaR* con orizzonte temporale di 10 giorni e confidenza pari al 99%.

[49] La trattazione è comunque estendibile, seppur con alcune modifiche, ad ulteriori misure di rischio, come l'*Expected Shortfall (ES)*.

Il $VaR_T^{1-\alpha}$ è definito come la massima perdita potenziale del portafoglio, espressa in forma di rendimento, in un orizzonte temporale *T*, dato il livello di confidenza *1-α*[50]. In formula:

(33) $P(r_t < -VaR_T^{1-\alpha}) = \alpha$

dove r_t rappresenta il rendimento di portafoglio al tempo *t* e *0≤α≤1*.

Nonostante si tratti di una misura relativamente semplice e immediata, esistono diverse modalità di determinazione del *VaR*, che tendono a distinguersi per la presenza o meno di ipotesi sulla distribuzione dei rendimenti. Nel caso in cui la distribuzione dei rendimenti venga approssimata mediante la distribuzione normale, il *VaR* può essere calcolato nella seguente maniera:

(34) $VaR_T^{1-\alpha} = \hat{\mu}_T - \phi_\alpha^{-1}\hat{\sigma}_T$

dove $\hat{\mu}_t$ indica il rendimento medio stimato del portafoglio, $\widehat{\sigma_t}$ la deviazione standard stimata e ϕ_α^{-1} il quantile della distribuzione normale standard con livello di probabilità α.

Nel caso in cui non venga effettuata alcuna ipotesi distributiva rispetto ai rendimenti di portafoglio, il *VaR* è calcolato in maniera differente:

(35) $VaR_T^{1-\alpha} = \Omega_\alpha^{-1}$

dove Ω_α^{-1} rappresenta il quantile con livello di probabilità α associato alla distribuzione simulata dei rendimenti.

In sintesi, esso è dunque calcolabile a partire dalla distribuzione dei rendimenti del portafoglio e coincide perciò con il quantile associato al prefissato livello di confidenza *1-α,* solitamente compreso tra 0.90 e 0.99.

Nonostante sia diventato il *benchmark* per il calcolo del rischio di mercato, il *VaR* è caratterizzato da limiti e punti di debolezza. Infatti, l'approccio di misurazione in questione non considera in alcun modo gli eventi estremi, non fornendo alcuna informazione riguardo gli scenari riferiti allivello di probabilità *α.*

[50] È opportuno puntualizzare che la misura di rischio $VaR_T^{1-\alpha}$ sia calcolata in valore assoluto.

Inoltre, il *VaR* presuppone l'invarianza della composizione di portafoglio durante l'orizzonte di analisi *T*, la quale costituisce un'ipotesi irrealistica soprattutto quando il periodo di previsione è elevato. Per di più, la scelta dei parametri *T* e α è soggetta a diverse interpretazioni all'interno del mondo finanziario, rendendo in alcuni casi difficoltosa la comparabilità tra diversi portafogli.

3.2 BACKTESTING

Il *backtesting* è definito come metodologia di analisi e valutazione *ex-post* dell'efficacia del modello utilizzato e dell'accuratezza delle stime *out-of-sample*[51] prodotte. Esso si configura come ultima fase prevista all'interno della procedura di *Risk Management* e si basa sul confronto tra misure di rischio stimate e rendimenti osservati all'interno dell'orizzonte temporale *T*.

Il *backtesting* consente dunque di monitorare la performance delle misure di rischio prodotte, valutandone l'effettiva efficacia di fronte agli scenari di mercato effettivamente osservati. Affinché però la valutazione sull'affidabilità del modello sia corretta, è necessario ripetere la procedura di *backtesting* per diversi periodi storici, evitando perciò che le i risultati dipendano esclusivamente dallospecifico lasso temporale prescelto.

3.2.1 Backtesting sul VaR

Come già evidenziato, la misura $VaR_{t+1}^{1-\alpha}$ incorpora l'assunzione che il rendimento osservato al tempo *t+1* sarà minore rispetto alla stima $VaR_{t+1}^{1-\alpha}$ α*100% volte. Detto ciò, è possibile definire la variabile indicatrice I_{t+1}, la quale assume valore 1 in caso di violazione osservata al tempo *t+1* del $VaR_{t+1}^{1-\alpha}$ e 0 in presenza di evento opposto.

[51] Per stima *out-of-sample* si intende una specifica stima effettuata in relazione ad osservazioni non presenti nel campione.

(36) $$I_{t+1} = \begin{cases} 1 & se\ r_{t+1} < -VaR_{t+1}^{1-\alpha} \\ 0 & se\ r_{t+1} > -VaR_{t+1}^{1-\alpha} \end{cases}$$

Quando viene effettuata un'analisi di *backtesting* in relazione ad un modello di rischio, viene costruita la sequenza $\{I_{t+1}\}_{t=1}^{T}$, la quale misura il numero di violazioni osservate nell'orizzonte di riferimento *T.*

La sequenza $\{I_{t+1}\}_{t=1}^{T}$ costituisce un processo di *Bernoulli,* definito come processo di variabili casuali bernoulliane indipendenti e identicamente distribuite con probabilità *α.* Infatti, la presenza di violazioni nell'orizzonte *T* è necessariamente imprevedibile: in caso contrario, l'analista disporrebbe di informazioni tali per cui il modello sottoposto a procedura di *backtesting* sia migliorabile. In sintesi:

(37) $$I_{t+1} \sim i.i.d.Bernoulli\ (\alpha),$$

avente la funzione di probabilità descritta dall'equazione (38):

(38) $$f(I_{t+1};\ \alpha) = (1-\alpha)^{1-I_{t+1}}\alpha^{I_{t+1}}.$$

Sulla base di quanto affermato finora, è possibile definire il test *Proportion Of Failure (POF),* proposto da Kupiec nel 1995. Il test in questione mira a stabilire se il valore π, rappresentante il tasso di violazione riscontrato*,* sia significativamente differente rispetto al livello di probabilità α del *VaR*:

(39) $$\begin{cases} H_0: & \pi = \alpha \\ H_1: & \pi \neq \alpha \end{cases}$$

Il valore π può essere stimato mediante il *failure rate* $\hat{\pi}$, calcolato come rapporto tra numero di violazioni effettivamente rilevate T_1 e orizzonte *T.*

Dal punto di vista statistico, il *POF* si configura come un *likelihood ratio*[52] test (*LR)* e segue la seguente formulazione:

[52] Un *Likelihood Ratio* test (*LR*) è un test statistico che valuta la bontà di adattamento tra due modelli alternativi, ottenuti rispettivamente a seguito di un'ottimizzazione libera e vincolata. Il test *LR* si configura come rapporto tra i valori di verosimiglianza dei due modelli.

$$(40) \quad LR_{POF} = -2\ln\left[\frac{L(\alpha)}{L(\hat{\pi})}\right] = -2\ln\left[\frac{(1-\alpha)^{1-T_1}\alpha^{T_1}}{\left(1-{}^{T_1}/_{T}\right)^{1-T_1}\left({}^{T_1}/_{T}\right)^{T_1}}\right] \sim \chi_1^2$$

dove $L(\alpha)$ e $L(\hat{\pi})$ indicano i valori delle funzioni di verosimiglianza calcolati rispettivamente in α e $\hat{\pi}$, mentre χ_1^2 rappresenta una distribuzione *chi-quadro* con un grado di libertà.

È opportuno precisare come la determinazione del livello di significatività del test dipenda dalla valutazione dei costi associati a due tipologie di errori: errore di primo tipo (rifiutare un modello corretto) e di secondo tipo (non rifiutare un modello sbagliato)[53]. Quest'ultimo caso può costituire un maggiore costo per un istituto finanziario: per questo motivo nell'ambito del *risk management* la prassi comune prevede l'utilizzo di un livello di significatività pari al 10%[54].

Oltre a valutare dal punto di vista statistico il *failure rate* del modello utilizzato, l'analista deve necessariamente indagare sulla collocazione temporale delle violazioni osservate. Infatti, l'eventuale presenza di fenomeni di *clustering* nell'orizzonte *T* di osservazione costituirebbe un rischio nettamente maggiore per l'istituto finanziario rispetto al caso in cui le violazioni siano "randomicamente" distribuite nel tempo. Inoltre, nel caso di *clustering*, il *risk manager,* di fronte all'osservazione di una violazione al tempo *t*, può essenzialmente anticipare nel breve termine la presenza di rendimenti inferiori al livello $VaR_T^{1-\alpha}$ prefissato. Tale capacità previsiva consente dunque al *risk manager* di poter innalzare il valore del $VaR_T^{1-\alpha}$, mitigando il rischio gravante sull'istituto finanziario.

Per questi motivi viene ora illustrato il test d'indipendenza, elaborato da Christoffersen nel 1998 e conosciuto anche come *test di Markov*. Si assuma che la

[53] Esiste un *tradeoff* tra livello di significatività ed errori di primo tipo e secondo tipo. Infatti, un livello di significatività più alto comporta rispettivamente un maggiore errore di prima specie e un minore errore di seconda specie.

[54] Allo scopo di fornire una trattazione completa, i test condotti nel presente capitolo saranno condotti con tre livelli di significatività: 1%, 5% e 10%.

sequenza $\{I_{t+1}\}_{t=1}^{T}$sia dipendente nel tempo e che possa essere descritta dalla sequenza di Markov di primo ordine, caratterizzata dalla seguente matrice di transizione:

$$(41) \quad \Pi_1 = \begin{bmatrix} 1-\pi_{01} & \pi_{01} \\ 1-\pi_{11} & \pi_{11} \end{bmatrix}$$

dove π_{01}e π_{11}indicano le probabilità di osservare una violazione al tempo *t+1* data la presenza o meno di una violazione al tempo *t*[55]. In formule:

$$(42) \quad \pi_{01} = \Pr(I_{t+1} = 1 \mid I_t = 0)$$

$$(43) \quad \pi_{11} = \Pr(I_{t+1} = 1 \mid I_t = 1).$$

Dunque, il processo di Markov di primo ordine si basa sull'assunzione che il risultato di domani dipenda esclusivamente da quanto osservato oggi: le violazioni al tempo *t+1* sono dipendenti solamente da quelle al tempo *t*. Nel caso di un campione di ampiezza *T,* la funzione di verosimiglianza del processo di Markov di primo ordine assume la seguente formulazione:

$$(44) \quad L(\Pi_1) = (1-\pi_{01})^{T_{00}} {\pi_{01}}^{T_{01}} (1-\pi_{11})^{T_{10}} {\pi_{11}}^{T_{11}}$$

dove T_{ij}indica il numero di osservazioni tali per cui il valore *j=0,1* sia preceduto dal valore *i=0,1*[56].

È possibile dimostrare che la funzione di verosimiglianza $L(\Pi_1)$ risulta essere massimizzata quando:

$$(45) \quad \widehat{\pi_{01}} = \frac{T_{01}}{T_{00}+T_{01}} \text{ e } \widehat{\pi_{11}} = \frac{T_{11}}{T_{10}+T_{11}}.$$

Considerando che necessariamente $\widehat{\pi_{00}} = 1 - \widehat{\pi_{01}}$ e $\widehat{\pi_{10}} = 1 - \widehat{\pi_{11}}$[57], la matrice di transizione stimata presenta la seguente connotazione:

---

[55] Si noti come le due probabilità π_{01}e π_{11}descrivano interamente il processo di Markov di primo ordine, vista la presenza di due soli possibili risultati (violazione/non violazione).

[56] Ad esempio, T_{11} corrispone al numero di osservazioni caratterizzate dalla presenza di due violazioni consecutive.

[57] Considerando che $\widehat{\pi_{00}}$ e $\widehat{\pi_{10}}$ sono probabilità, deve valere uno dei tre assiomi della probabilità: la probabilità dell'interno spazio campionario Ω è pari a 1 ($\Pr(\Omega) = 1$).

$$(46) \quad \widehat{\Pi_1} = \begin{bmatrix} \widehat{\pi_{00}} & \widehat{\pi_{01}} \\ \widehat{\pi_{10}} & \widehat{\pi_{11}} \end{bmatrix} = \begin{bmatrix} 1-\widehat{\pi_{01}} & \widehat{\pi_{01}} \\ 1-\widehat{\pi_{11}} & \widehat{\pi_{11}} \end{bmatrix} = \begin{bmatrix} \frac{T_{00}}{T_{00}+T_{01}} & \frac{T_{01}}{T_{00}+T_{01}} \\ \frac{T_{10}}{T_{10}+T_{11}} & \frac{T_{11}}{T_{10}+T_{11}} \end{bmatrix}$$

La presenza di dipendenza di primo ordine all'interno della sequenza $\{I_{t+1}\}_{t=1}^{T}$ può essere valutata testando statisticamente le probabilità π_{01}e π_{11}. Infatti, in caso di indipendenza la probabilità di osservare una violazione domani non dipende in alcun modo dal risultato osservato oggi e per questo motivo deve valere che $\pi_{01} = \pi_{11} = \pi$. Sotto l'ipotesi d'indipendenza, la matrice di transizione è perciò uguale a:

$$(47) \quad \widehat{\Pi} = \begin{bmatrix} 1-\hat{\pi} & \hat{\pi} \\ 1-\hat{\pi} & \hat{\pi} \end{bmatrix},$$

mentre il test di indipendenza di Christoffersen assume la seguente struttura:

$$(48) \quad \begin{cases} H_0: & \pi_{01} = \pi_{11} \rightarrow \textit{Le violazioni sono indipendenti nel tempo} \\ H_1: & \pi_{01} \neq \pi_{11} \rightarrow \textit{Le violazioni sono dipendenti nel tempo} \end{cases}$$

Anche in questo caso, l'ipotesi nulla H_0 può essere testata mediante un *likelihood ratio test:*

$$(49) \quad LR_{IND} = -2\ln\left[\frac{L(\hat{\pi})}{L(\widehat{\Pi_1})}\right] \sim \chi_1^2,$$

dove $L(\hat{\pi})$ rappresenta la funzione di verosimiglianza dell'ipotesi alternativa del test *POF.*

Infine, risulta importante poter testare simultaneamente l'indipendenza delle violazioni osservate e la correttezza del *failure rate.* A tal fine è possibile combinare i test di Kupiec e Christoffersen, ottenendo il *Conditional Coverage (CC) Test,* il quale considera l'ipotesi nulla del *test POF* e l'ipotesi alternativa del test d'indipendenza:

$$(50) \quad \begin{cases} H_0: & \pi_{01} = \pi_{11} = \alpha \\ H_1: & \pi_{01} \neq \pi_{11} \neq \alpha \end{cases}$$

Anche il *CC Test* si basa sul *likelihood ratio test* ed essendo una combinazione di due metodi in questione può essere calcolato sommando le statistiche test associate ai singoli test individuali[58]. Nello specifico[59]:

$$(51) \quad LR_{CC} = -2\ln\left[\frac{L(\alpha)}{L(\widehat{\Pi_1})}\right] = LR_{POF} + LR_{IND} \sim \chi_2^2$$

Concludendo, una misura $VaR_T^{1-\alpha}$ ideale dovrebbe garantire simultaneamente la proprietà d'indipendenza tra le violazioni osservate e un adeguato *failure rate.* A tal fine il *CC Test* risulta essere uno strumento adatto per effettuare la presente analisi, data la sua facilità di calcolo e il suo elevato potere esplicativo.

3.2.2 Backtesting: analisi empirica

Dopo aver illustrato il processo di *backtesting* dal punto di vista teorico, viene ora analizzata la capacità previsiva del metodo di simulazione FHS. Nello specifico, il *VaR* giornaliero ottenuto dal presente metodo durante l'orizzonte di *backtesting* viene confrontato con i rendimenti di portafoglio realmente osservati, focalizzando l'attenzione sul numero di violazioni riscontrate. La stessa tipologia di analisi viene replicata per il *VaR* storico gaussiano e per il *VaR* ottenuto dalla simulazione parametrica attraverso ARMA-GARCH e successivamente i poteri previsivi dei diversi metodi vengono comparati ed analizzati mediante i test statistici illustrati nel precedente paragrafo.

Il *VaR* storico gaussiano viene calcolato applicando l'equazione (34), utilizzando quindi la seguente formulazione:

---

[58] Si noti come il *CC Test* converga ad una distribuzione χ_2^2 : ciò si verifica a causa della proprietà additiva della distribuzione chi-quadrato.

[59] È possibile ottenere l'equazione (51) mediante la seguente rielaborazione algebrica:

$$LR_{CC} = -2\ln\left[\frac{L(\alpha)}{L(\widehat{\Pi_1})}\right] = -2\ln\left[\{L(\alpha)/L(\widehat{\Pi})\}\{L(\widehat{\Pi})/L(\widehat{\Pi_1})\}\right] =$$
$$= -2\ln\left[L(\alpha)/L(\widehat{\Pi})\right] - 2\ln\left[L(\widehat{\Pi})/L(\widehat{\Pi_1})\right] = LR_{POF} + LR_{IND}$$

(52) $$VaR_{t+i}^{0.99} = \bar{\mu} - \phi_{0.01}^{-1}\bar{\sigma}$$

dove $VaR_{t+i}^{0.99}$ indica il *VaR* con confidenza pari al 99% al tempo *t+i*, $\bar{\mu}$ il rendimento medio calcolato sulle ultime 252 osservazioni, $\phi_{0.01}^{-1}$ il quantile con confidenza pari a 0.99 della distribuzione normale standardizzata ($\phi_{0.01}^{-1} = 2.326$) e $\bar{\sigma}$ la deviazione standard media calcolata sulle ultime 252 osservazioni[60].

La simulazione parametrica attraverso ARMA-GARCH si differenzia dal metodo FHS, poiché presuppone la distribuzione gaussiana dei rendimenti. I valori di rendimento medio e volatilità al tempo *t+i* vengono determinati mediante i modelli ARMA-GARCH e da essi vengono prodotte le simulazioni giornaliere di rendimento di portafoglio. Nello specifico:

(1) $$r_{t+i}^{*} = \mu_{t+i}^{*} + \phi^{-1}\sigma_{t+i}^{*}$$

dove r_{t+i}^{*} indica il rendimento simulato di portafoglio al tempo *t+i,* μ_{t+i}^{*} il rendimento medio previsto dal modello ARMA al tempo *t+i*, ϕ^{-1} un quantile associato alla distribuzione normale standardizzata e σ_{t+i}^{*} la volatilità prevista mediante il modello GARCH al tempo *t+i*.

È opportuno precisare come l'orizzonte temporale *T* del processo di *backtesting* sia stato fissato ad un valore pari a 504, coincidente con il numero di giorni di apertura delle principali borse europee in due anni. Infatti, tale valore è stato ritenuto congruo al fine di evitare che l'analisi di *backtesting* sia eccessivamente focalizzata su un particolare periodo storico, rendendo le valutazioni ottenute non applicabili in altri lassi temporali. Allo stesso tempo, bisogna considerare il *tradeoff* tra periodo di backtesting e ampiezza dei dati utilizzati dai metodi FHS e di simulazione ARMA-GARCH. Infatti, una maggiore copertura temporale dell'analisi di *backtesting* riduce allo stesso tempo le osservazioni alimentanti i modelli ARMA- GARCH e il numero di residui

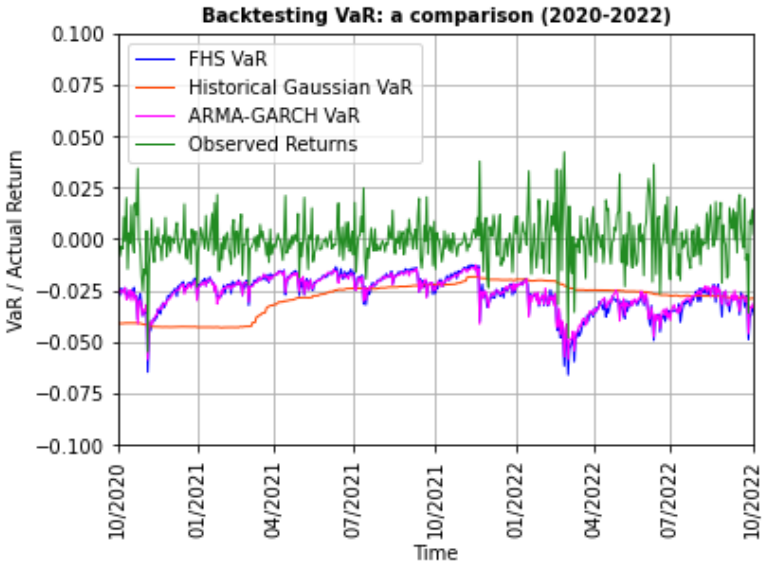
[60] Si noti come siano state selezionate le ultime 252 osservazioni al fine di rappresentare un interno anno solare di analisi.

standardizzati su cui avviene il ricampionamento *bootstrap* all'interno del metodo FHS, con effetti negativi sul potere previsivo di entrambi i metodi.

Inoltre, l'analisi di *backtesting* viene effettuata in relazione a due periodi storici differenti: nel primo caso viene considerato il biennio 2020-2022, mentre nel secondo il biennio 2018-2020. Tale scelta ha lo scopo di rendere la trattazione più completa poiché mediante questa comparazione l'analista può valutare in modo oggettivo, tenendo conto del *tradeoff* sopracitato, l'adattabilità dei vari modelli ai diversi scenari macroeconomici, politici e sociali osservati nei due differenti bienni.

Il confronto tra rendimenti osservati e *VaR* prodotti dai tre metodi nel biennio 2020-2022 viene evidenziato dalla Figura 19:

Figura 19 - Rendimenti e Var a confronto: 2020-2022



Fonte: Elaborazione propria

È possibile notare come il *VaR* storico gaussiano sia caratterizzato da minore variabilità rispetto alle misure di rischio delle metodologie alternative. Questo è dovuto all'assenza di stime parametriche del rendimento medio e della volatilità di portafoglio, cosa che lo rende meno rapido ad adattarsi agli shock che avvengono nei mercati finanziari. Inoltre, risulta evidente come il *VaR* storico gaussiano

dipenda fortemente dai valori osservati di portafoglio negli ultimi 252 giorni, ragion per cui esso risulta fortemente conservativo nel periodo ottobre 2020-luglio 2021. Infatti, ciò dipende esclusivamente dalla vicinanza temporale della pandemia, la quale ha creato instabilità e di conseguenza maggiore volatilità nei mercati. La forte dipendenza dai rendimenti di breve termine rende il *VaR* storico gaussiano inadatto a prevedere adeguatamente gli scenari futuri e per tale motivo è più frequente osservare violazioni a seguito di periodi caratterizzati da bassa volatilità.

Essendo basati su stime semi parametriche e parametriche, i metodi FHS e ARMA-GARCH gaussiano sono caratterizzati da maggiore reattività alle variazioni repentine del contesto economico-finanziario e allo stesso tempo non sono eccessivamente conservativi nei periodi di bassa volatilità. Bisogna considerare infatti che un *VaR* troppo conservativo implica un più elevato assorbimento patrimoniale a carico dell'istituto finanziario e la conseguente impossibilità di utilizzo alternativo di tali risorse finanziarie: in sintesi, un *VaR* eccessivamente gravoso implica l'assunzione di costi opportunità.

La Tabella 10 consente di analizzare la *performance* dei diversi metodi, focalizzando l'attenzione sul numero di violazioni riscontrate e sui corrispondenti test statistici associati. Nonostante le considerazioni precedentemente fatte riguardo le diverse metodologie, è possibile notare come i tre metodi producano risultati analoghi. Infatti, tutti i test statistici sembrerebbero confermare la robustezza dei modelli sia dal lato della correttezza del numero di violazioni che dal punto di vista dell'indipendenza temporale tra le stesse. Allo stesso tempo però è necessario tener conto che questi risultati sono contestualizzati alla finestra temporale di riferimento (biennio 2020-2022), la quale è stata preceduta dalla pandemia e per tale motivo i *VaR* storici gaussiani sono tendenzialmente più conservativi e producono minori violazioni.

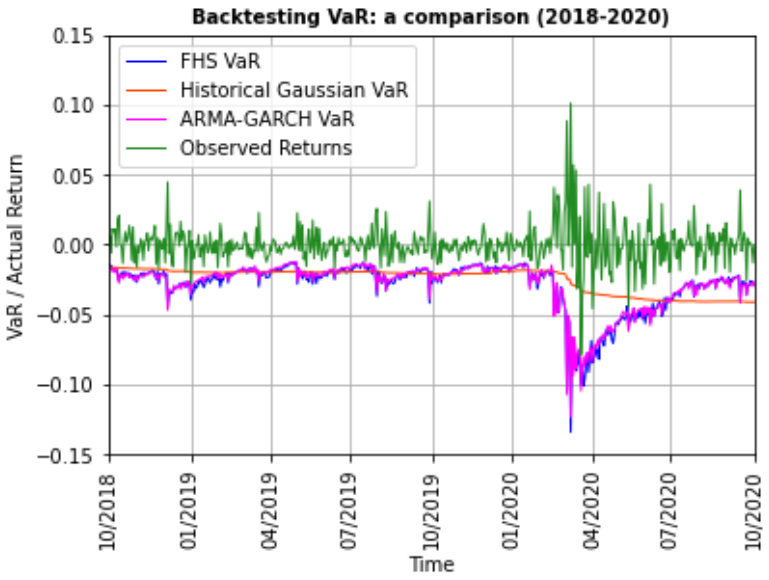
Tabella 10 - Analisi di Backtesting: un confronto tra i diversi metodi (2020-2022)

BIENNIO 2020-2022		**FHS**	**Gaussiana storica**	**ARMA-GARCH Gaussiano**
Numero di violazioni		5	6	5
***POF* Test**	Statistica test	0.0003	0.1741	0.0003
	p-value	0.9857	0.6765	0.9857
	Risultato test: α=0.01	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.05	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.10	Non rifiuto	Non rifiuto	Non rifiuto
***Indipendence* Test**	Statistica test	0	0	0
	p-value	1.00	1.00	1.00
	Risultato test: α=0.01	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.05	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.10	Non rifiuto	Non rifiuto	Non rifiuto
CC Test	Statistica test	0	0	0
	p-value	1.00	1.00	1.00
	Risultato test: α=0.01	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.05	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.10	Non rifiuto	Non rifiuto	Non rifiuto

Fonte: Elaborazione propria

L’analisi appena effettuata viene ora replicata per il biennio 2018-2020, al fine di valutare l’efficacia dei modelli testati per una diversa finestra temporale. La Figura 20 mostra il confronto tra rendimenti storici di portafoglio e *VaR* prodotti dai tre modelli alternativi.

Figura 20 - Rendimenti e Var a confronto: 2020-2022



Fonte: Elaborazione propria

La finestra temporale di riferimento della presente analisi è caratterizzata da un periodo di bassa volatilità del portafoglio seguito da forti perturbazioni all’interno dei mercati, dovute allo scoppio e al protrarsi della pandemia. Considerato ciò, risulta evidente constatare la completa inadeguatezza del *VaR* storico gaussiano di fronte a rapidi mutamenti del contesto macroeconomico-finanziario, come visibile nel periodo 02/2020-05/2020. Inoltre, la forte lentezza di “apprendimento” del modello in questione produce nell’ultimo trimestre analizzato misure di rischio eccessivamente cautelative, aumentando i costi opportunità sopportati dall’istituto finanziario.

Inoltre, i metodi FHS e ARMA-GARCH gaussiano sembrano adattarsi in maniera eccellente rispetto alla forte variabilità dovuta alla pandemia, seppur vi siano delle sporadiche violazioni a marzo 2020.

Le *performance* dei modelli vengono illustrate nel dettaglio nella Tabella 11, la quale evidenzia la forte divergenza in termini di risultati tra il modello storico puro e i modelli basati su componenti parametriche.

Tabella 11 - Analisi di Backtesting: un confronto tra i diversi metodi (2020-2022)

BIENNIO 2018-2020		**FHS**	**Gaussiana storica**	**ARMA-GARCH Gaussiano**
Numero di violazioni		2	7	1
***POF* Test**	Statistica test	2.4014	0.6868	4.8778
	p-value	0.1212	0.4073	0.0272
	Risultato test: α=0.01	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.05	Non rifiuto	Non rifiuto	Rifiuto
	Risultato test: α=0.10	Non rifiuto	Non rifiuto	Rifiuto
***Indipendence* Test**	Statistica test	0	3.1285	0
	p-value	1.00	0.0769	1.00
	Risultato test: α=0.01	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.05	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.10	Non rifiuto	Rifiuto	Non rifiuto
CC Test	Statistica test	0	3.8153	0
	p-value	1.00	0.1484	1.00

	Risultato test: α=0.01	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.05	Non rifiuto	Non rifiuto	Non rifiuto
	Risultato test: α=0.10	Non rifiuto	Non rifiuto	Non rifiuto

Fonte: Elaborazione propria

La Tabella 11 mostra come nel biennio 2018-2020 il metodo FHS produca più violazioni rispetto al modello ARMA-GARCH gaussiano, lasciando intuire una preferenza per quest'ultimo in termini di *performance*. In realtà, il confronto tra le due metodologie deve tenere conto delle rispettive caratteristiche strutturali dei modelli.

Infatti, come evidenziato nei precedenti capitoli, la simulazione FHS si basa su un ricampionamento *bootstrap* sull'insieme dei residui storici standardizzati, utilizzato poi per il processo di *re-scaling* e per le conseguenti stime parametriche. Dunque, l'ampiezza e la rappresentatività del campione sono cruciali ai fini di una corretta simulazione: il modello FHS deve avere perciò sufficiente "profondità storica" per poter dapprima simulare in modo più accurato il rendimento e la volatilità di portafoglio e successivamente produrre misure di rischio solide e adattabili agli *shock* di mercato.

Il modello ARMA-GARCH invece non sia basa su un ricampionamento dei residui, ma soltanto su una stima parametrica. L'ampiezza del dataset è comunque importante per la bontà delle stime dei modelli ARMA e GARCH, ma non risulta essere centrale come nel metodo FHS. Per tale motivo il modello ARMA-GARCH, ma in generale i modelli parametrici in senso stretto, risultano essere preferibili in presenza di limitati dati storici,[61] data la loro rapida adattabilità a fenomeni di forte

[61] I dati storici devono essere comunque sufficienti per garantire una convergenza degli stimatori.

variabilità nel breve termine. Nel caso invece di serie storiche ampie, e in particolare rappresentative di periodi finanziari molto perturbati, il metodo FHS garantisce maggiore affidabilità, poiché l'utilizzo della memoria passata del processo stocastico analizzato consente una maggiore dinamicità e versatilità delle misure di rischio prodotte.

CONCLUSIONE

L'elaborato ha avuto lo scopo di analizzare in modo dettagliato la metodologia di simulazione FHS, ideata da Barone-Adesi e Giannopoulos con l'obiettivo di ovviare alle problematiche insite nei metodi di simulazione strettamente parametrici. Questi ultimi presuppongono l'approssimazione della distribuzione dei rendimenti di portafoglio mediante funzione di probabilità teorica e questo ha impatti significativi sull'affidabilità delle stime e della simulazione.

L'intuizione di Barone-Adesi e Giannopoulos consiste nell'aggirare la problematica della corretta stima della matrice varianza-covarianza, non privando allo stesso tempo il modello della componente parametrica necessaria. Ciò può avvenire utilizzando i rendimenti di portafoglio storici all'interno del processo di simulazione, dopo aver adattato i residui precedentemente standardizzati alle condizioni di volatilità presenti in ogni periodo temporale. Dunque, il ricampionamento *bootstrap* e i successivi processi di *scaling* e *re-scaling* consentono ai modelli ARMA-GARCH di essere alimentati da osservazioni realmente prodotte dal processo stocastico analizzato e allo stesso tempo adattate alle condizioni di mercato attuali, a differenza delle simulazioni storiche pure. Il meccanismo appena illustrato permette la produzione di rendimenti simulati non soggetti ad alcun vincolo distributivo (ad eccezione di quello insito nel modello GARCH) e perciò non sottoposti al fenomeno delle *fat tails.* La simulazione è in grado di creare maggiori rendimenti estremi, come evidenziato nella parte finale del Capitolo 2, consentendo successivamente una misura più accurata del rischio di portafoglio mediante *VaR.*

Infine, la trattazione presente nel Capitolo 3 ha permesso di valutare in modo dettagliato la resilienza del modello FHS rispetto ai diversi scenari macroeconomici

e politici osservati nel tempo, comparando la performance della tecnica di Barone-Adesi e Giannopoulos con quella di due metodi alternativi: gaussiano storico e gaussiano parametrico mediante ARMA-GARCH. L'analisi effettuata permette di evidenziare come la tecnica FHS sia caratterizzata da una forte adattabilità alle variazioni repentine di mercato, proprio come il modello gaussiano parametrico, mentre il *VaR* gaussiano storico appare troppo lento nel recepire mutamenti improvvisi degli scenari macroeconomici. Inoltre, il modello strettamente parametrico tende ad essere più efficace in presenza di ridotta disponibilità di osservazioni.

Concludendo, il metodo FHS supera il limite dei modelli parametrici derivante dall'imposizione di vincoli distributivi ai rendimenti di portafoglio e allo stesso tempo utilizza la memoria del processo stocastico, dopo averla riadattata alle condizioni di mercato odierne, per poter simulare i prezzi e i rendimenti futuri. Ciò consente la considerazione di scenari distanti dal centro della distribuzione, spesso difficili da riprodurre mediante distribuzioni teoriche convenzionali. Allo stesso tempo, il modello in questione è caratterizzato da una forte capacità di adattamento a forti eventi di volatilità di mercato. Tuttavia, al fine di produrre stime di VaR affidabili, è necessario che le osservazioni disponibili siano sufficientemente ampie e, in particolare, tali da includere periodi caratterizzati da differenti regimi di volatilità e crisi finanziarie, in modo da incrementare l'apporto informativo del campione utilizzato.

APPENDICE

I risultati illustrati nel presente elaborato sono stati ottenuti mediante l'utilizzo del linguaggio di programmazione *Python 3.9.* Di seguito viene mostrato lo *script* sviluppato:

```
'''CODICE TESI'''
import pandas as pd

#Definizione funzione (massima visibilità dataframe)
def print_full(x):
    pd.set_option('display.max_columns', len(x))
    print(x)
    pd.reset_option('display.max_columns')

#Definizione funzione test ADF
def adf_test(timeseries):
    from statsmodels.tsa.stattools import adfuller
    #Dictionary con vari nuclei deterministici
    nucleo_deter= {"c" : "Constant","ct" : "Constant and Linear Trend",
                   "ctt" : "Constant and Quadratic Trend","nc" : "No Trend"}
    print("Results of Augmented Dickey-Fuller Test:")
    #creazione dataframe popolato in seguito dal ciclo for
    adfoutput = pd.DataFrame(index=[
        "Test Statistic",
        "p-value",
        "#Lags Used",
        "Number of Observations Used",
        "Critical Value (1%)",
        "Critical Value (5%)",
        "Critical Value (10%)",
        "Result"
        ])
    #Test ADF con vari nuclei deterministici
    for i in nucleo_deter:
        adftest = adfuller(timeseries, regression=i, autolag="BIC")
        output=list(adftest[0:4])
        for key, value in adftest[4].items():
            output.append(value)
        if adftest[1]<=0.05:
            output.append("No unit root")
        else:
            output.append("Unit root")
        adfoutput[nucleo_deter[i]] = output
    print_full(adfoutput)
```

```python
#Definizione funzione test KPSS
def kpss_test(timeseries):
    #rimozione warnigs
    import warnings
    warnings.filterwarnings("ignore")
    from statsmodels.tsa.stattools import kpss
    print("Results of KPSS Test:")
    #dictionary con vari nuclei deterministici
    nucleo_deter= {"c" : "Constant","ct" : "Constant and Linear Trend"}
    #creazione dataframe popolato in seguito dal ciclo for
    kpss_output = pd.DataFrame(index=[
        "Test Statistic",
        "p-value",
        "#Lags Used",
        "Critical Value (10%)",
        "Critical Value (5%)",
        "Critical Value (2,5%)",
        "Critical Value (1%)",
        "Result"
        ])
    #Test KPSS con vari nuclei deterministici
    for i in nucleo_deter:
        kpsstest = kpss(timeseries, regression=i, nlags="auto")
        output=list(kpsstest[0:3])
        for key, value in kpsstest[3].items():
            output.append(value)
        if kpsstest[1]<=0.05:
            output.append("Unit root")
        else:
            output.append("No unit root")
        kpss_output[nucleo_deter[i]] = output
    print_full(kpss_output)
```

```
#Definizione funzione test PhillipsPerron
def pp_test(timeseries):
    from arch.unitroot import PhillipsPerron
    #dictionary con vari nuclei deterministici
    nucleo_deter= {"n" : "No constant", "c" : "Constant",
                   "ct" : "Constant and Linear Trend"}
    for i in nucleo_deter:
        pp = PhillipsPerron(returns["Portfolio returns"], trend=i)
        print(pp.summary().as_text())
        print(" ", end="\n")

#Definizione funzione elemento più vicino
def find_nearest(array, value):
    array = np.asarray(array)
    #idx = indice del valore della serie che minimizza la distanza dal
    #value dato in input
    idx = (np.abs(array - value)).argmin()
    return idx

#definizione funzione simulazione FHS
def fhs_simulation(N, T, last_port_price, returns_series, model_arma,
                   model_garch, bootstrap_library=True):

    '''Variabili funzione:
        N = numero di simulazioni
        T = orizzonte temporale di simulazione
        last_port_price = ultimo prezzo della serie storica dei
        prezzi di portafoglio
        returns_series = serie storica dei rendimenti
        giornalieri di portafoglio
        model_arma = oggetto restituito dal modello ARMA
        model_garch = oggetto restituito dal modello GARCH
        bootstrap_library = flag sulla modalità di simulazione:
            se bootstrap_library == True --> ricampionamento
            bootstrap con libreria di Kevin Sheppard
            se bootstrap_library == False --> ricampionamento
            con metodo del posizionale da uniforme continua (0,1)
    '''
```

```python
import pandas as pd
import numpy as np

p = len(model_arma.arparams) #grado del polinomio AR (ARMA)
q = len(model_arma.maparams) #grado del polinomio MA (ARMA)
m = len(list(filter(lambda x: "alpha" in x, model_garch._names)))
#grado del polinomio AR (GARCH)
s = len(list(filter(lambda x: "beta" in x, model_garch._names)))
#grado del polinomio dei residui al quadrato (GARCH)
garch_params = model_garch._params #parametri ottenuti dal modello GARCH
arma_params = model_arma.params #parametri ottenuti dal modello ARMA
residuals_arima = model_arma.resid #serie residui modello ARMA
volatility_garch = model_garch.conditional_volatility
#serie volatilità condizionale modello GARCH
const_arma = len(model_arma.params)-p-q #presenza costante nell'ARMA:
    #const_arma==1 (constante presente)
    #const_arma==0 (constante assente)
const_garch = len(model_garch._params)-m-s
    #se const_garch == 1 --> costante presente
    #se const_garch == 0 --> costante assente

#salvataggio varianza condizionale storica
variance= volatility_garch**2

#scaling dei residui
e = residuals_arima/volatility_garch

#Creazione df, liste e dictionary vuoti (da popolare nel ciclo)
e_star=pd.DataFrame(columns=list(pd.Series(range(1,T+1))),
                    index=range(1,N+1), dtype=float) #df e_star
variance_hat=pd.DataFrame(columns=list(pd.Series(range(1,T+1))),
                    index=range(1,N+1), dtype=float) #df volatilità (GARCH)
z_star=pd.DataFrame(columns=list(pd.Series(range(1,T+1))),
                    index=range(1,N+1), dtype=float) #df z_star=e_star*h_hat
r_star=pd.DataFrame(columns=list(pd.Series(range(1,T+1))),
                    index=range(1,N+1), dtype=float) #df rendimenti (ARMA)
p_star=pd.DataFrame(columns=list(pd.Series(range(1,T+1))),
                    index=range(1,N+1), dtype=float) #df prezzi simulati
```

```
#gestione parametri e osservazioni tramite numpy e pandas
arma_params=np.array(arma_params) #parametri ARMA impostati come array
arma_params=np.insert(arma_params, len(arma_params), 1)
#aggiungo 1 al vettore dei parametri (parametro per z_star_t+i)
arma_vars=pd.DataFrame(columns=list(pd.Series(range(1,N+1))),
                       index=range(1,len(arma_params)+1)) #df variabili arma
garch_params=np.array(garch_params) #parametri GARCH impostati come array
garch_vars=pd.DataFrame(columns=list(pd.Series(range(1,N+1))),
                        index=range(1,len(garch_params)+1)) #df variabili arma

'''Ciclo di simulazione'''

for t in range(1, T+1, 1): #iterazione del tempo
    #condizione if sul ricampionamento bootstrap
    if bootstrap_library==True:
        from arch.bootstrap import IndependentSamplesBootstrap
        import random
        bs_series=dict.fromkeys(list(pd.Series(range(1,N+1))))
        #serie vuota bootstrap con dictionary: key=id_simulation
        bs = IndependentSamplesBootstrap(e=e)
        i=0 #iteratore necessario per ciclo bootstrap
        for data in bs.bootstrap(N):
            i+=1
            bs_series[i]=bs.e #salvataggio dataset ricampionati
            e_star.iloc[i-1,t-1]=bs_series[i][random.randint(0,len(e)-1)]
            #estrazione casuale e_star

    for sim in range(1, N+1, 1):
        #condizione if sul ricampionamento bootstrap
        if bootstrap_library==False:
            from scipy.stats import uniform
            y = 1. * np.arange(len(e)) / (len(e) - 1) #funzione di ripartizione
            u = uniform.rvs() #estrazione casuale da uniforme continua (0,1)
            e_star.iloc[sim-1,t-1]=e[find_nearest(y,u)] #estrazione casuale e_star
```

```
        if t==1: #condizione utilizzo variabili storiche
        #(es. residui storici e volatilità storica: No valori simulati)
            #calcolo varianza_t+1 --> nota 17 (vedi tesi)
            if const_garch==1:
                garch_vars.loc[:, sim] = np.concatenate(([1],
                residuals_arima[0:m].values**2, variance[0:s].values ), axis=None)
                pos_temp_garch=1 #posizione inserimento occorrenza
                #(vedi linea 187) (NO CONST=0, CONST=1)
            else:
                garch_vars.loc[:, sim] = np.concatenate((
                residuals_arima[0:m].values**2, variance[0:s].values ), axis=None)
                pos_temp_garch=0 #posizione inserimento occorrenza
                #(vedi linea 187) (NO CONST=0, CONST=1)
            variance_hat.loc[sim,t]=np.dot(garch_params.T,
                                           np.array(garch_vars.loc[:, sim]))
            #calcolo residui simulati
            z_star.loc[sim,t]=np.dot(e_star.loc[sim,t],
                                     np.sqrt(variance_hat.loc[sim,t]))
            #costruzione vettore osservazioni da moltiplicare con i parametri
            #vettore variabili arma(p,q)
            if const_arma==1:
                arma_vars.loc[:,sim]=np.concatenate(([1], returns_series[0:p].values,
                        residuals_arima[0:q].values,[z_star.loc[sim,t]] ), axis=None)
                pos_temp_arma=1 #posizione inserimento occorrenza
                #(vedi linea 202) (NO CONST=0, CONST=1)
            else:
                arma_vars.loc[:,sim]=np.concatenate((returns_series[0:p].values,
                     residuals_arima[0:q].values,[z_star.loc[sim,t]] ), axis=None)
                pos_temp_arma=0 #posizione inserimento occorrenza
                #(vedi linea 202) (NO CONST=0, CONST=1)
            #calcolo rendimento simulato t+1 (r_star_t+1)
            r_star.loc[sim,t]=np.dot(arma_params.T,np.array(arma_vars.loc[:,sim]))
            #calcolo prezzo simulato t+1 (p_star_t+1)
            p_star.loc[sim,t]=last_port_price+last_port_price*r_star.loc[sim,t]
```

```
        else:
            ##calcolo varianza simulata t+i con i>1
            temp = np.insert(garch_vars.loc[:,sim].values,
                             pos_temp_garch, z_star.loc[sim,t-1]**2)
            temp[m+1]=variance_hat.loc[sim,t-1]
            temp=np.delete(temp,-1)
            garch_vars.loc[:,sim] = temp
            #calcolo varianza simulata
            variance_hat.loc[sim,t]=np.dot(garch_params.T,
                                           garch_vars.loc[:, sim]) #eq.9 (vedi tesi)
            #calcolo residui simulati t+i con i>1
            z_star.loc[sim,t]=np.dot(e_star.loc[sim,t],np.sqrt(variance_hat.loc[sim,t]))
            #eq.6 (vedi tesi)
            #modifica vettore arma_vars (rolling delle osservazioni)
            #utilizzo della variabile temporanea temp
            temp=np.insert(arma_vars.loc[:,sim].values,pos_temp_arma,r_star.loc[sim,t-1])
            temp[p+1]=z_star.loc[sim,t-1]
            temp[p+q+1]=z_star.loc[sim,t]
            temp=np.delete(temp,-1)
            arma_vars.loc[:,sim]=temp #fine uso temp
            #calcolo rendimenti simulati t+i con i>1
            r_star.loc[sim,t]=np.dot(arma_params.T,np.array(arma_vars.loc[:,sim]))
            #calcolo prezzi simulati t+i con i>1
            p_star.loc[sim,t]=p_star.loc[sim,t-1]+p_star.loc[sim,t-1]*r_star.loc[sim,t]
    output_simulation= {"Rendimenti simulati" : r_star, "Prezzi simulati" : p_star,
                        "Errori simulati" : z_star, "Varianza simulata" : variance_hat}
    return output_simulation
```

```python
#Definizione simulazione ARMA-GARCH gaussiano
def gaussian_simulation(N, T, last_port_price, model_arma, model_garch):

    prediction, se, conf = model_arma.forecast(T, alpha=0.01)
    #predizione modello ARMA
    forecasts_garch = model_garch.forecast(horizon=T, start=0, reindex=False)
    #predizione modello GARCH
    forecasts_volatility=np.sqrt(forecasts_garch.residual_variance.iloc[0:1])
    #volatilità predetta

    #creazione dei dataframe vuoti --> saranno popolati nel ciclo
    r_gaussian=pd.DataFrame(columns=list(pd.Series(range(1,T+1))),
                            index=range(1,N+1), dtype=float)
    p_gaussian=pd.DataFrame(columns=list(pd.Series(range(1,T+1))),
                            index=range(1,N+1), dtype=float)

    #Ciclo di simulazione
    for t in range(1, T+1, 1):
        for sim in range(1, N+1,1):
            #Simulazione del rendimento
            r_gaussian.loc[sim,t]=prediction[t-1]+np.random.normal(loc=0.0,
                        scale=1.0, size=1)*forecasts_volatility.iloc[0,t-1]
            #calcolo del prezzo simulato
            if t==1:
                p_gaussian.loc[sim,t]=last_port_price+last_port_price*r_gaussian.loc[sim,1]
            else:
                p_gaussian.loc[sim,t]=p_gaussian.loc[sim,t-1]+p_gaussian.loc[sim,t-1]*r_gaussian.loc[sim,t]
    output_gaussian_simulation= {"Rendimenti simulati" : r_gaussian, "Prezzi simulati" : p_gaussian}
    return output_gaussian_simulation
```

```python
#Definizione funzione test di Kupiec
def POF_test(T1, T_test, alpha=0.01):
    from scipy.stats import chi2
    '''POF_test restituisce l'output del test di Kupiec (Proportion of Failure);
        Le variabili richieste sono:
            a) T1= Numero delle violazioni osservate
            b) T_test= Ampiezza del campione analizzato
            c) alpha= failure rate atteso (coincide con alpha della misura VaR)
            d) alpha_test = livello di significatività del test:
                    alpha_test=0.01, 0.05, 0.10
    '''
    T0=T_test-T1
    POF_test_statistic=-2*np.log(((1-alpha)**T0*alpha**T1)/((1-T1/T_test)**T0*(T1/T_test)**T1))
    chi_square_test = chi2.cdf(POF_test_statistic, 1)  # one degree of freedom
    p_value=1-chi_square_test
    alpha_test_list=[0.01, 0.05, 0.1]
    result=[]
    for alpha_test in alpha_test_list:
        if p_value > alpha_test:
            result.append("Fail to reject H0")
        else:
            result.append("Reject H0")

    return {
        "statistic test": np.round(POF_test_statistic,4),
        "p_value": np.round(p_value,4),
        "null hypothesis": f"Probability of failure is {round(alpha,3)}",
        "result: alpha=" +str(alpha_test_list[0]): result[0],
        "result: alpha=" +str(alpha_test_list[1]): result[1],
        "result: alpha=" +str(alpha_test_list[2]): result[2]
    }
```

```python
#Definizione Funzione test di indipendenza di Christoffersen
def indipendence_test(Hits, method_backtesting=0):
    from scipy.stats import chi2
    '''indipendence_test restituisce l'output del test di indipendenza di Christoffersen;
        Le variabili richieste sono:
            a) Hits = dataframe contente le violazioni
            b) method_backtesting= Metodo di backtesting utilizzato:
                0 = FHS
                1 = Gaussiana storica
                2 = Gaussiana parametrica
            c) alpha_test = livello di significatività del test:
                    alpha_test=0.01, 0.05, 0.10
    '''
    T1=Hits.iloc[:, method_backtesting].sum() #n. violazioni
    T_test=len(Hits) #Ampiezza del campione analizzato
    T0= T_test-T1 # n. no violazioni
    #Contatori
    T00=0 #no violazione -- no violazione
    T01=0 #no violazione --  violazione
    T10=0 #violazione -- no violazione
    T11=0 #violazione -- violazione
    for i in range(T_test-1, 0, -1): #ciclo di calcolo dei contatori
        if Hits.iloc[i, method_backtesting]==0:
            if Hits.iloc[i-1, method_backtesting]==0:
                T00+=1
            else:
                T01+=1
        else:
            if Hits.iloc[i-1, method_backtesting]==0:
                T10+=1
            else:
                T11+=1
    #Probabilità associate ai contatori
    pigreco01=T01/(T00+T11) #probabilità di no violazione -- violazione
    pigreco11=T11/(T10+T11) #probabilità di violazione -- violazione
    pigreco00=1-pigreco01 #probabilità di no violazione -- no violazione
    pigreco10=1-pigreco11 #probabilità di violazione -- no violazione
```

```python
    LR_test_statistic=-2*np.log(((1-T1/T_test)**T0*(T1/T_test)**T1)/
                                (pigreco00**T00*pigreco01**T01*pigreco10**T10*pigreco11*T11))
    chi_square_test = chi2.cdf(LR_test_statistic, 1)  # one degree of freedom
    p_value=1-chi_square_test
    alpha_test_list=[0.01, 0.05, 0.1]
    result=[]
    for alpha_test in alpha_test_list:
        if p_value > alpha_test:
            result.append("Fail to reject H0")
        else:
            result.append("Reject H0")

    return {
        "statistic test": np.round(LR_test_statistic,4),
        "p_value": np.round(p_value,4),
        "null hypothesis": "Hits are indipendent over time",
        "result: alpha=" +str(alpha_test_list[0]): result[0],
        "result: alpha=" +str(alpha_test_list[1]): result[1],
        "result: alpha=" +str(alpha_test_list[2]): result[2]
    }

#Test conditional_coverage
def conditional_coverage_test(POF_test, indipendence_test, method_backtesting=0):
    from scipy.stats import chi2
    '''conditional_coverage_test restituisce l'output del joint test (POF + indipendence) di Christoffersen;
        Le variabili richieste sono:
            a) POF_test = dictionary del test POF
            b) indipendence_test = dictionary del test di indipendence
            c) method_backtesting= Metodo di backtesting utilizzato:
                0 = FHS
                1 = Gaussiana storica
                2 = Gaussiana parametrica
            d) d) alpha_test = livello di significatività del test:
                    alpha_test=0.01, 0.05, 0.10
    '''
```

```
    LR_uc=POF_test["statistic test"] #POF test
    LR_ind=indipendence_test["statistic test"] #indipendence test

    LR_test_statistic=LR_uc+LR_ind
    chi_square_test = chi2.cdf(LR_test_statistic, 2)  # one degree of freedom
    p_value=1-chi_square_test
    alpha_test_list=[0.01, 0.05, 0.1]
    result=[]

    for alpha_test in alpha_test_list:
        if p_value > alpha_test:
            result.append("Fail to reject H0")
        else:
            result.append("Reject H0")

    return {
        "statictic test": np.round(LR_test_statistic,4),
        "p_value": np.round(p_value,4),
        "null hypothesis": "Hits are indipendent over time and average number of violations is correct",
        "result: alpha=" +str(alpha_test_list[0]): result[0],
        "result: alpha=" +str(alpha_test_list[1]): result[1],
        "result: alpha=" +str(alpha_test_list[2]): result[2]
    }

'''IMPORTAZIONE DEI PREZZI DEGLI ETF EUROPEI'''
#Import Invesco EURO STOXX 50 ETF
Invesco = pd.read_excel("Invesco EURO STOXX 50 UCITS ETF Acc.xlsx", skiprows=32, usecols="A:B")
#Import iShares Core MSCI Europe ETF
iSharesCore = pd.read_excel("iShares Core MSCI Europe UCITS ETF EUR (Acc).xlsx", skiprows=29, usecols="A:B")
#Import iShares MSCI Europe Financials ETF
iSharesFin = pd.read_excel("iShares MSCI Europe Financials ETF.xlsx", skiprows=29, usecols="A:B")
#Import SPDR MSCI Europe Industrials UCITS ETF
Spdr = pd.read_excel("SPDR MSCI Europe Industrials UCITS ETF.xlsx", skiprows=30, usecols="A:B")
#Import Vanguard FTSE Developed Europe ETF
Vanguard = pd.read_excel("Vanguard FTSE Developed Europe UCITS ETF EUR D.xlsx", skiprows=28, usecols="A:B")
#Import Xtrackers STOXX Europe 600
Xtrackers = pd.read_excel("Xtrackers STOXX Europe 600.xlsx", skiprows=29, usecols="A:B")
```

```
'''Intersezione dataframes con funzione reduce'''
import functools
#salvo i df in una tupla
dfs = (Invesco, iSharesCore, iSharesFin, Spdr, Vanguard, Xtrackers)
#intersenzione dfs
df_prices = functools.reduce(lambda left,right: pd.merge(left,right,on='Exchange Date'), dfs)
#Filling i Nan con il metodo backfill
df_prices.fillna(method="backfill", inplace=True)
#cambio formato data
df_prices["Exchange Date"]=df_prices['Exchange Date'].dt.strftime('%d/%m/%Y')

'''Plot prezzi storici'''
import matplotlib.pyplot as plt
for i in range(1, df_prices.shape[1]-1):
    plt.plot(df_prices["Exchange Date"], df_prices.iloc[:,i]/df_prices.iloc[-1,i]*100, label=df_prices.columns[i], linewidth=1)
plt.title("Relative Daily ETF Closings", fontweight="bold", fontsize=10)
plt.xlabel("Time")
plt.xlim(xmin="09/12/2014", xmax="10/10/2022")
plt.xticks(["15/01/2015", "15/01/2016", "17/01/2017", "16/01/2018", "15/01/2019", "15/01/2020", "15/01/2021", "14/01/2022"],
           [2015, 2016, 2017, 2018, 2019, 2020,2021,2022], rotation ='vertical')
plt.ylabel("ETF Price")
plt.legend(fontsize=7)
plt.grid()
plt.show()


'''Creazione dataset Rendimenti'''
#calculating log-returns
import numpy as np
#creazione dataframe vuoto
returns= pd.DataFrame()
#calcolo log returns con lambda function
returns = df_prices[list(df_prices.columns[df_prices.columns.str.\
                  contains("Close")])].pct_change().apply(lambda x: np.log(1+x))
#inserimento colonna date
returns.insert(0, "Date", value=df_prices["Exchange Date"])
#set index colonna Date
returns.set_index('Date', inplace=True)
```

```
#rimozione prima riga piena di NaN
returns.drop(index=returns.index[0],
        axis=0,
        inplace=True)
#cambio nomi colonne rendimenti
returns = returns.rename(columns={'Invesco - Close' : 'Invesco - Returns',
                                  'iSharesCore - Close' : 'iSharesCore - Returns',
                                  'iSharesFin - Close' : 'iSharesFin - Returns',
                                  'SPDR - Close' : 'SPDR - Returns',
                                  'Vanguard - Close' : 'Vanguard - Returns',
                                  'Xtrackers - Close' : 'Xtrackers - Returns' } )

''' Calcolo prezzi e rendimenti portafoglio'''
#Calcolo dei pesi (EW Portfolio)
weights =  np.repeat(1/returns.shape[1], repeats=returns.shape[1])
#Prezzi portafoglio
df_prices["Portfolio prices"] = df_prices.iloc[:, 1:].mul(weights, axis=1).sum(axis=1)
#Rendimenti portafoglio
returns["Portfolio returns"] = returns.iloc[:, :].mul(weights, axis=1).sum(axis=1)

'''Grafico prezzi e rendimenti portafoglio'''
plt.subplot(2,1,1)
plt.plot(df_prices["Exchange Date"], df_prices["Portfolio prices"], label="Portfolio Prices")
plt.title("Daily Portfolio Closings", fontweight="bold", fontsize=10)
plt.xlim(xmin="09/12/2014", xmax="07/10/2022")
plt.xticks(["15/01/2015", "15/01/2016", "17/01/2017", "16/01/2018", "15/01/2019", "15/01/2020", "15/01/2021", "14/01/2022"],
           [2015, 2016, 2017, 2018, 2019, 2020,2021,2022], color="w")
plt.yticks(list(np.linspace(40, 100, 7)))
plt.grid()
plt.ylabel(" Closing Level")
```

```
plt.subplot(2,1,2)
plt.plot(returns.index, returns["Portfolio returns"]*100, label="Portfolio Returns")
plt.title("Historical Portfolio Returns", fontweight="bold", fontsize=10)
plt.xlabel("Time")
plt.xlim(xmin="09/12/2014", xmax="07/10/2022")
plt.yticks(list(np.linspace(-10, 10, 9)))
plt.xticks(["15/01/2015", "15/01/2016", "17/01/2017", "16/01/2018", "15/01/2019", "15/01/2020", "15/01/2021", "14/01/2022"],
           [2015, 2016, 2017, 2018, 2019, 2020,2021,2022], rotation ='vertical')
plt.ylabel("Portfolio Returns (%)")
plt.grid()
plt.show()

#definizione indice dataframe returns
returns.index=df_prices.iloc[0:1952,0]

'''Analisi esplorativa'''
#head del dataframe
head = returns.iloc[:, :-1].head()
#Statistiche descrittive
descr = returns.describe()
print_full(descr)

#Curtosi e asimmetria
from scipy.stats import kurtosis, skew
skew_empirical = round(skew(returns["Portfolio returns"]),3)
kurtosis_empirical = round(kurtosis(returns["Portfolio returns"]),3)
print(f"L'assimetria della distribuzione empiriaca è pari a {skew_empirical}\n")
print("L'assimetria della distribuzione normale è pari a 0")
print(f"L'eccesso di curtosi della distribuzione empirica è pari a {kurtosis_empirical}")
print("L'eccesso di curtosi della distribuzione normale è pari a 3 \n")
```

```python
#Test di normalità
from scipy import stats
jarque_bera_test = stats.jarque_bera(returns["Portfolio returns"])

'''Test di radice unitaria sui rendimenti del portafoglio'''
adf_test(returns["Portfolio returns"])
kpss_test(returns["Portfolio returns"])
pp_test(returns["Portfolio returns"])

'''VaR storico e fitting Gaussiana'''
from scipy.stats import norm
#Istogramma dei rendimenti empirici
plt.hist(returns["Portfolio returns"], bins=1000, density=True, alpha=0.6, color='b', label="Empirical Returns")
#Fitting della normale sui dati
mu, std = norm.fit(returns["Portfolio returns"])
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, len(returns["Portfolio returns"]))
p = norm.pdf(x, mu, std)
#Plot distribuzione normale
plt.plot(x, p, 'r', linewidth=1, label="Fitted Returns (Normal)")
plt.title("Empirical vs Normal Distribution", fontweight="bold", fontsize=10)
plt.xlabel("Returns")
plt.ylabel("Frequency")
plt.grid()
plt.legend(loc="best", fontsize=8)
plt.ylim(top=70)
plt.show()

#Calcolo VaR empirico e VaR della distribuzione normale (ultimi 252 giorni)
historical_var_daily = np.quantile(returns.iloc[0:253,-1], 0.01)
mu, std = norm.fit(returns.iloc[0:253,-1])
var_gauss_annualizzato= mu*252 - 2.33*std*(252)**0.5
annual_return=(df_prices.iloc[0,-1]/df_prices.iloc[253,-1])-1 #rendimento annuo
print(historical_var_daily*100, annual_return*100, var_gauss_annualizzato*100)
```

```python
#Autocorrelazione rendimenti portafoglio
#ACF
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(returns["Portfolio returns"], lags=30)
plt.xlabel("Lag")
plt.ylabel("Sample Autocorrelation")
plt.ylim(-0.2,0.4)
plt.title("Sample ACF of Returns", fontweight="bold", fontsize=10)
plt.grid()
plt.show()
#PACF
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(returns["Portfolio returns"], lags=30)
plt.xlabel("Lag")
plt.ylabel("Partial Autocorrelation")
plt.ylim(-0.2,0.4)
plt.title("Sample PACF of Returns", fontweight="bold", fontsize=10)
plt.grid()
plt.show()
```

```
'''MODELLO ARIMA'''
from statsmodels.tsa.arima_model import ARIMA
#creazione dataframe con IC
df_aic=pd.DataFrame(data=np.zeros((8,8)))
df_bic=pd.DataFrame(data=np.zeros((8,8)))
df_hqic=pd.DataFrame(data=np.zeros((8,8)))
#ciclo for per e salvataggio IC
for i in range(1,8,1):
    for j in range(1,8,1):
        arima_model = ARIMA(returns["Portfolio returns"], order=(i,0,j))
        model = arima_model.fit(method="mle", disp=0)
        df_aic.iloc[i,j] =  model.aic
        df_bic.iloc[i,j] =  model.bic
        df_hqic.iloc[i,j] =  model.hqic
#Best I and J per AIC
aic = df_aic.where(df_aic==df_aic.min().min()).dropna(how='all').dropna(axis=1)
aic_lags=(aic.index,aic.columns)
#Best I and J per BIC
bic = df_bic.where(df_bic==df_bic.min().min()).dropna(how='all').dropna(axis=1)
bic_lags=(bic.index,bic.columns)
#Best I and J per HQIC
hqic = df_hqic.where(df_hqic==df_hqic.min().min()).dropna(how='all').dropna(axis=1)
hqic_lags = (hqic.index,hqic.columns)
#Print totale
print(aic_lags, bic_lags, hqic_lags)

df_aic.to_excel('aic.xlsx', index=False)
df_bic.to_excel('bic.xlsx', index=False)
df_hqic.to_excel('hqic.xlsx', index=False)

'''BEST MODEL: FITTING'''
arima_model = ARIMA(returns["Portfolio returns"], order=(3,0,3))
model_arma = arima_model.fit(method="mle", disp=0)
#residui arima
residuals_arima = model_arma.resid
print(model_arma.summary())
```

```
'''Plot Roots'''
figure, axes = plt.subplots()
#asse y
plt.plot(np.repeat(0,20), np.linspace(-2,2,20), linestyle='dashed', color="black", zorder=5, linewidth=1.0)
#asse x
plt.plot(np.linspace(-2,2,20),np.repeat(0,20), linestyle='dashed', color="black", zorder=5, linewidth=1.0)
#Cerchio unitario
Drawing_uncolored_circle = plt.Circle( (0, 0 ),
                                       1 ,
                                       fill = False , linewidth=1.0, linestyle='dashed', zorder=10)
axes.set_aspect( 1 )
axes.add_artist( Drawing_uncolored_circle )
#Plot radici AR
for root in range(len(model_arma.arroots)):
    a, = plt.plot(model_arma.arroots[root].real, model_arma.arroots[root].imag, marker="x",
                  markersize=10, markeredgecolor="red", markerfacecolor="red", label="AR ROOTS", zorder=20)
#Plot radici MA
for root in range(len(model_arma.maroots)):
    b, = plt.plot(model_arma.maroots[root].real, model_arma.maroots[root].imag, marker=".",
                  markersize=10, markeredgecolor="green", markerfacecolor="green", label="MA ROOTS", zorder=15)
plt.title( 'ARMA(3,3): ROOTS' )
plt.xlabel("Real")
plt.ylabel("Imaginary")
plt.xlim(-2,2)
plt.xticks(ticks=np.linspace(-2,2,5))
plt.yticks(ticks=np.linspace(-2,2,5))
plt.ylim(-2,2)
plt.grid()
plt.legend(loc="upper left", fontsize=10, handles=[a, b])
plt.show()
```

```
'''DIAGNOSTICA'''
#Ljung-Box for autocorrelation ---> ARMA Residuals
import statsmodels.api as sm
ljung_box = sm.stats.acorr_ljungbox(residuals_arima, lags=np.linspace(7,30,24), model_df=6, return_df=True)
ljung_box["Result"]=np.zeros(shape=len(ljung_box))
for i in range(0,len(ljung_box),1):
    if ljung_box.iloc[i, 1]>0.05:
        ljung_box.iloc[i,2]="No autocorrelation"
    else:
        ljung_box.iloc[i,2]="Autocorrelation"
#Plot Trend Ljung Box P-value
plt.plot(ljung_box.index, ljung_box["lb_pvalue"], label="Ljung-Box P-Value")
plt.title("Arma Residuals - Ljung-Box Trend", fontweight="bold", fontsize=10)
plt.xlabel("Lags")
plt.ylabel("Ljung-Box P-Value")
plt.plot(ljung_box.index, np.repeat(0.05,len(ljung_box.index)), linestyle='dashed', label="P-Value=0.05")
plt.xlim(7,30)
plt.xticks(ticks=np.linspace(7,30,24))
plt.grid()
plt.legend(loc="best")
plt.show()


'''MODELLO GARCH'''
from arch import arch_model
am = arch_model(residuals_arima, mean="Zero",lags=1, vol="Garch", p=1, o=0, q=1, dist="Normal")
model_garch = am.fit(disp="off")
#salvataggio volatilità
volatility = model_garch.conditional_volatility
#salvataggio varianza
variance= volatility**2
variance.rename("variance", inplace=True)
print(model_garch.summary())
```

```
'''Grafico residui e conditional volatility'''
plt.subplot(2,1,1)
plt.plot(returns.index, residuals_arima , label="Residuals")
plt.title("ARMA Residuals", fontweight="bold", fontsize=10)
plt.xlim(xmin="09/12/2014", xmax="07/10/2022")
plt.xticks(["15/01/2015", "15/01/2016", "17/01/2017", "16/01/2018", "15/01/2019", "15/01/2020", "15/01/2021", "14/01/2022"],
           [2015, 2016, 2017, 2018, 2019, 2020,2021,2022], color="w")
plt.grid()
plt.ylabel("Residuals")

plt.subplot(2,1,2)
plt.plot(returns.index, volatility, label="Conditional Volatility")
plt.title("Conditional Volatility", fontweight="bold", fontsize=10)
plt.xlabel("Time")
plt.yticks([0, 0.01, 0.02, 0.03, 0.04, 0.05])
plt.xlim(xmin="09/12/2014", xmax="07/10/2022")
plt.xticks(["15/01/2015", "15/01/2016", "17/01/2017", "16/01/2018", "15/01/2019", "15/01/2020", "15/01/2021", "14/01/2022"],
           [2015, 2016, 2017, 2018, 2019, 2020,2021,2022], rotation ='vertical')
plt.ylabel("Conditional Volatility")
plt.grid()
plt.show()


#Scaling dei residui
e = residuals_arima/volatility
```

```
'''Ljung-Box for autocorrelation after scaling'''
ljung_box_e = sm.stats.acorr_ljungbox(e, np.linspace(7,30,24), model_df=6, return_df=True)
ljung_box_e["Result"]=np.zeros(shape=len(ljung_box_e))
for i in range(0,len(ljung_box_e),1):
    if ljung_box_e.iloc[i, 1]>0.05:
        ljung_box_e.iloc[i,2]="No autocorrelation"
    else:
        ljung_box_e.iloc[i,2]="Autocorrelation"
#Plot Trend Ljung Box P-value
plt.plot(ljung_box_e.index, ljung_box_e["Lb_pvalue"], label="Ljung-Box P-Value")
plt.title("Standardized Residuals - Ljung-Box Trend", fontweight="bold", fontsize=10)
plt.xlabel("Lags")
plt.ylabel("Ljung-Box P-Value")
plt.plot(ljung_box_e.index, np.repeat(0.05,len(ljung_box_e.index)), linestyle='dashed', label="P-Value=0.05")
plt.xlim(7,30)
plt.xticks(ticks=np.linspace(7,30,24))
plt.grid()
plt.legend(loc="best")
plt.show()

#Plot Std residuals vs Unit Variance Residuals
unit_var_resid = residuals_arima / residuals_arima.std()
df = pd.concat([e, unit_var_resid], 1)
df.columns = ["Std Resids", "Unit Variance Resids"]
subplot = df.plot(kind="kde", xlim=(-2.5,2.5), title="Standardized Residuals vs Unit Variance residuals", grid=True)


#SIMULAZIONE FHS
N=10000 #N simulazioni
T=10 #orizzonte di simulazione
last_port_price = df_prices.iloc[0,-1] #ultimo prezzo storico del portafoglio
returns_series = returns["Portfolio returns"] #serie dei rendimenti
model_arma=model_arma
model_garch=model_garch

#simulazione FHS
output_simulation=fhs_simulation(N, T, last_port_price, returns_series, model_arma, model_garch,
                                 bootstrap_library=True)
```

```
#creazione dataframe con varie simulazioni (compresi prezzi passati)
port_prices_fhs=pd.DataFrame(data=df_prices.iloc[0:3,-1])
port_prices_fhs=port_prices_fhs.iloc[::-1]
port_prices_fhs=pd.concat([port_prices_fhs] * (N), axis=1, ignore_index=True)
port_prices_fhs=pd.concat([port_prices_fhs,output_simulation["Prezzi simulati"].transpose()], ignore_index=True)
port_prices_fhs=port_prices_fhs.iloc[:,1:]

'''Calcolo del Var Cumulato'''
cumul_ptf_horizon_fhs=100*(output_simulation["Prezzi simulati"].iloc[:,-1]/last_port_price-1)
VaR_fhs=np.quantile(cumul_ptf_horizon_fhs, 0.01, method="Lower")

#Describe su prezzi e rendimenti al giorno T
output_simulation["Prezzi simulati"].loc[:,T].describe()
output_simulation["Rendimenti simulati"].loc[:,T].describe()

'''Plot rendimenti simulati'''
for sim in range(1, N+1, 1):
    if sim==1:
        plt.plot(port_prices_fhs.index.values[0:3], port_prices_fhs.loc[0:2,sim])
    plt.plot(port_prices_fhs.index.values[2:], port_prices_fhs.loc[2:,sim])
#linea inizio simulazione
plt.plot(np.repeat(2,10), np.linspace(-100,800,10), linestyle='dashed', color="red",
         linewidth=1.0, label="Start of simulation")
plt.xticks(port_prices_fhs.index.values,
           ["06/10/22", "07/10/22", "10/10/22", "11/10/22", "12/10/22", "13/10/22", "14/10/22",
            "17/10/22", "18/10/22", "19/10/22", "20/10/22", "21/10/22", "24/10/22"], rotation="vertical")
plt.xlim(0, T+2)
plt.ylim(port_prices_fhs.min().min()-10, port_prices_fhs.max().max()+10)
plt.title("Portfolio Price FHS Simulation", fontweight="bold", fontsize=9)
plt.xlabel("Time")
plt.ylabel("Value")
plt.grid()
plt.legend(loc="best")
plt.show()
```

```
'''Plot Distribuzione simulata FHS'''
plt.hist(output_simulation["Rendimenti simulati"].loc[:,T], bins=250, density=True, alpha=0.6, color='b', label="Simulated FHS Retur
#Fitting della normale sui dati
mu, std = norm.fit(np.array(output_simulation["Rendimenti simulati"].loc[:,T], dtype=float))
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, len(output_simulation["Rendimenti simulati"].loc[:,T]))
p = norm.pdf(x, mu, std)
skew_simulation = round(skew(output_simulation["Rendimenti simulati"].loc[:,T]),3)
kurtosis_simulation = round(kurtosis(output_simulation["Rendimenti simulati"].loc[:,T]),3)
#Plot distribuzione normale
plt.plot(x, p, 'r', linewidth=1, label="Fitted Returns (Normal)")
plt.title("Simulated (FHS) vs Normal Distribution", fontweight="bold", fontsize=10)
plt.xlabel("Returns")
plt.ylabel("Frequency")
plt.grid()
plt.legend(loc="best", fontsize=8)
plt.text(xmin, 35, "Skewness = " +str(skew_simulation))
plt.text(xmin, 32, "Excess Kurtosis = " +str(kurtosis_simulation))
plt.show()


'''Previsione con distribuzione gaussiana'''
output_gaussian_simulation=gaussian_simulation(N, T, last_port_price, model_arma, model_garch)

#creazione dataframe con varie simulazioni (compresi prezzi passati)
port_prices_gaussian=pd.DataFrame(data=df_prices.iloc[0:3,-1])
port_prices_gaussian=port_prices_gaussian.iloc[::-1]
port_prices_gaussian=pd.concat([port_prices_gaussian] * (N), axis=1, ignore_index=True)
port_prices_gaussian=pd.concat([port_prices_gaussian,output_gaussian_simulation["Prezzi simulati"].transpose()], ignore_index=True)
port_prices_gaussian=port_prices_gaussian.iloc[:,1:]

#Calcolo del VaR gaussiano
cumul_ptf_horizon_gaussian=100*(output_gaussian_simulation["Prezzi simulati"].iloc[:,-1]/last_port_price-1)
VaR_gaussian=np.quantile(cumul_ptf_horizon_gaussian, 0.01, method="lower")

#confronto VaR fhs e Gaussiano
print(VaR_fhs, VaR_gaussian)
```

```
'''Plot Prezzi simulati: FHS vs Gaussian'''
if T==10:

    plt.subplot(1,2,1)
    #plotting portafogli FHS
    for sim in range(1, N+1, 1):
        if sim==1:
            plt.plot(port_prices_fhs.index.values[0:3], port_prices_fhs.loc[0:2,sim])
        plt.plot(port_prices_fhs.index.values[2:], port_prices_fhs.loc[2:,sim])
    #linea inizio simulazione
    plt.plot(np.repeat(2,10), np.linspace(-100,800,10), linestyle='dashed', color="red",
             linewidth=1.0, label="Start of simulation")
    plt.xticks(port_prices_fhs.index.values,
               ["06/10/22", "07/10/22", "10/10/22", "11/10/22", "12/10/22", "13/10/22", "14/10/22",
                "17/10/22", "18/10/22", "19/10/22", "20/10/22", "21/10/22", "24/10/22"], rotation="vertical")
    plt.xlim(0, T+2)
    plt.ylim(port_prices_fhs.min().min()-10, port_prices_fhs.max().max()+10)
    plt.title("Portfolio Price FHS Simulation", fontweight="bold", fontsize=9)
    plt.xlabel("Time")
    plt.ylabel("Value")
    plt.grid()
    plt.legend(loc="best", fontsize=8)


    plt.subplot(1,2,2)
    #plotting portafogli Gaussian
    for sim in range(1, N+1, 1):
        if sim==1:
            plt.plot(port_prices_gaussian.index.values[0:3], port_prices_gaussian.loc[0:2,sim])
        plt.plot(port_prices_gaussian.index.values[2:], port_prices_gaussian.loc[2:,sim])
    #linea inizio simulazione
    plt.plot(np.repeat(2,10), np.linspace(-100,800,10), linestyle='dashed', color="red",
             linewidth=1.0, label="Start of simulation")
    plt.xticks(port_prices_gaussian.index.values,
               ["06/10/22", "07/10/22", "10/10/22", "11/10/22", "12/10/22", "13/10/22", "14/10/22",
                "17/10/22", "18/10/22", "19/10/22", "20/10/22", "21/10/22", "24/10/22"], rotation="vertical")
    plt.xlim(0, T+2)
    plt.ylim(port_prices_fhs.min().min()-10, port_prices_fhs.max().max()+10)
    plt.title("Portfolio Price Gaussian Simulation", fontweight="bold", fontsize=9)
    plt.xlabel("Time")
    plt.grid()
    plt.legend(loc="best", fontsize=8)
    plt.show()
```

```
'''Plot Distribuzione simulata FHS vs Gaussiana'''
#Plot FHS
plt.subplot(2,1,1)
plt.hist(output_simulation["Rendimenti simulati"].loc[:,T], bins=250, density=True, alpha=0.6, color='b', label="Simulated FHS Returns")
#Fitting della normale sui dati
mu, std = norm.fit(np.array(output_simulation["Rendimenti simulati"].loc[:,T], dtype=float))
xmin_fhs, xmax_fhs = plt.xlim()
x = np.linspace(xmin_fhs, xmax_fhs, len(output_simulation["Rendimenti simulati"].loc[:,T]))
p = norm.pdf(x, mu, std)
skew_simulation = round(skew(output_simulation["Rendimenti simulati"].loc[:,T]),3)
kurtosis_simulation = round(kurtosis(output_simulation["Rendimenti simulati"].loc[:,T]),3)
#Plot distribuzione normale
plt.plot(x, p, 'r', linewidth=1, label="Fitted Returns (Normal)")
plt.title("Simulated Distribution: FHS vs Gaussian", fontweight="bold", fontsize=9)
plt.ylabel("Frequency")
plt.grid()
plt.ylim(0,50)
plt.xlim(-0.12, 0.12)
plt.legend(loc="best", fontsize=6)
plt.text(-0.11, 45, "Skewness = " +str(skew_simulation), fontsize=7)
plt.text(-0.11, 42, "Excess Kurtosis = " +str(kurtosis_simulation), fontsize=7)
```

```
#Plot Gaussiana
plt.subplot(2,1,2)
plt.hist(output_gaussian_simulation["Rendimenti simulati"].loc[:,T], bins=250, density=True, alpha=0.6, color='b', label="Simulated Gaussian Returns")
#Fitting della normale sui dati
mu, std = norm.fit(np.array(output_gaussian_simulation["Rendimenti simulati"].loc[:,T], dtype=float))
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, len(output_gaussian_simulation["Rendimenti simulati"].loc[:,T]))
p = norm.pdf(x, mu, std)
skew_simulation = round(skew(output_gaussian_simulation["Rendimenti simulati"].loc[:,T]),3)
kurtosis_simulation = round(kurtosis(output_gaussian_simulation["Rendimenti simulati"].loc[:,T],fisher=True),3)
#Plot distribuzione normale
plt.plot(x, p, 'r', linewidth=1, label="Fitted Returns (Normal)")
plt.xlabel("Simulated Returns")
plt.grid()
plt.legend(loc="best", fontsize=6)
plt.ylabel("Frequency")
plt.ylim(0,50)
plt.xlim(-0.12, 0.12)
plt.text(-0.11, 45, "Skewness = " +str(skew_simulation), fontsize=7)
plt.text(-0.11, 42, "Excess Kurtosis = " +str(kurtosis_simulation), fontsize=7)
plt.show()
```

```
'''BACKTESTING 2020-2022'''
#TEST VAR -->2020-2022
N=10000 #n. simulazioni
t_end=0 #data fine backtesting --> corrisponde all'indice della serie returns
durata_backtesting=504 #indica la durata espressa in giorni del backtesting
t_start=t_end+durata_backtesting
#creazione DataFrame vuoti
VaR=pd.DataFrame(columns=["VaR fhs", "VaR gaussiano storico", "VaR ARMA_GARCH"],
                 index=returns.index[t_end:t_start], dtype=float) #dataframe VaR
Hits=pd.DataFrame(data=np.zeros((durata_backtesting, 3)),
                  columns=["Hits fhs", "Hits gaussiano storico", "Hits ARMA_GARCH"],
                  index=returns.index[t_end:t_start], dtype=int) #dataframe Hits (Hit: violazione del VaR)
#Contatori
fhs_bigger_hg=0 #Contatore |VaR_fhs| > |Var_gaussiano_storico| (hg=historical gaussian)
fhs_bigger_ag=0 #Contatore |VaR_fhs| > |Var_ARMA_GARCH| (ag=armagarch)
#Ciclo di backtesting
for t in range(t_start-1,t_end-1,-1): #range inverso perchè le serie dei rendimenti sono ordinate dal più recente al più vecchio
    #stime parametriche sui rendimenti pre-backtesting
    arima_model = ARIMA(returns.iloc[t:,-1], order=(3,0,3))
    model_arma = arima_model.fit(method="mle", disp=0)
    am = arch_model(model_arma.resid, mean="Zero",lags=1, vol="Garch", p=1, o=0, q=1, dist="Normal")
    model_garch = am.fit(disp="off")
    #Simulazione FHS a 1 giorno
    output_simulation=fhs_simulation(N, 1 , df_prices.iloc[t,-1], returns.iloc[t:,-1], model_arma, model_garch, bootstrap_library=True)
    #Simulazione Gaussiana Parametrica a 1 giorno
    output_gaussian_simulation=gaussian_simulation(N, 1, df_prices.iloc[t,-1], model_arma, model_garch)
    #Calcolo mu e sigma storici (1 anno di riferimento)
    mu_storico=returns.iloc[t:t+253,-1].mean() #media storica
    sigma_storico=returns.iloc[t:t+253,-1].std() #std storica
    #Calcolo VaR --> Confidenza 0.99
    VaR.iloc[t-t_end,0]=np.quantile(output_simulation["Rendimenti simulati"], 0.01)
    VaR.iloc[t-t_end,1]=mu_storico-sigma_storico*2.326
    VaR.iloc[t-t_end,2]=np.quantile(output_gaussian_simulation["Rendimenti simulati"], 0.01)
    #Condizioni per i contatori
    if VaR.iloc[t-t_end,0]<VaR.iloc[t-t_end,1]:
            fhs_bigger_hg+=1
    if VaR.iloc[t-t_end,0]<VaR.iloc[t-t_end,2]:
            fhs_bigger_ag+=1
    if returns.iloc[t-1, -1]<VaR.iloc[t-t_end,0]:
            Hits.iloc[t-t_end,0]=1
    if returns.iloc[t-1, -1]<VaR.iloc[t-t_end,1]:
            Hits.iloc[t-t_end,1]=1
    if returns.iloc[t-1, -1]<VaR.iloc[t-t_end,2]:
            Hits.iloc[t-t_end,2]=1
```

```
#Test di Kupiec (POF)
POF_test_fhs= POF_test(Hits["Hits fhs"].sum(), len(Hits))
POF_test_hg=POF_test(Hits["Hits gaussiano storico"].sum(), len(Hits))
POF_test_ag=POF_test(Hits["Hits ARMA_GARCH"].sum(), len(Hits))
#Test di indipendenza (Christoffersen)
indipendence_fhs=indipendence_test(Hits)
indipendence_hg=indipendence_test(Hits, method_backtesting=1)
indipendence_ag=indipendence_test(Hits, method_backtesting=2)
#Conditional Coverage Test
cc_test_fhs=conditional_coverage_test(POF_test_fhs,indipendence_fhs)
cc_test_hg=conditional_coverage_test(POF_test_hg, indipendence_hg)
cc_test_ag=conditional_coverage_test(POF_test_ag, indipendence_ag)

'''Plot Backtesting 2020-2022'''
#Plot Trend Ljung Box P-value
plt.plot(VaR.index, VaR.iloc[:,0], label="FHS VaR", color="blue", linewidth=1 )
plt.plot(VaR.index, VaR.iloc[:,1], label="Historical Gaussian VaR", color="orangered", linewidth=1)
plt.plot(VaR.index, VaR.iloc[:,2], label="ARMA-GARCH VaR", color="magenta", linewidth=1)
plt.plot(VaR.index, returns.iloc[:504, -1], label="Observed Returns", color="forestgreen", linewidth=1)
plt.title("Backtesting VaR: a comparison (2020-2022)", fontweight="bold", fontsize=10)
plt.xlabel("Time")
plt.ylabel("VaR / Actual Return")
plt.xticks(["07/10/2020", "07/01/2021", "07/04/2021", "07/07/2021", "07/10/2021", "07/01/2022", "07/04/2022", "07/07/2022", "10/10/2022" ],
           ["10/2020", "01/2021", "04/2021", "07/2021", "10/2021", "01/2022", "04/2022", "07/2022", "10/2022"], rotation ='vertical')
plt.xlim(VaR.index[-1], VaR.index[0])
plt.ylim(-0.10, 0.10)
plt.grid()
plt.legend(loc="best")
plt.show()
```

```
'''BACKTESTING 2018-2020'''

#TEST VAR -->2018-2020
N=10000 #n. simulazioni
t_end=504 #data fine backtesting --> corrisponde all'indice della serie returns
durata_backtesting=504 #indica la durata espressa in giorni del backtesting
t_start=t_end+durata_backtesting
#creazione DataFrame vuoti
VaR_2=pd.DataFrame(columns=["VaR fhs", "VaR gaussiano storico", "VaR ARMA_GARCH"],
                   index=returns.index[t_end:t_start], dtype=float) #dataframe VaR
Hits_2=pd.DataFrame(data=np.zeros((durata_backtesting, 3)),
                   columns=["Hits fhs", "Hits gaussiano storico", "Hits ARMA_GARCH"],
                   index=returns.index[t_end:t_start], dtype=int) #dataframe Hits (Hit: violazione del VaR)
#Contatori
fhs_bigger_hg_2=0 #Contatore |VaR_fhs| > |Var_gaussiano_storico| (hg=historical gaussian)
fhs_bigger_ag_2=0 #Contatore |VaR_fhs| > |Var ARMA_GARCH| (ag=ARMA_GARCH)

#Ciclo di backtesting
for t in range(t_start-1,t_end-1,-1): #range inverso perchè le serie dei rendimenti sono ordinate dal più recente al più vecchio
    #stime parametriche sui rendimenti pre-backtesting
    arima_model = ARIMA(returns.iloc[t:,-1], order=(3,0,3))
    model_arma = arima_model.fit(method="mle", disp=0)
    am = arch_model(model_arma.resid, mean="Zero",lags=1, vol="Garch", p=1, o=0, q=1, dist="Normal")
    model_garch = am.fit(disp="off")
    #Simulazione FHS a 1 giorno
    output_simulation=fhs_simulation(N, 1 , df_prices.iloc[t,-1], returns.iloc[t:,-1], model_arma, model_garch, bootstrap_library=True)
    #Simulazione Gaussiana Parametrica a 1 giorno
    output_gaussian_simulation=gaussian_simulation(N, 1, df_prices.iloc[t,-1], model_arma, model_garch)
    #Calcolo mu e sigma storici (1 anno di riferimento)
    mu_storico=returns.iloc[t:t+253,-1].mean() #media storica
    sigma_storico=returns.iloc[t:t+253,-1].std() #std storica
    #Calcolo VaR --> Confidenza 0.99
    VaR_2.iloc[t-t_end,0]=np.quantile(output_simulation["Rendimenti simulati"], 0.01)
    VaR_2.iloc[t-t_end,1]=mu_storico-sigma_storico*2.326
    VaR_2.iloc[t-t_end,2]=np.quantile(output_gaussian_simulation["Rendimenti simulati"], 0.01)
    #Condizioni per i contatori
    if VaR_2.iloc[t-t_end,0]<VaR_2.iloc[t-t_end,1]:
            fhs_bigger_hg_2+=1
    if VaR_2.iloc[t-t_end,0]<VaR_2.iloc[t-t_end,2]:
            fhs_bigger_ag_2+=1
    if returns.iloc[t-1, -1]<VaR_2.iloc[t-t_end,0]:
            Hits_2.iloc[t-t_end,0]=1
    if returns.iloc[t-1, -1]<VaR_2.iloc[t-t_end,1]:
            Hits_2.iloc[t-t_end,1]=1
    if returns.iloc[t-1, -1]<VaR_2.iloc[t-t_end,2]:
            Hits_2.iloc[t-t_end,2]=1
```

```
#Test di Kupiec (POF)
POF_test_fhs_2= POF_test(Hits_2["Hits fhs"].sum(), len(Hits_2))
POF_test_hg_2=POF_test(Hits_2["Hits gaussiano storico"].sum(), len(Hits_2))
POF_test_ag_2=POF_test(Hits_2["Hits ARMA_GARCH"].sum(), len(Hits_2))
#Test di indipendenza (Christoffersen)
indipendence_fhs_2=indipendence_test(Hits_2)
indipendence_hg_2=indipendence_test(Hits_2, method_backtesting=1)
indipendence_ag_2=indipendence_test(Hits_2, method_backtesting=2)
#Conditional Coverage Test
cc_test_fhs_2=conditional_coverage_test(POF_test_fhs_2,indipendence_fhs_2)
cc_test_hg_2=conditional_coverage_test(POF_test_hg_2, indipendence_hg_2)
cc_test_ag_2=conditional_coverage_test(POF_test_ag_2, indipendence_ag_2)

'''Plot Backtesting 2018-2020'''
#Plot Trend Ljung Box P-value
plt.plot(VaR_2.index, VaR_2.iloc[:,0], label="FHS VaR", color="blue", linewidth=1 )
plt.plot(VaR_2.index, VaR_2.iloc[:,1], label="Historical Gaussian VaR", color="orangered", linewidth=1)
plt.plot(VaR_2.index, VaR_2.iloc[:,2], label="ARMA-GARCH VaR", color="magenta", linewidth=1)
plt.plot(VaR_2.index, returns.iloc[504:1008, -1], label="Observed Returns", color="forestgreen", linewidth=1)
plt.title("Backtesting VaR: a comparison (2018-2020)", fontweight="bold", fontsize=10)
plt.xlabel("Time")
plt.ylabel("VaR / Actual Return")
plt.xticks(["01/10/2018", "04/01/2019", "05/04/2019", "05/07/2019", "07/10/2019", "06/01/2020", "06/04/2020", "06/07/2020", "06/10/2020" ],
           ["10/2018", "01/2019", "04/2019", "07/2019", "10/2019", "01/2020", "04/2020", "07/2020", "10/2020"], rotation ='vertical')
plt.xlim(VaR_2.index[-1], VaR_2.index[0])
plt.ylim(-0.15, 0.15)
plt.grid()
plt.legend(loc="best")
plt.show()
```

BIBLIOGRAFIA

Barone-Adesi Giovanni, Bourgoin Frederick and Giannopoulos Kostas (1998), "Don't look back" , Risk magazine, 11, August, 100-104

Barone-Adesi Giovanni, Engle Robert and Mancini Loriano (2008), "GARCH options in incomplete markets", Review of Financial Studies, 21, 1223 – 1258

Barone-Adesi Giovanni, Giannopoulos Kostas and Vosper Les (1999), "VaR without correlations for non-linear portfolios", Journal of Futures Markets, 19, August, 583-602.

Barone-Adesi Giovanni, Giannopoulos Kostas and Vosper Les (2002), "Backtesting derivative portfolios with filtered historical simulation", European Financial Management, 8, 1, 31-58

Beveridge S. and Nelson C.R. (1981) "A new approach to decomposition of economic time series into permanent and transitory components with particular attention to measurement of the 'business cycle'", Journal of Monetary Economics, Volume 7, Issue 2, 1981, Pages 151-174

Bollerslev Tim (1986), "Generalised Autoregressive Conditional Heteroskedasticity", Journal of Econometrics, 31, 307-27

Christoffersen, P.F. – "Elements of Financial Risk Management", Academic Press, 2003

Dickey, D.A. and Fuller, W. A. (1979) "Distribution of the Estimators for Autoregressive Time Series With a Unit Root", Journal of the American Statistical Association

Efron Bradeley e Tibshirani Robert J. (1993), "An Introduction to the Bootstrap"

Green. "Econometric Analysis,", 5th ed., Pearson, 2003.

Hamilton, J.D. "Time Series Analysis". Princeton, 1994.

Hastie T, James G., Tibshirani R., Witten D. – "An Introduction to Statistical Learning", 2nd edition, Springer, 2021

Hull, J.C. - “Risk Management and Financial Institutions”, 5th ed., Wiley, 2018

Jarque, C. and Bera, A. (1980) “Efficient tests for normality, homoscedasticity and serial independence of regression residuals”, 6 Econometric Letters 255-259.

Kupiec, P - “Techniques for Verifying the Accuracy of Risk Measurement Models”, American Enterprise Institute, 1995

Kwiatkowski, D., Phillips, P.C.B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. Journal of Econometrics, 54: 159-178.

Lucchetti, Riccardo “Jack” – “Appunti di analisi delle serie storiche”

MacKinnon, J.G. 1994. “Approximate asymptotic distribution functions for unit-root and cointegration tests. Journal of Business and Economic Statistics 12, 167-76.

MacKinnon, J.G. 2010. “Critical Values for Cointegration Tests.” Queen”s University, Dept of Economics, Working Papers.

Palomba, Giulio – “Dispensa di Econometria delle Serie Storiche”

Phillips, P. C. B. and Perron, P (1988) “Testing for a Unit Root in Time Series Regression”, Biometrika, 75: 335-346