Università degli Studi di Padova

Dipartimento di Scienze Statistiche

Corso di Laurea Triennale in

STATISTICA PER LE TECNOLOGIE E LE SCIENZE

RELAZIONE FINALE

Realizzazione di un laboratorio di serie storiche con R: l'approccio moderno

Relatore: Prof. Francesco Lisi

Dipartimento di Scienze Statistiche

Laureando: Riccardo Calisti

Matricola N 2054351

Anno Accademico 2023/2024

Sommario

In questa relazione vengono descritti e spiegati i risultati di un laboratorio di programmazione in R che aveva la finalità di implementare uno strumento interattivo per l'illustrazione di quantità teoriche e di metodi di analisi relativi ai processi stocastici e all'approccio moderno alle serie storiche. In particolare, sfruttando la libreria shiny di R e altri pacchetti grafici correlati, quali dygraphs e shinydashboard, si è creata un'applicazione che, partendo dal concetto di variabile aleatoria, si sviluppa in esempi sui processi stocastici, per poi mostrare graficamente alcune caratteristiche principali dei processi ARIMA, i passi della procedura di Box e Jenkins e la previsione secondo questi modelli. Conclude l'esposizione una generalizzazione ai modelli SARIMA.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Il software statistico R (R Core Team 2021) offre un ricchissimo insieme di librerie e funzioni per l'analisi delle serie storiche e la loro rappresentazione grafica. Tuttavia, quello che sembra mancare nella, pur cospicua, letteratura in questo campo è uno strumento che raccolga, organizzi e mostri interattivamente tutto questo materiale: uno strumento che possa essere d'interesse sia per lo studente meno esperto che si è avvicinato da poco alla materia, sia per chi invece voglia approfondire o verificare alcuni aspetti meno intuitivi. A tal fine è stato svolto il laboratorio di cui qui si dà una dettagliata relazione, il cui obiettivo ultimo è in effetti quello di rendere intuitivi, visibili, quasi tangibili, quegli argomenti che invece rischiano di non essere compresi appieno se si leggono soltanto le pagine di libri e articoli, perché difficilmente si possono mostrare con questi mezzi.

L'analisi delle serie storiche può essere suddiviso in due approcci: il primo, più tradizionale, legato alle componenti sistematiche che agiscono alla base dei dati osservati; il secondo, più moderno, specializzato soprattutto nello studio delle componenti di errore. Su entrambi questi fronti la teoria (e di conseguenza i linguaggi di programmazione statistici come R) è estremamente vasta e profonda. Pertanto in questa sede si è scelto di concentrarsi solo sull'approccio moderno, anche in considerazione del fatto che contestualmente allo sviluppo di questo progetto ne era già stato svolto un altro basato sull'approccio tradizionale.

Stabilito l'obiettivo e la macroarea di trattazione, non resta che definire più precisamente i dettagli tecnici della realizzazione del progetto stesso: si è deciso per la creazione di un'applicazione, la cui interfaccia grafica è divisa in varie pagine, in modo che ogni pagina tratti uno specifico

argomento. Nel linguaggio R esiste una libreria, *shiny* (Chang, Cheng et al. 2024), in grado di creare un'interfaccia grafica che può raccogliere varie schermate, permettere l'interazione tra utente e applicazione e mostrare grafici e risultati numerici. L'applicazione quindi si basa su questa libreria e sugli altri pacchetti ad essa collegati (ad esempio *shinydashboard* (Chang e Borges Ribeiro 2021) o *dygraphs* (Vanderkam et al. 2018)).
La relazione è organizzata nel modo seguente: nel primo capitolo, dopo questa breve introduzione, verranno presentati un sintetico excursus sulle funzioni e librerie R più rilevanti per l'analisi delle serie storiche e su alcune caratteristiche generali della libreria *shiny*. Nel secondo capitolo verranno descritte, distinte per sezioni, le varie pagine dell'applicazione implementata, corredate di note teoriche che motivano la presenza della pagina stessa tra gli argomenti presenti nell'app. A seguire e concludere alcune osservazioni finali.

1.1 Breve panoramica delle principali librerie e funzioni in R

Secondo quanto detto sopra, ad oggi non sono presenti applicazioni in R che si prefiggano di mostrare ed esemplificare argomenti teorici legati alle serie storiche nello stesso modo interattivo che si vuole realizzare. Sono altresì presenti software addetti all'analisi di serie storiche (lo stesso R ad esempio), o anche (in maniera più limitata) a mostrare alcuni aspetti maggiormente teorici. È dunque conveniente, se non necessario, usare le funzioni rese disponibili da questi software come base algoritmica su cui poi andare a sviluppare l'applicazione. Ci si concentri sul caso del software R. Tralasciando le funzioni più generiche (`plot`, `mean`, le varie funzioni per generare variabili casuali o calcolare le densità, ecc.), si possono menzionare `ts`, per trasformare vettori di dati in un oggetto time series in R, la funzione `arima.sim`, per la generazione di serie da modelli ARIMA, `ARMAacf`, che calcola ACF e PACF teoriche di modelli ARMA, `polyroot`, per il calcolo delle radici di polinomi (l'uso diventerà chiaro nel secondo capitolo), `sarima.sim` della libreria *astsa* (Stoffer e Poison 2024) per la generazione di serie da modelli SARIMA (sono in realtà da prendere in considerazione anche altre funzioni, più conosciute ed utilizzate per la generazione da modelli SARIMA, ad esempio `sim_sarima` della libreria *sarima* (Boshnakov e Halliday 2024), tuttavia la funzione `sarima.sim` si adatta meglio ad essere inserita in un'applicazione quale quella implementata per il progetto, perché permette maggior snellezza del codice per l'inserimento ai fini dell'interazione con l'utente). Fondamentale è poi,

per stima dei parametri di modelli e previsioni la libreria *forecast* (Hyndman et al. 2024): di questo pacchetto sono da ricordare soprattutto la funzione `Arima`, per la stima di parametri di serie SARIMA, e la funzione `forecast`, per le previsioni. Per concludere, sono presenti anche parecchie funzioni in grado di fornire test di vario genere, soprattutto per i residui del modello stimato: della libreria *tseries* (Trapletti e Hornik 2024) la funzione `jarque.bera.test` per il test di Jarque e Bera verificante la normalità dei residui (Jarque e Bera 1980), o anche la funzione `Ljung.Box.2` per il test di Ljung e Box di assenza di correlazione globale residua (Ljung e Box 1978), la funzione `pierce.test` per il test di Pierce di assenza di correlazione residua a ritardi stagionali (Pierce 1978) (le due ultime funzioni appartengono al pacchetto *sse_v4*, sviluppato all'interno del Dipartimento di Scienze Statistiche dell'Università di Padova, e messo a disposizione durante il corso di Serie Storiche per le lauree triennali in Statistica) e infine la funzione `dm.test` per il test di Diebold e Mariano di confronto in efficienza tra le previsioni da due modelli differenti (Diebold e Mariano 1995), sempre della libreria *forecast*.

1.2 La libreria *shiny*

La libreria *shiny* è il mezzo più adatto per poter sfruttare le funzioni e i pacchetti fin qui visti, trasformandoli in strumenti interattivi e di semplice utilizzo per l'utente, grazie all'intuitiva interfaccia grafica. Il pacchetto infatti genera automaticamente una schermata interattiva quando si avvia l'esecuzione dello script R (generalmente chiamato `app.R`) che contiene tutti i blocchi di comandi necessari o i collegamenti ad altri script relativi al funzionamento dell'app (tutti gli aspetti dell'app sono gestibili tramite i soli comandi R). L'utente può, da un menù a tendina sulla sinistra, navigare all'interno di varie pagine, ognuna trattante un determinato argomento. Dal punto di vista dello script R, la schermata intuitiva che appare all'occhio dell'utente finale si trasforma in una struttura piuttosto complessa, divisibile in due macroblocchi che riflettono i due aspetti dell'app, quello computazionale e quello interattivo:

- il primo blocco, l'interfaccia utente (ui), è addetto al collegamento tra l'applicazione e l'utente. È il responsabile della gestione, aspetto e presenza delle varie tipologie di input che l'utilizzatore può modificare per inviare dati da elaborare alla macchina. Le tipologie disponibili (chiamabili e modificabili in base a specifici comandi R) vanno dai semplici

input testuali a quelli numerici, sia discreti che continui, per finire con quelli che offrono un elenco di possibili scelte tra cui selezionarne una (radiobuttons). Oltre a definire le tipologie di input, l'ui li dispone anche sulle varie pagine, oltre a generare il menù a tendina e a organizzare gli spazi per i riquadri che ospitano gli output processati. Per questo secondo compito, è stata sviluppata anche la libreria *shinydashboard* (Chang e Borges Ribeiro 2021), che permette la creazione di spazi dalla grafica più accattivante e moderna. La libreria permette anche di suddividere ulteriormente le funzionalità dell'ui in tre blocchi: il primo è l'header, che modifica il titolo dell'app, il secondo è la sidebar, con l'incarico di gestire il menù a tendina e i collegamenti alle varie pagine dell'applicazione, il terzo infine è il body, addetto alla disposizione nelle schermate degli spazi per gli output.

- il secondo blocco, il server, si occupa di tutto il lavoro invisibile che permette la trasformazione degli input negli output richiesti dall'utente: ricevuti i dati, li trasforma e li elabora, per poi riempire i riquadri degli output. Oltre ad essere il processore degli output, il server ne gestisce anche gli aspetti grafici, dai vari tipi di plot alle tabelle, ai semplici testi scritti.

Un'altra libreria di supporto molto utile per costruire grafici output dall'estetica più funzionale e essi stessi in parte interattivi è *dygraphs* (Vanderkam et al. 2018), sviluppata proprio per le serie storiche. Essa permette di creare grafici che ad esempio permettono all'utilizzatore di ingrandire parti del grafico, o di visualizzare le coordinate di tutti i punti del grafico, oltre a evidenziare o nascondere una o più serie storiche quando nello stesso plot ne compaiono diverse parzialmente sovrapposte.

Esplicitato lo scheletro dello script alla base dell'applicazione, resta da trattare di come effettivamente la libreria *shiny* riesca a trasformare gli input in output e ancora come riesca ad aggiornare automaticamente gli output quando l'utente modifica l'input. Il concetto che muove la libreria è quello di reattività (reactivity). All'interno dell'app sono presenti due oggetti: input, che è una sorta di lista contenente tutti i dati settati dall'utente, ed output, che invece è una lista che raccoglie tutti gli output che devono essere mandati dal server all'applicazione perché li mostri all'utente. Quando l'input viene modificato, automaticamente (reattivamente) esso invia un avviso ad R, che provvede a rileggere l'input e rieseguire il codice usando l'input aggiornato. Tutto ciò avviene all'interno dell'ambiente creato dalla funzione `render`, che verifica e memorizza i collegamenti tra gli input e i rispettivi output, in una, anche complessissima, ragnatela di

legami che fanno sì che al modificarsi di un dato di input tutti gli output che condividono quel dato vengano riaggiornati. Inoltre, a differenza del normale comportamento di R, ai fini di un minor dispendio computazionale, il codice non viene eseguito sempre in maniera integrale, ad esempio quando si lancia l'applicazione: *shiny* fa eseguire solo il codice che crea e modifica la pagina dell'applicazione visibile all'utente.

Tuttavia, anche così alleggerito, il costo computazionale dell'applicazione potrebbe essere molto alto, specialmente per pagine che contengono molti input ed output. Si ponga per esempio una pagina in cui vi sia un insieme I_1 di n_1 input che serve per calcolare un certo valore x_1 tramite la funzione g_1 e un altro insieme I_2 di n_2 input che serve per calcolare un certo valore x_2 tramite la funzione g_2. Si supponga poi che l'output da mostrare sia il risultato di una certa funzione $f(x_1,x_2)$: allora, qualora l'utente decida di modificare uno degli, ad esempio, input di I_1, la funzione `render` farebbe sì che vengano riaggiornati e lanciati di nuovo tutti i blocchi di codice che servono per il calcolo dell'output. Eppure, per ottimizzare l'efficienza dell'algoritmo, andrebbe rieseguito solo il calcolo di x_1 e conseguentemente $f(x_1,x_2)$, fatto che non avviene perché *shiny*, ricevuto il messaggio che un valore di input è stato modificato, riesegue e ricalcola tutto il codice relativo all'output mostrato, riaggiornando dunque anche x_2, ovviamente ad un valore eguale al precedente. Per evitare questo comportamento dispendioso, esiste una funzione, `reactive`: il valore x_1 si assegna al risultato di `reactive`, che prende come argomento g_1 (che ha come argomento gli input di I_1). x_1 ora non è più una variabile, ma si comporta come una funzione (pertanto ad esempio per chiamarla la si invoca aggiungendo nello script delle parentesi tonde dopo il nome). Lo stesso si può fare con x_2. A questo punto, se l'utente cambia il valore di un input, per esempio appartenente a I_1, allora *shiny* ricalcola soltanto x_1, perché riconosce che è stato modificato soltanto uno degli input che era usato per il calcolo di x_1. Riassumendo, `reactive` isola pezzi di codice nello script R, e li fa eseguire soltanto se uno degli input presenti all'interno viene modificato dall'utente.

Capitolo 2

L'applicazione

Finora sono stati esposti gli obiettivi del laboratorio computazionale e sono stati illustrati alcuni concetti fondamentali che ne spiegano il funzionamento. Ora è dunque è giunto il momento di passare alla descrizione del funzionamento dell'applicazione, seguendo in parallelo i risultati teorici che sono la motivazione per gli argomenti e le pagine presenti. Per la teoria si è seguito soprattutto il libro "Serie storiche economiche" (Di Fonzo e Lisi 2005).

2.1 Uno sguardo generale

Se si osserva il menù che compare a sinistra della prima schermata quando viene lanciata l'app (Figura 2.1), si può notare la presenza di cinque macroaree. In realtà deve essere sottolineata una divisione ancora più generale tra la prima macroarea, introduttiva e riguardante i processi stocastici, e le restanti aree, che riguardano più prettamente i modelli SARIMA. Il percorso attraverso le pagine è infatti organizzato secondo una spinta "ascensionale": dalla base generale delle variabili aleatorie e dei processi stocastici, quest'ultimi affrontati anche nell'ottica della stazionarietà, si passa a parlare dei processi AR, MA, ARMA e ARIMA all'interno della seconda macroarea, usando soprattutto lo strumento dell'ACF. Nel terzo blocco, il più rilevante in termini sia di ampiezza dell'esposizione che di approfondimento, vengono illustrati passaggi legati alla procedura di Box e Jenkins (Box et al. 2015), specialmente per quanto riguarda l'analisi dei residui, mentre il quarto blocco è completamente dedicato alle previsioni dei modelli ARIMA. Da ultima, la quinta macroarea introduce, in maniera simile, solo più generalizzata rispetto a

Figura 2.1: Menù generale dell'applicazione.

quanto fatto in precedenza, le serie ARIMA con ritardi stagionali.

Dal punto di vista dello script, esso è organizzato all'interno di una cartella in un file centrale, chiamato, seguendo la tradizione, `app.R`, in cui sono contenuti il codice relativo al titolo, al menù, al server e ai collegamenti con gli altri file, che invece ospitano la gestione degli input e degli spazi per gli output (*shiny* infatti, per snellire la complessità derivante dall'avere un unico file `app.R` permette di richiamare da altri script i blocchi che si riferiscono alla parte ui tramite la funzione `source`), e appunto negli altri file, uno per ognuna delle quattordici schede appartenenti all'applicazione. All'interno della stessa cartella è anche presente il file del pacchetto *sse_v4* di cui si è già parlato.

2.1.1 L'avvio dell'app

Un aspetto, forse banale, ma comunque imprescindibile, dal punto di vista dell'utente è quello dell'avvio dell'app. Da RStudio, il principale IDE di R, sono possibili tre modalità di lancio:

- la modalità "run in Window", che al lancio dell'app apre una nuova schermata sul desktop contenente l'app, perciò all'esterno della finestra di RStudio.
- la modalità "run in Viewer Pane", che fa comparire la schermata dell'app all'interno della scheda Viewer di RStudio.
- la modalità "run External", che apre una schermata sul browser.

In tutti questi casi ovviamente l'applicazione è totalmente navigabile e interattiva. Riguardo all'avvio su RStudio dell'app, esso avviene tramite il pulsante Run App a destra nella barra

orizzontale alta, che si trova al posto del consueto pulsante Run usato in genere per eseguire codice R riga per riga. Il pulsante Run App è presente solo sulla scheda del file `app.R`, contenente anche la chiamata alla libreria *shiny*.

2.2 I processi stocastici

Il primo "capitolo" dell'applicazione riguarda i processi stocastici. Prima di addentrarsi nella descrizione di come viene sviluppato e analizzato questo argomento è necessario definire formalmente un processo stocastico, anche per comprendere la notazione usata in seguito.

Definizione 2.2.1 (Processo stocastico)**.** Sia (Ω, F, P) uno spazio di probabilità, con Ω spazio degli eventi elementari, F σ-algebra su Ω e P una misura di probabilità. Se T è uno spazio parametrico discreto, allora un processo stocastico è una funzione finita Y a valori reali di $\omega \in \Omega$ e $t \in T$, t.c. per ogni t, $Y_t(\omega)$ è una funzione misurabile di ω.

Si può vedere T come un insieme di indici che creano un ordine nella successione delle $Y_t(\omega)$ e gli $\omega \in \Omega$ come eventi elementari che specificano quale risultato si ottiene per t fissato. Infatti $\forall \omega = \omega_0$ fissato, $Y_t(\omega_0)$ è una funzione di t detta traiettoria o realizzazione del processo stocastico (è la serie storica che si può osservare), mentre appunto $\forall t = t_0$ fissato, $Y_{t_0}(\omega)$ è una funzione misurabile di $\omega \in \Omega$, cioè una variabile aleatoria. Pertanto per $\omega = \omega_0$ fissato e $t = t_0$ fissato, $Y_{t_0}(\omega_0)$ è un numero reale. Si possono inoltre definire i seguenti momenti:

- Media:
  $\mu_t = E[Y_t]$.
- Varianza:
  $\sigma_t^2 = Var[Y_t] = E[Y_t - \mu_t]^2$.
- Autocovarianza:
  $\gamma_{t_1,t_2} = E\{[Y_{t_1} - \mu_{t_1}][Y_{t_2} - \mu_{t_2}]\}$.
- Autocorrelazione:
  $\rho_{t_1,t_2} = \dfrac{\gamma_{t_1,t_2}}{\sigma_{t_1}\sigma_{t_2}}$.

È utile anche ricordare la definizione di stazionarietà e alcune proprietà connesse.

Definizione 2.2.2 (Stazionarietà in senso forte)**.** Un processo stocastico si dice stazionario in senso forte o stretto se le distribuzioni congiunte di $(Y_{t_1},Y_{t_2},...,Y_{t_n})$ e di $(Y_{t_1+\tau},Y_{t_2+\tau},...,Y_{t_n+\tau})$ $\forall t_i$ e $\forall \tau$ sono uguali.

Un processo fortemente stazionario ha media $\mu_t = \mu$ e $\sigma_t^2 = \sigma^2$ costanti $\forall t$. Per quanto concerne l'autocovarianza, si ha $\gamma_{t_1,t_2} = \gamma_{|t_2-t_1|}$ $\forall t_1, t_2$. Sia ora $k = t_2 - t_1$. Allora $\gamma_{t_1,t_2} = \gamma_k = \gamma_{-k}$. Passando alle autocorrelazioni, $\rho_{t_1,t_2} = \rho_k = \dfrac{\gamma_k}{\gamma_0} = \rho_{-k}$.

Definizione 2.2.3 (Stazionarietà in senso debole)**.** Un processo stocastico si dice stazionario in senso debole se:

- $E[Y_t] = \mu, \forall t.$
- $Cov[Y_t,Y_{t+k}] = \gamma_k, \forall t, \forall k.$

L'ipotesi di stazionarietà debole è quella più usata perché più flessibile. Se le Y_{t_i} siano variabili aleatorie normali, stazionarietà in senso debole e in senso forte coincidono.
Per una trattazione più esauriente si rimanda a Di Fonzo e Lisi 2005, pag. 160 e seguenti, da cui pure sono state tratte queste definizioni.

2.2.1 Una variabile casuale

L'esplorazione dell'app incomincia da un concetto che dovrebbe essere già piuttosto familiare agli utenti: quello di variabile aleatoria. L'aspetto della pagina è mostrato nella Figura 2.2. Il grafico rappresenta la curva di densità di una variabile aleatoria (l'utente può scegliere tra tre distribuzioni continue: normale, uniforme ed esponenziale). L'integrazione tra quanto già conosciuto e il concetto di processo stocastico avviene attraverso la notazione che compare sul grafico: l'asse delle ascisse corrisponde all'insieme Ω degli eventi elementari, mentre la curva di densità rappresenta la corrispettiva immagine di Ω tramite Y. Il grafico deve essere dunque interpretato come una rappresentazione di un processo stocastico, in cui lo spazio parametrico T sia costituito da un solo elemento ed è da identificare con una variabile aleatoria. Le realizzazioni sono il valore della densità di tale variabile negli $\omega_i \in \Omega$. Per alcune difficoltà a rappresentare nelle didascalie in R le lettere greche in presenza di un grande numero di ω_i, si è scelto di scrivere il generico elemento di Ω come `w_i`, e la corrispondente realizzazione come `Y(w_i)`.

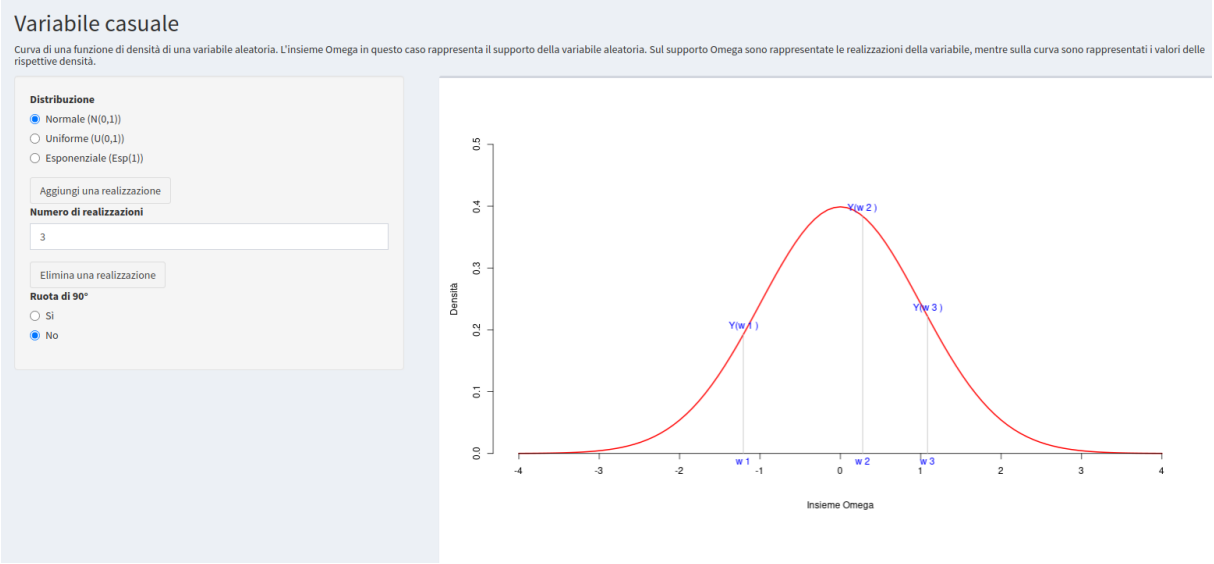


Figura 2.2: Prima pagina dell'applicazione: variabile casuale.

È anche possibile capovolgere il grafico di 90°, per rendere il passaggio dalla rappresentazione grafica di una variabile aleatoria alla rappresentazione di un processo stocastico quale si vedrà nella seguente sezione.

2.2.2 Primo esempio di processi stocastici

La seconda pagina dell'app si conforma a quella che è l'immagine "classica" di (una parte di) un processo stocastico: una successione di variabili aleatorie, in generale con distribuzione non identica, le cui realizzazioni ordinate corrispondono ai grafici delle serie storiche cui si è abituati. L'applicazione permette di scegliere il numero di variabili aleatorie successive da prendere in considerazione (fino a un massimo di cinque), indicizzate da generici $t+1$,...,$t+5$, e inoltre per ogni variabile permette di modificare la distribuzione, purché gran parte delle realizzazioni ricadano nell'intervallo $(-4, 4)$, per ragioni leggibilità. L'utente può inoltre generare fino a cento traiettorie. Fino a due traiettorie generate è presente per ogni punto delle serie la notazione teorica. La Figura 2.3 mostra l'aspetto della schermata con cinque variabili diversamente distribuite e due traiettorie. Dal punto di vista algoritmico, ogni traiettoria è ottenuta settando un diverso seme (che rappresenta un certo ω_i) per generare pseudocasualmente le realizzazioni, che sono una per ognuna delle variabili aleatorie in successione.

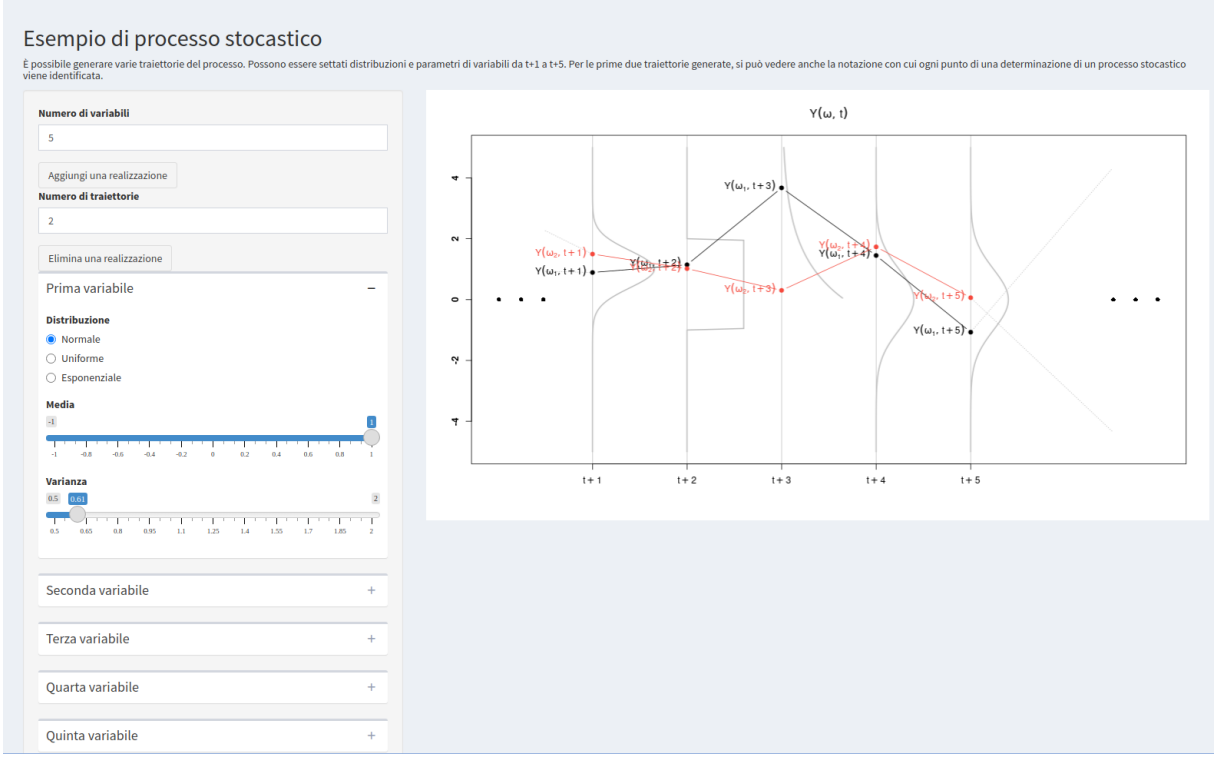


Figura 2.3: Seconda pagina dell'applicazione: un primo esempio di processo stocastico.

2.2.3 Secondo esempio di processi stocastici

Per far comprendere pienamente quali siano le possibilità di un processo stocastico e come una certa distribuzione delle variabili aleatorie che ne fanno parte influenzi i valori delle traiettorie, la terza pagina offre, grazie al pacchetto *dygraphs*, un grafico in cui si possono gestire le distribuzioni di al massimo sei variabili aleatorie in successione, le cui realizzazioni possono andare in un intervallo più ampio rispetto alla pagina precedente. È inoltre permesso generare un massimo di mille traiettorie.

2.2.4 Processi stazionari

Il concetto di processo debolmente o fortemente stazionario, e le differenze tra i due tipi di stazionarietà sono forse piuttosto difficili da cogliere intuitivamente. È per questa ragione che nella quarta pagina si è deciso di rappresentare la stazionarietà in una maniera simbolica, più efficace dal punto di vista esemplificativo. In Figura 2.4 (a) viene disegnato, come una successione di densità di variabili aleatorie normali identicamente distribuite un "pezzo" di un processo stocastico debolmente stazionario (si ricordi che stazionarietà in senso debole e in senso forte

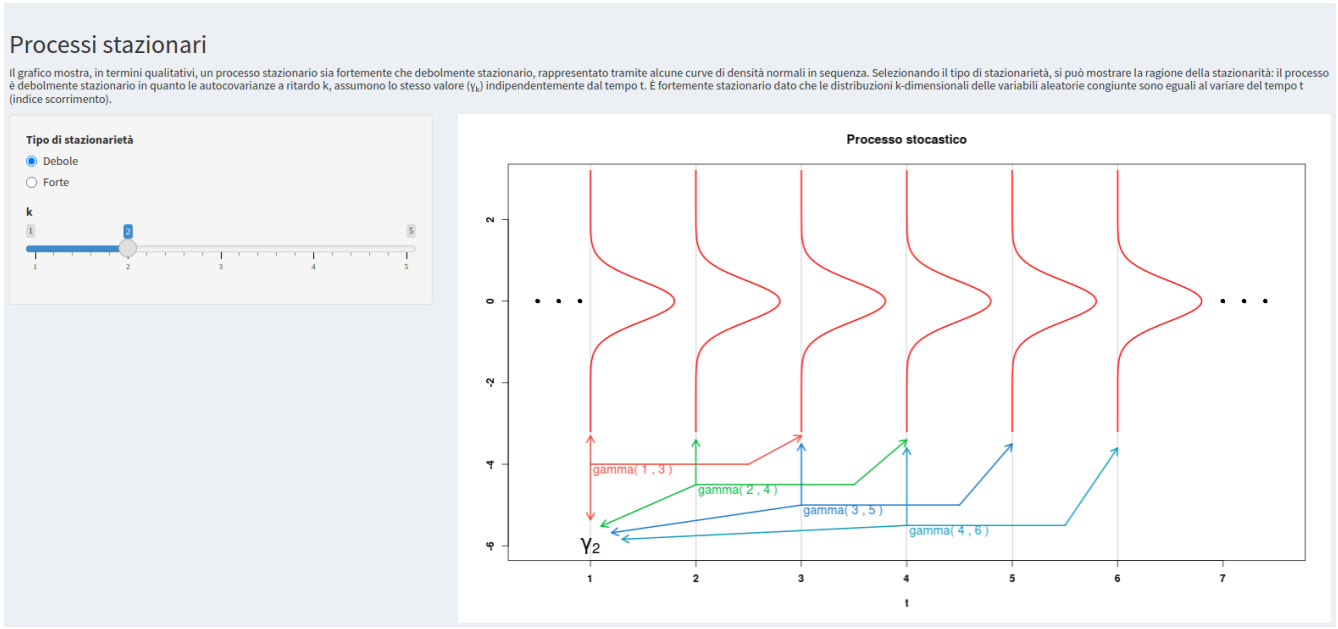
coincidono per processi a variabili casuali normali). Le didascalie sotto alle frecce di colore uguale indicano le autocovarianze tra variabili a distanza k fissato (l'utente può scegliere k in un range da 1 a 5). Tutte le frecce puntano poi su un certo valore γ_k, ad indicare che tutte le autocovarianze hanno eguale valore. Quanto all'eguale varianza e media, dovrebbero essere percepibili dal fatto che tutte le curve sono uguali, e tutte sono centrate sul valore 0.
La stazionarietà in senso forte è gestita come in Figura 2.4 (b). Sullo sfondo della successione di variabili i.i.d. normali si può scegliere il ritardo k, che include all'interno di un rettangolo verde $k+1$ variabili consecutive. Grazie ad un altro indice presente sotto al grafico, il rettangolo può scorrere dalle prime $k+1$ alle ultime $k+1$ variabili . Come la didascalia spiega, il rettangolo indica la distribuzione congiunta delle variabili al suo interno. Il fatto che questa "cornice" inquadri sempre un'immagine uguale, indica uguaglianza nella distribuzione congiunta.

2.2.5 Confronto tra processi stazionari e non stazionari

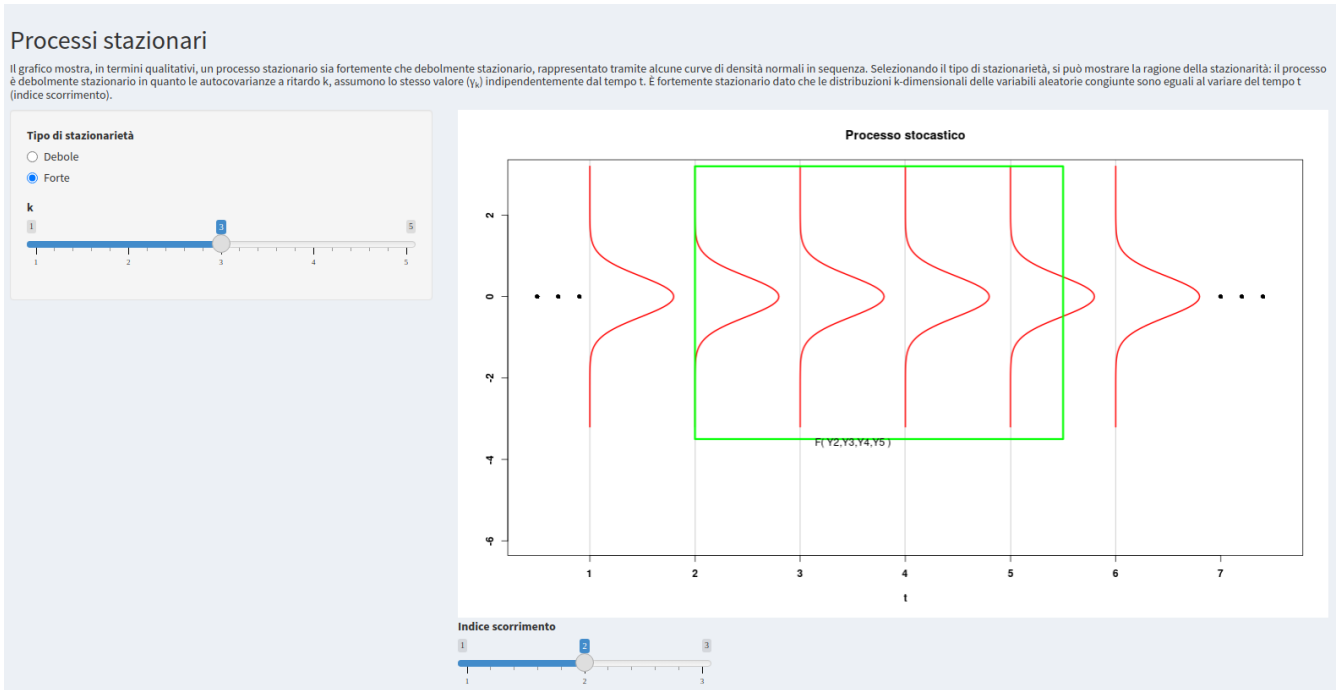
Conclude la sezione relativa ai processi stocastici un confronto tra un processo (debolmente e fortemente) stazionario e uno non stazionario. Il primo processo viene chiamato white noise, ed è semplicemente la successione di una serie di variabili aleatorie i.i.d. (nel caso dell'applicazione delle variabili normali standard). Il secondo è un processo random walk. Per ogni Y_t appartenente alla successione di variabili aleatorie di un processo random walk,

$$Y_t = \begin{cases} \mu & \text{se } t = 0 \\ Y_{t-1} + \varepsilon_t & \text{se } t \geq 1 \end{cases}$$

con ε_t variabile aleatoria di media 0 e varianza σ_ε^2 (in questo caso la variabile è una normale standard). Un processo random walk non è stazionario perché la sua varianza è $\sigma_t^2 = t\sigma_\varepsilon^2$, non costante. L'applicazione (Figura 2.5) mostra nella prima riga una serie storica generata da un white noise. Nel primo grafico compare la serie originaria (in grigio), la media campionaria $\bar{y}_t$ fino al tempo t (in blu) e la media teorica $E[Y_t]$, che è sempre uguale a 0 (in rosso). Il grafico affianco invece mostra (in verde) la varianza campionaria S_t^2 fino al tempo t e la varianza teorica $Var[Y_t]$, uguale a 1 (in rosso). L'andamento asintotico della media e della varianza campionarie tende a sovrapporsi alle linee dei rispettivi momenti teorici.



(a) *Stazionarietà in senso debole*



(b) *Stazionarietà in senso forte*

Figura 2.4: Quarta pagina dell'applicazione: la stazionarietà.

Figura 2.5: Quinta pagina dell'applicazione: confronto tra processi stazionari e non stazionari.

La serie generata da un processo random walk viene analizzata nella seconda riga di grafici: nel primo grafico vi è la serie originale, la media campionaria fino al tempo t e la media teorica. Nel secondo grafico vi è varianza campionaria fino al tempo t e la corrispondente varianza teorica (negli stessi colori dei due grafici precedenti). Dai plot è evidente l'andamento asintotico non stazionario, sia in media che in varianza: infatti per un processo random walk $E[Y_t \mid Y_{t-1}] = Y_{t-1}$, pertanto il valore atteso condizionato a $Y_{t-1} = y_{t-1}$ è in generale diverso $\forall t$. Quanto alla varianza campionaria, pur non seguendo pedissequamente la linea rossa della varianza teorica, non si stabilizza intorno a un valore fissato, come invece succedeva nel processo white noise.

L'utente può scegliere tra diverse lunghezze delle serie generate. Per quanto riguarda l'aspetto computazionale, anche questi grafici sono stati ottenuti con *dygraphs* per una migliore esperienza di visualizzazione e interazione.

2.3 Processi ARIMA

La seconda sezione dell'app riguarda una prima introduzione ai processi ARIMA, soprattutto tramite lo studio dell'ACF. Come in precedenza, è bene introdurre teoricamente questi argomenti.

Definizione 2.3.1 (Processo a media mobile MA(q))**.** Sia $\{\varepsilon_t\}$ una successione di variabili alea-

torie i.i.d. di media 0 e varianza σ_ε^2. Un processo stocastico si dice a media mobile di ordine q se

$$Y_t = \varepsilon_t - \theta_1 \varepsilon_{t-1} - ... - \theta_q \varepsilon_{t-q}$$

con $\theta_1, ..., \theta_q$ costanti.

Se si definisce un operatore B, t.c. $B^k Y_t = Y_{t-k}$, si può vedere Y_t come $\theta(B)\varepsilon_t$, con $\theta(B) = (1 - \theta_1 B - ... - \theta_q B^q)$. Eguagliato a zero, $\theta(B)$ viene chiamata equazione caratteristica del processo. La media del processo a media mobile è uguale a zero, mentre l'autocovarianza ha una formula più complicata, ma comunque non dipendente dal tempo. Un processo MA(q) è dunque sempre stazionario. Per una trattazione più esauriente, si veda Di Fonzo e Lisi 2005, pag. 171 e seguenti.

Definizione 2.3.2 (Processo autoregressivo AR(p))**.** Sia $\{\varepsilon_t\}$ una successione di variabili aleatorie i.i.d. di media 0 e varianza σ_ε^2. Un processo stocastico si dice autoregressivo di ordine p se

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + ... + \phi_p Y_{t-p} + \varepsilon_t$$

con $\phi_0, ..., \phi_p$ costanti.

Con $\phi(B) = (1 - \phi_1 B - ... - \phi_p B^p)$ eguagliato a zero, si indica l'equazione caratteristica del processo. La media del processo autoregressivo è uguale a $\dfrac{\phi_0}{1 - \phi_1 - ... - \phi_p}$, mentre l'autocovarianza ha una formula più complicata. Un processo AR(p) è stazionario solo se le radici dell'equazione caratteristica sono tutte in modulo maggiori di 1. Per una trattazione più esauriente, si veda Di Fonzo e Lisi 2005, pag. 174 e seguenti.
Si può inoltre introdurre un modello misto ARMA:

Definizione 2.3.3 (Processo autoregressivo a media mobile ARMA(p,q))**.** Sia $\{\varepsilon_t\}$ una successione di variabili aleatorie i.i.d. di media 0 e varianza σ_ε^2. Un processo stocastico si dice autoregressivo a media mobile di ordine p e q se

$$Y_t - \sum_{i=1}^{p} \phi_i Y_{t-i} = \phi_0 + \varepsilon_t - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

.

Per una trattazione più esauriente, si veda Di Fonzo e Lisi 2005, pag. 178 e seguenti.
È infine da includere una tipologia di processi detti non stazionari omogenei di grado d. Introdotto l'operatore differenza d-esima $\Delta^d = (1 - B)^d$, un processo non stazionario a variabili aleatorie Y_t si dice non stazionario omogeneo di grado d se $\Delta^d Y_t = X_t$ è un processo stazionario. L'operatore differenza dunque stabilizza il processo. Guardando al caso ARMA, se X_t è un processo ARMA(p,q), allora Y_t si dice processo autoregressivo integrato a media mobile ARIMA (p,d,q).
Per tutti questi tipi di processi la funzione delle autocorrelazione viene chiamata ACF, e il suo grafico ai primi k ritardi correlogramma, mentre per PACF si intende l'autocorrelazione tra due generiche Y_t e Y_{t+k} condizionata ai valori di Y_{t+1},...,Y_{t+k-1}.

2.3.1 ACF e PACF teoriche dei processi ARMA

Nella prima schermata della seconda sezione vengono illustrati i cambiamenti nell'ACF e PACF teoriche al mutare dei parametri. L'utente può modificare il valore dei parametri θ_i riferiti ai processi a media mobile (ordine massimo 3), e il valore dei parametri ϕ_i inerenti i processi autoregressivi (ordine massimo 2; è presente anche un parametro additivo ϕ_0, che essendo però solo di intercetta non influenza in alcun modo le autocorrelazioni: è stato inserito soltanto perché l'utente possa verificare questo fatto), per studiare il comportamento dei correlogrammi corrispondenti. Per un processo autoregressivo di ordine p, l'ACF decresce a zero con un andamento misto tra esponenziale e pseudoperiodico, mentre la PACF è diversa da zero solo fino al ritardo p. Il comportamento di un processo MA(q) è esattamente al contrario: l'ACF è diversa da zero solo ai primi q ritardi, mentre la PACF decresce con comportamento esponenziale-pseudoperiodico. Un processo ARMA(p,q) assume un andamento misto, non ben definibile e variabile in base ai valori dei parametri ϕ_i e θ_i. Nella Figura 2.6 è esposto l'aspetto della pagina per un processo AR(1) ($\phi_1 = 0.6$).

2.3.2 ACF e PACF empiriche per processi ARIMA

Lo scopo della seconda pagina della seconda sezione è offrire uno strumento di comparazione tra i valori teorici visti sopra e i valori empirici. Rispetto a quanto fatto in precedenza, qui l'utente può scegliere il numero e il valore di parametri da un processo ARIMA (ordini massimi: 2 per

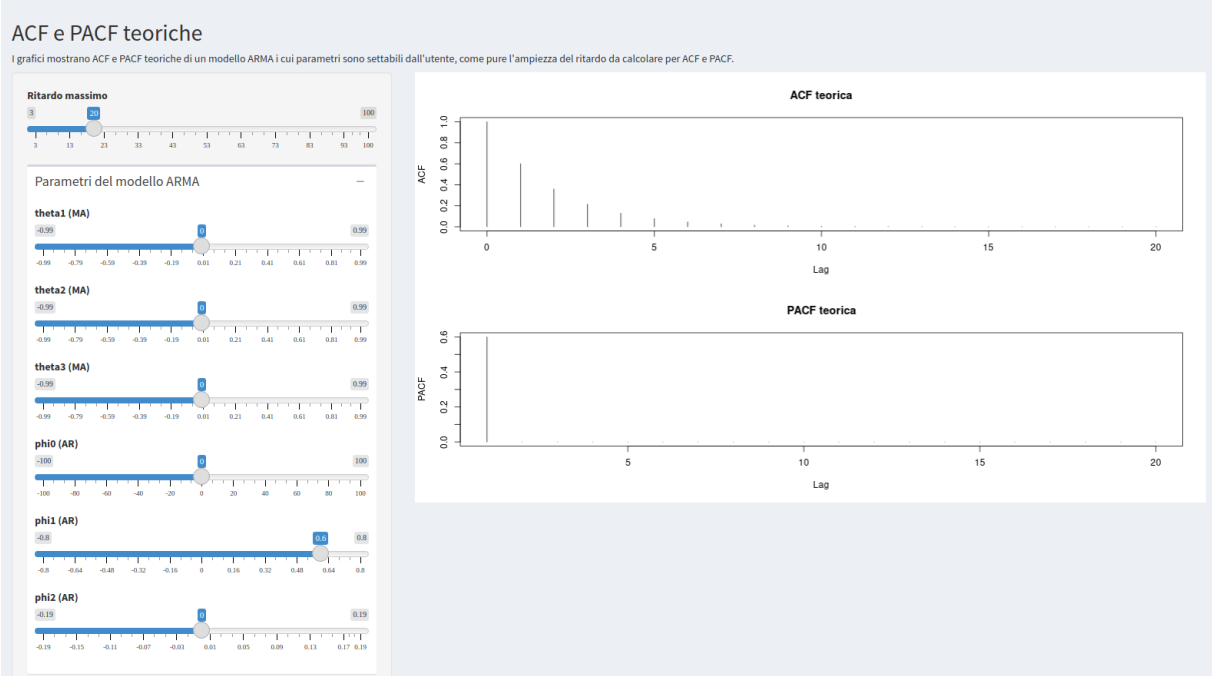


Figura 2.6: Sesta pagina dell'applicazione: ACF e PACF teoriche.

la parte autoregressiva, 3 per la parte a media mobile, 1 per le differenze). La serie generata dal modello scelto (Figura 2.7) viene visualizzata nel grafico in alto (anche la lunghezza della serie può essere decisa da chi interagisce con l'app), mentre nei due grafici sottostanti sono espresse le ACF e PACF empiriche calcolate sulla serie. Si nota la presenza di due linee parallele tratteggiate in blu: esse rappresentano i limiti degli intervalli di confidenza al 95% per il test statistico che testa l'ipotesi che le autocorrelazioni della serie a ritardo k siano uguali a zero (serie white noise, a valori incorrelati). Le linee hanno ordinata $\pm 1.96 \frac{1}{\sqrt{n}}$, dove n è la lunghezza della serie, perché viene supposta la normalità della serie white noise. Pertanto all'aumentare della lunghezza della serie anche l'ampiezza dell'area all'interno delle linee tratteggiate diminuisce. La principale differenza dunque tra ACF teoriche ed empiriche è che le autocorrelazioni hanno un comportamento meno netto nel caso empirico: ad esempio, nella particolare configurazione in Figura 2.7, la PACF del processo, che dovrebbe essere un AR(1) (si veda la parte di pulsantiera sulla sinistra), non si annulla del tutto dal ritardo due in avanti, pur rimanendo all'interno delle bande di confidenza. È per questo che in generale, davanti a casi con dati reali, bisogna prestare maggior attenzione nell'interpretazione dei modelli.

Figura 2.7: Settima pagina dell'applicazione: ACF e PACF empiriche.

2.4 Procedura di Box e Jenkins

Si entra ora nel cuore dell'applicazione e dell'analisi delle serie storiche: la cosiddetta procedura di Box e Jenkins (Box et al. 2015), utile per costruire modelli partendo da una serie con dati reali o simulati. La procedura si divide in tre fasi:

- identificazione del modello, ovvero, partendo dall'ACF e PACF empiriche, l'obiettivo è determinare gli ordini della struttura sottostante alla serie, utilizzando anche criteri come l'AIC per evitare modelli sovraparametrizzati. L'AIC è uguale in questo caso a $\frac{2}{n}(\log L(\hat{\delta}) - k)$, dove n è la lunghezza della serie, $L(\hat{\delta})$ rappresenta la verosimiglianza massimizzata (δ è il vettore dei parametri stimati con il metodo della massima verosimiglianza) e k è il numero di parametri stimati.

- la stima dei parametri del modello. L'applicazione non affronta direttamente questo argomento, dato che la stima viene effettuata attraverso algoritmi dalle formule difficilmente rappresentabili con strumenti interattivi. Inoltre, dato che i moderni calcolatori sono molto più veloci di qualsiasi utente umano nello svolgere questi calcoli, non vi è neanche un interesse diretto nello studiare approfonditamente il loro funzionamento. Basti semplicemente dire che esistono due metodi di stima più popolari, il metodo dei minimi

quadrati non lineari e quello della massima verosimiglianza condizionata, quest'ultimo implementato dalla funzione `Arima` del pacchetto *forecast*.

- il controllo diagnostico, che deve controllare che il modello identificato e stimato sia adeguato. Si basa sull'analisi dei residui del modello stimato. Si utilizzano ACF e PACF empiriche dei residui e alcuni test, come quello di Ljung e Box (Ljung e Box 1978, che prende in considerazione autocorrelazioni successive fino al ritardo m, calcolando una statistica che nell'ipotesi di incorrelazione dei residui si distribuisce come un χ^2_{m-p-q} (p e q rappresentano gli ordini AR e MA del modello stimato). Un altro test impiegato è quello di Jarque e Bera (Jarque e Bera 1980), che testa l'ipotesi di normalità dei residui, utilizzando la stima campionaria del coefficiente di asimmetria S e la stima campionaria del coefficiente di curtosi K. La statistica derivata dovrebbe distribuirsi, nell'ipotesi di normalità dei residui, come un χ^2_2.

Un raffronto più dettagliato si trova a pag. 195 e seguenti di Di Fonzo e Lisi 2005.

2.4.1 Trasformazioni per la stazionarietà

Prima di procedere all'identificazione di un certo modello ARMA, è sempre necessario sincerarsi che esso sia stazionario in media e in varianza. Se la serie non risultasse stazionaria, si possono principalmente fare due cose:

- se la serie non risulta stazionaria in media, è utile una o più differenziazioni, finché la media non si stabilizza.

- se la serie è non stazionaria in varianza, allora si può stabilizzarla attraverso la funzione $T(Y_t)$ di Box-Cox, così definita:

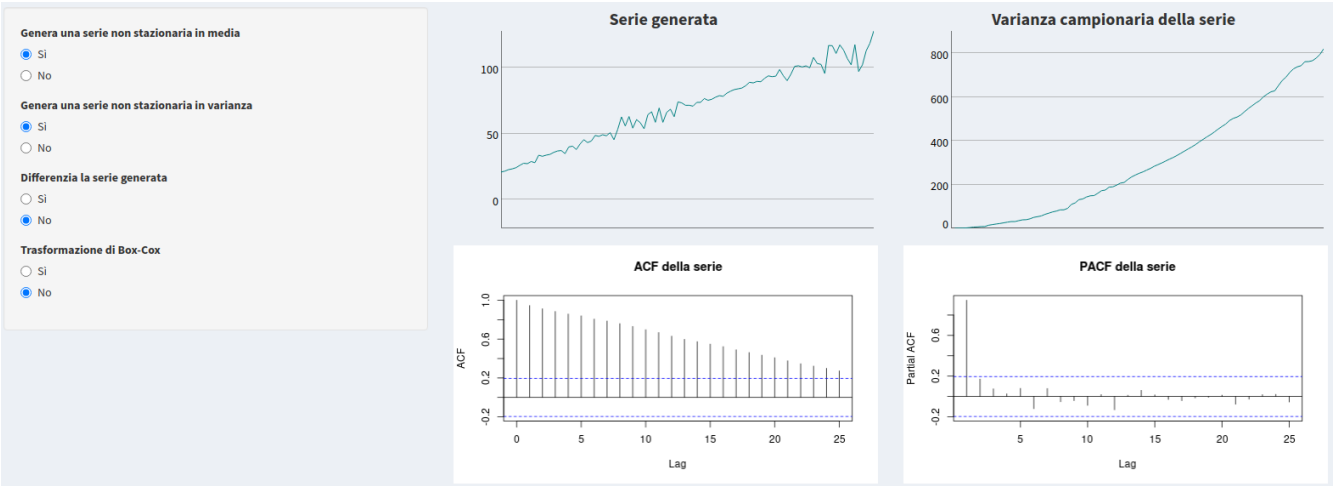
$$T(Y_t) = \begin{cases} \dfrac{Y_t^\lambda - 1}{\lambda} & \text{se } \lambda \neq 0 \\ \log Y_t & \text{se } \lambda = 0 \end{cases}$$

La prima pagina, introduttiva, della sezione riguardante la procedura di Box e Jenkins mostra proprio quali sono gli effetti di queste procedure sulla serie originaria. L'utente in questo caso

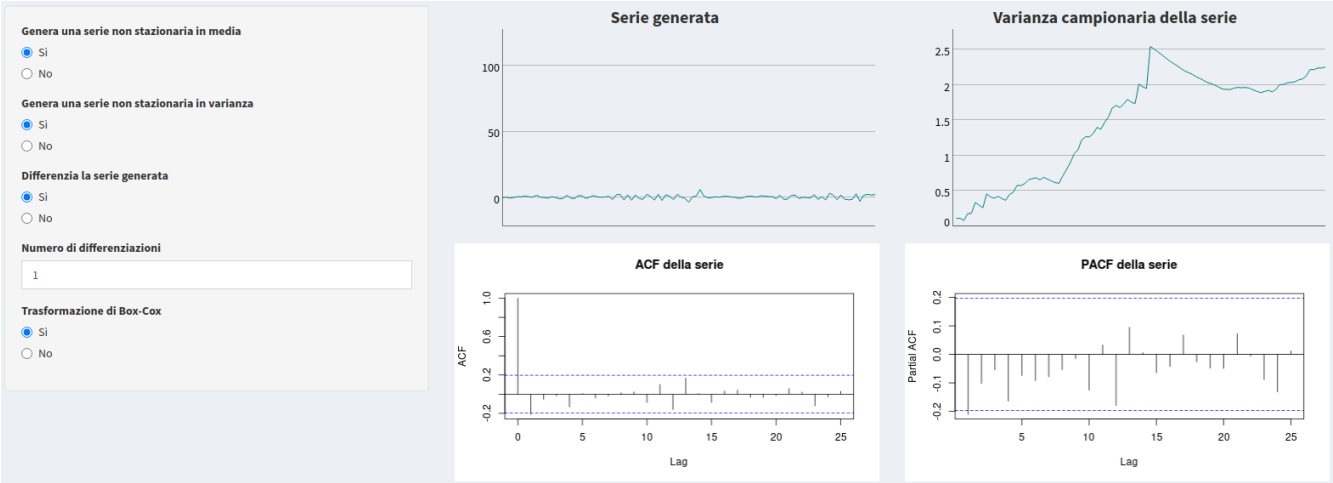
non può scegliere i parametri da cui generare la serie, ma soltanto di generare una serie white noise, una non stazionaria in media (computazionalmente, a un white noise viene sommato un trend deterministico), o una non stazionaria in varianza (ottenuta assemblando quattro frammenti di serie generate da white noise con varianze differenti e un ultimo frammento in cui ogni punto ha varianza maggiore del precedente). Una volta scelto il tipo di serie da generare, si possono applicare le trasformazioni. In Figura 2.8 (a) si vede una serie non stazionaria in media e in varianza. Il grafico della serie mostra un forte trend lineare ascendente. Tuttavia è ben evidente anche la diversa varianza della serie nei vari tratti, specialmente confrontando la parte iniziale e quella finale della serie. Affianco anche la serie della varianza campionaria al tempo t sembra avere un trend quasi esponenziale, anche se probabilmente parte dell'aumento della varianza è anche dovuto alla presenza di non stazionarietà in media. L'ACF empirica mostra il tipico andamento lentamente discendente, uno degli indizi più sicuri di non stazionarietà in media. La PACF mostra solo la prima autocorrelazione fuori dalle bande di confidenza, indicando un comportamento autoregressivo, da identificare sempre con il trend lineare positivo della serie.
La Figura 2.8 (b) mostra invece l'andamento della serie dopo una differenziazione e la trasformazione di Box e Cox. La serie differenziata si comporta praticamente come un white noise. La varianza in realtà non è del tutto stabilizzata: nel tratto centrale essa aumenta di molto, tuttavia è molto più contenuta rispetto al valore di circa 800 che assumeva alla fine della serie nel caso non trasformato. L'ACF e la PACF confermano il comportamento white noise di questa serie: solo il ritardo 1 della PACF fuoriesce di poco dalle bande di confidenza.

2.4.2 Identificazione del modello

Nella seconda pagina della sezione sulla procedura di Box e Jenkins si passa alla vera e propria fase dell'identificazione del processo. L'utente può generare una serie da un modello ARMA, scegliendo il valore dei parametri necessari. Una volta generata la serie, può utilizzarla per cercare di ricavare gli ordini del processo originario, ad esempio specificando un modello con ordini differenti rispetto a quello di partenza. A volte infatti succede che una serie, realizzazione di un certo processo, si adatti bene anche ad essere interpretata secondo un modello malspecificato. Nella Figura 2.9 è illustrato un caso del genere: nella prima riga di grafici sono visibili ACF e PACF empiriche della serie generata dal modello scelto dall'utente. Nella seconda riga



(a) *La serie originaria senza trasformazioni.*



(b) *La serie trasformata.*

Figura 2.8: Ottava pagina dell'applicazione: trasformazioni per la stazionarietà.

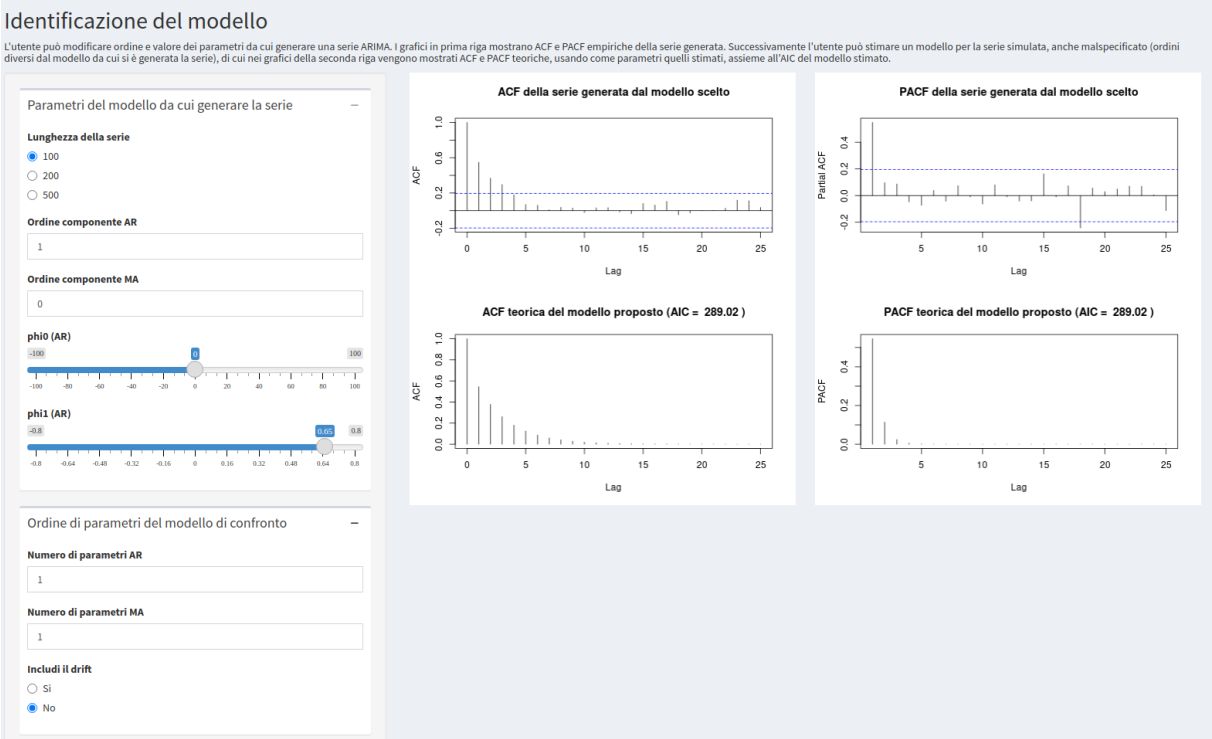


Figura 2.9: Nona pagina dell'applicazione: Identificazione del processo.

vengono mostrate le ACF e PACF teoriche di un modello che ha ordine quello proposto dall'utente per specificare la serie precedentemente generata. Il valore dei parametri è ottenuto tramite la stima secondo il modello proposto in base alla serie originaria. In linea di principio, se il modello proposto avesse ordine diverso da quello di partenza, le ACF e PACF del primo dovrebbero comportarsi in maniera dissimile rispetto alle ACF e PACF del secondo. Tuttavia nel mondo dei dati reali, questi aspetti non sono così netti, e dunque si ha, come ad esempio in Figura 2.9, che l'ACF empirica di una serie da un processo AR(1) sia quasi sovrapponibile con l'ACF teorica di un modello ARMA(1,1), i cui parametri sono stimati a partire dai dati della serie di partenza.

2.4.3 Stima dei parametri

Come si sottolineava sopra, la parte vera e propria di stima dei parametri del modello non è stata inserita nell'applicazione, per ragioni di scarsa rappresentabilità e utilità. È tuttavia presente al suo posto una specie di "esperimento montecarlo": la stima dei parametri di un certo processo tramite la distribuzione dei parametri stimati usando $N \geq 1$ serie generate da tale processo. Il

procedimento è semplice: l'utente sceglie gli ordini di un processo e il valore dei parametri. In seguito da questo modello vengono generate N serie (anche il valore di N è a scelta dell'utente: ovviamente scegliere N grande implica accettare onerosi costi computazionali). A questo punto l'utente può specificare gli ordini di un modello per l'identificazione del modello di partenza. Le stime dei parametri secondo questo modello vengono basate sulle serie generate in precedenza: per ogni serie si stimano i parametri in base al modello adattato. Infine R crea i grafici degli istogrammi con le frequenze delle stime di ogni singolo parametro dal modello specificato.
La Figura 2.10 (a) illustra un caso di cattiva specificazione: il modello originario era un ARMA(2,1), per il quale è stato proposto un ARMA(1,2). $N = 100$. Si nota come le frequenze delle stime non abbiano in generale media e moda coincidenti, e come le frequenze siano alte su un lungo intervallo, segno di grande varianza nelle stime stesse. La Figura 2.10 (b) invece mostra lo stesso modello di partenza, ma correttamente specificato. Si nota una distribuzione più centrata sui valori dei reali parametri. Anche le medie (linee verticali in rosso) sono più precise.

2.4.4 Diagnostica e analisi dei residui

L'ultima pagina di questa sezione è dedicata principalmente all'analisi dei residui. Come di consueto, all'utente viene data la possibilità di gestire il modello da cui generare una serie. Poi può cercare di specificare un altro modello che si adatti alla serie generata. Gli output forniti sono le stime dei parametri del modello specificato, a cui si accostano, nella colonna adiacente, l'errore standard per ognuna d queste stime. A destra vi sono invece le radici delle equazioni caratteristiche inerenti il processo, separatamente per la parte AR e per la parte MA. Si ricordi che tali radici sono importanti per alcune caratteristiche del modello: radici dell'equazione caratteristica della parte autoregressiva in modulo maggiori di 1 indicano che il processo è stazionario, mentre se sono le radici della parte a media mobile ad essere maggiori di 1, allora il processo è invertibile. Un processo si dice invertibile se $\forall t$, $Y_t = h(Y_{t-1}, Y_{t-2},...) + \varepsilon_t$, dove h è una funzione lineare e ε_t è una variabile aleatoria di media 0 e varianza σ_ε^2 (la successione delle ε_t è a componenti incorrelati). L'applicazione automaticamente avverte se l'ipotesi di non stazionarietà è stata violata per il modello stimato.
Nello stesso riquadro è presente anche il valore della statistica per il test di Jarque e Bera e il

(a) *Un caso di cattiva specificazione del modello.*

(b) *La serie trasformata.*

Figura 2.10: Decima pagina dell'applicazione: Distribuzione delle stime dei parametri.

Figura 2.11: Undicesima pagina dell'applicazione: Stima dei parametri, diagnostica e residui.

corrispettivo p-value. A destra è invece la rappresentazione grafica dei p-value del test di Ljung e Box per i primi 20 ritardi. La seconda riga di grafici mostra ACF e PACF empiriche dei residui del modello stimato. Nella figura 2.11 è mostrato il caso in cui il modello è correttamente specificato. Il processo è anche stazionario, dato che tutte le radici delle equazioni caratteristiche sono in modulo maggiori di 1.

2.5 Previsioni

Stimare un modello da una serie storica a valori reali e di lunghezza finita ovviamente non è la conclusione di un'analisi statistica. Il modello da identificare deve essere in grado di dire qualcosa sull'evoluzione del processo anche nel futuro. È per questo che si parla di previsioni di una serie storica.

La teoria della previsione, soprattutto per gli aspetti più computazionali e algoritmici, non è stata trattata a fondo nell'applicazione a fondo, né verrà ripresa in questa relazione. Basti solo menzionare quello che è il previsore ottimo per le serie storiche, ovvero quella funzione g, che ha per argomento i dati della serie storica fino al tempo t e che oltre a non essere distorta per la

previsione, ovvero tale che $E[Y_{t+k} - g(I_t)] = 0$ (dove I_t rappresenta l'insieme dei dati della serie fino al tempo t), minimizzi anche l'errore quadratico medio $e^2_{t+k} = E[(Y_{t+k} - g(I_t))^2]$. Studi dimostrano che il previsore ottimo è il valore atteso condizionato $g(I_t) = \hat{Y}_{t+k} = E[Y_{t+k} \mid I_t]$. Nei processi gaussiani, g è una funzione lineare. Inoltre il comportamento asintotico del previsore ottimo per i processi ARMA stazionari è il seguente:

$$\lim_{k \to +\infty} E[Y_{t+k} \mid I_t] = E[Y_t].$$

$$\lim_{k \to +\infty} Var[e_{t+k}] = Var[Y_t].$$

Per un'esposizione più approfondita dei metodi di previsione, si rimanda a Di Fonzo e Lisi 2005, pag. 267 e seguenti.

2.5.1 La previsione nell'app

La Figura 2.12 mostra l'aspetto della schermata relativa alla previsione nell'applicazione. Come al solito, l'utente ha la facoltà di scegliere gli ordini e il valore dei parametri del processo da cui generare la serie che verrà poi analizzata. Allo stesso modo può specificare a posteriori gli ordini di ben due modelli alternativi per la serie generata. I due grafici in alto nella figura rappresentano le ACF e PACF empiriche dei residui della serie da uno dei due modelli così specificati (l'utente può scegliere se farsi mostrare l'ACF dal primo o dal secondo modello). La corretta specificazione del modello è infatti condizione imprescindibile per un calcolo consistente dei valori di predizione, assieme a stime accurate dei parametri originari. Nell'esempio mostrato pertanto si è scelto di generare da un modello ARIMA(1,1,1) e di specificarlo poi correttamente a posteriori nel primo dei due modelli alternativi, mentre l'altro rimane un modello white noise. La serie generata è costituita da 500 punti. Tuttavia l'utente può scegliere di usarne solo $n \leq 500$ per stimare i parametri. Può inoltre scegliere l'orizzonte predittivo, ovvero il numero di passi k da prevedere partendo dall'ultimo dato a disposizione. Nell'esempio, $n = 250$, e l'orizzonte predittivo ha la stessa lunghezza: in pratica metà della serie viene utilizzata per prevedere l'andamento dell'altra metà. Il grafico in basso a destra in Figura 2.12 proietta in arancio la serie originaria, lunga 500 passi. La linea verticale tratteggiata in nero segnala l'inizio della

previsione. Da essa in effetti dipartono tre curve: le due curve tratteggiate in blu rappresentano gli estremi degli intervalli al 95% per la previsione al tempo $t + k$, mentre la curva tratteggiata rossa è la previsione puntuale. A differenza del modello ARMA stazionario, gli estremi degli intervalli, che sono una funzione della varianza della previsione, asintoticamente non convergono, mentre la previsione puntuale non si stabilizza sulla media della serie usata per la stima (linea orizzontale tratteggiata in nero), ma sul livello del valore della serie a $t = 250$, ovvero per l'ultimo dato della serie disponibile. Come ultimo appunto da fare sul grafico, in realtà il livello di confidenza reale mostrato è minore del 95%: infatti per parametri stimati e non esatti la previsione è più debole. Tuttavia in questo caso comunque la serie originale in arancio per $t = 251, 252,...,500$ è totalmente contenuta all'interno degli estremi degli intervalli.
Rimane da descrivere solo la piccola tabella che contiene i dati del test di Diebold e Mariano. Questo test confronta le funzioni di perdita, cioè quella funzione che quantifica l'entità dell'errore di predizione e_{t+k} (essa è uguale a 0 se $e_{t+k} = 0$ ed è maggiore di zero altrimenti) associate ai due modelli alternativi per la serie originaria. L'ipotesi alternativa del test è che la funzione di perdita associata al primo modello stimato sia minore rispetto a quella del secondo. La statistica test finale si distribuisce come una normale standard. Nel caso in esame il valore della statistica è minore di -19, pertanto l'ipotesi nulla di eguaglianza delle funzioni di perdita associata viene rifiutata, evidenziando che il primo modello, come ci si aspettava, fa delle previsioni più efficienti rispetto al secondo.

2.6 Modelli SARIMA

Si affronta ora l'ultima generalizzazione dei processi ARIMA: i processi SARIMA, ovvero quei processi che presentano una componente periodica o stagionale. In generale tale componente può comportarsi a sua volta come un modello ARIMA, perciò un modello SARIMA può essere visto come un modello ARIMA i cui residui sono ancora un modello ARIMA, e un modello SARIMA può essere indicato con la scrittura ARIMA$(p,d,q)(P,D,Q)_S$, dove S rappresenta il ritardo stagionale, P,D,Q invece sono gli ordini del modello ai ritardi multipli di S. La formula generale di un modello ARIMA$(p,d,q)(P,D,Q)_S$ è la seguente:

$$\phi(B)\Phi(B^S)(1-B)^d(1-B^S)^D Y_t = \phi_0 + \theta(B)\Theta(B^S)\varepsilon_t,$$

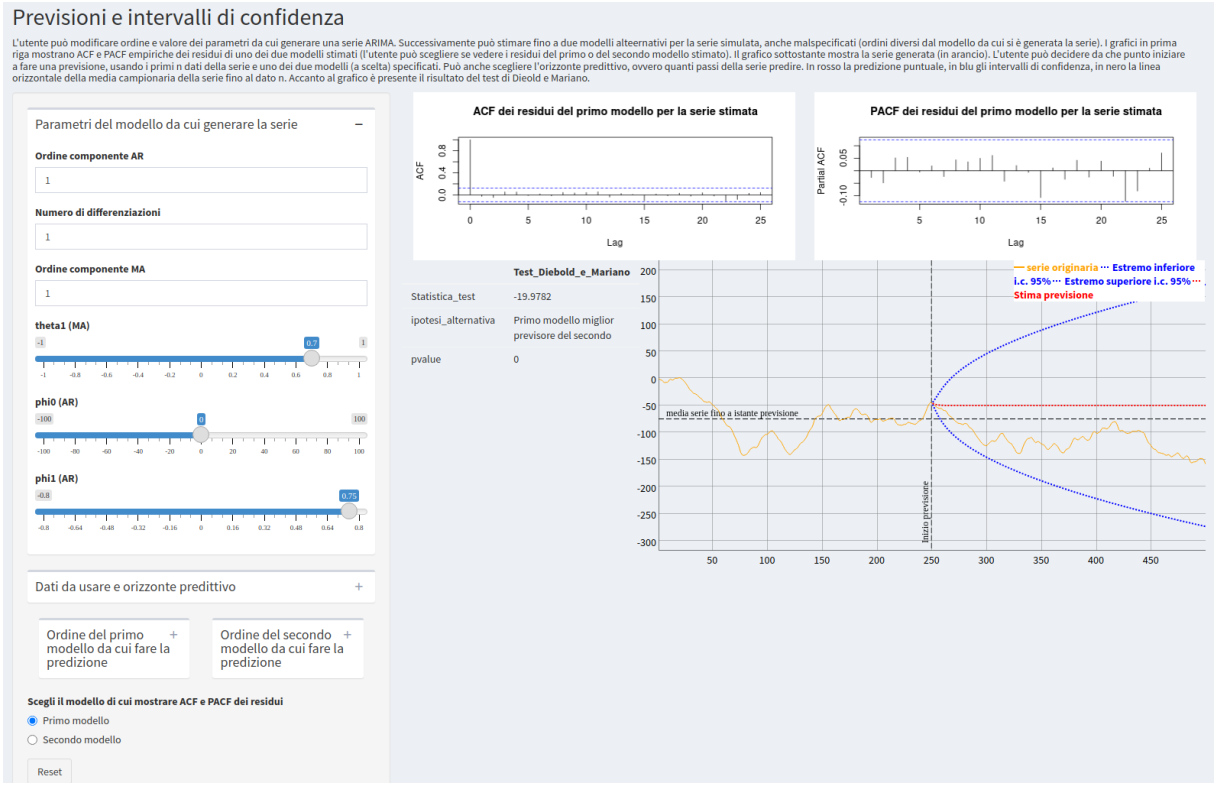


Figura 2.12: Dodicesima pagina dell'applicazione: Previsione.

dove, rispetto ai polinomi già conosciuti dei modelli ARIMA non stagionali, S è il ritardo stagionale, $\Phi(B^S) = (1 - \Phi_1 B^S - \Phi_2 B^{2S} - ... - \Phi_P B^{PS})$ e $\Theta(B^S) = (1 - \Theta_1 B^S - \Theta_2 B^{2S} - ... - \Theta_Q B^{QS})$.

Stazionarietà ed invertibilità sono legate, come nel caso ARIMA al fatto che le radici di $\phi(B)\Phi(B^S)$ e $\theta(B)\Theta(B^S)$ sono in modulo maggiori di 1. Per esempi e una trattazione più esauriente, si veda Di Fonzo e Lisi 2005, pag. 186 e seguenti.

2.6.1 ACF e PACF empiriche

La prima pagina dell'ultima sezione ha un approccio all'analisi delle serie SARIMA simile a quello per le serie non stagionali. In Figura 2.13 viene mostrato l'aspetto della pagina per un modello ARIMA$(0,1,1)(0,1,1)_4$ (questo particolare modello viene chiamato airline, con S in generale diverso da 4). Il grafico superiore indica la serie generata (il cui ordine e valore dei parametri sono a discrezione dell'utente), mentre i due grafici sottostanti mostrano la corrispettiva ACF e PACF empiriche. Si nota, sia dal grafico della serie che da quello dell'ACF, che la serie non è stazionaria né in media né in varianza. La serie tuttavia ha un comportamento "a

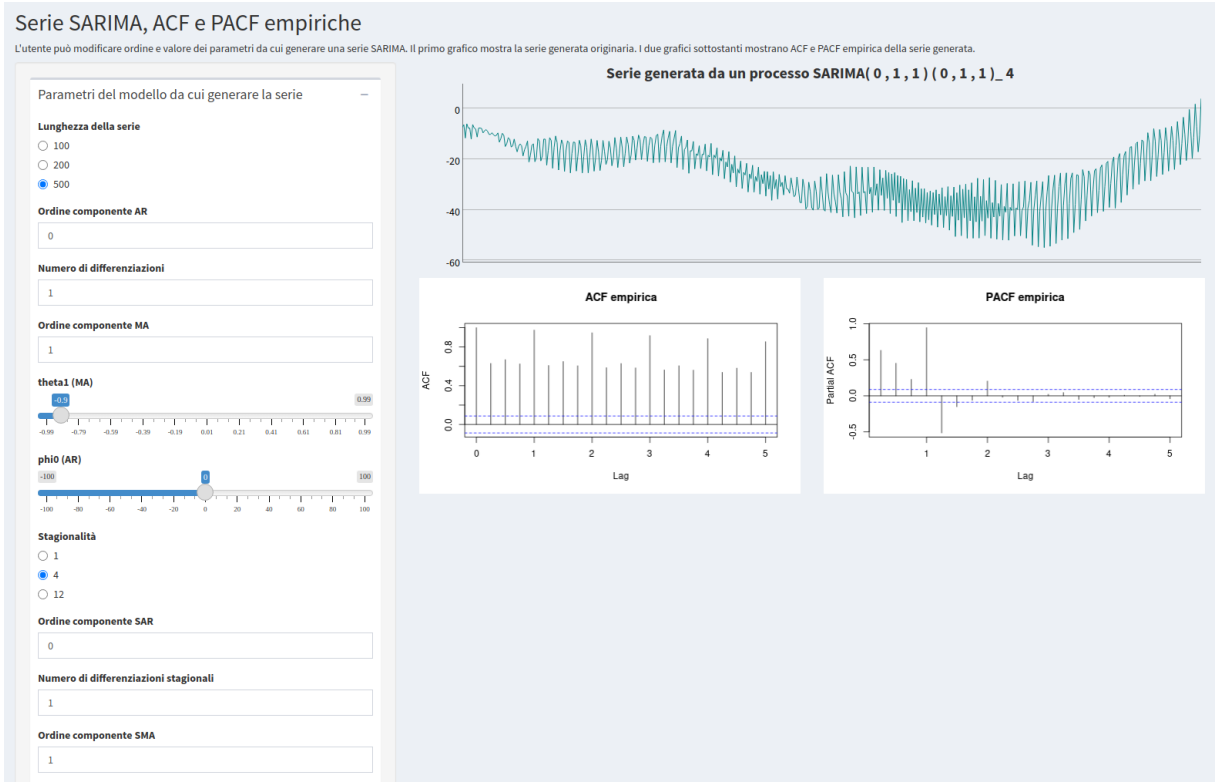


Figura 2.13: Tredicesima pagina dell'applicazione: ACF e PACF empiriche di un ARIMA stagionale.

zig-zag", con ripetizione della stessa struttura ogni quattro punti (infatti la serie è stagionale di periodo 4). L'ACF mostra invece valori che scendono molto lentamente, ma ogni quattro ritardi si registra un picco in controtendenza. In effetti considerando solo l'ACF a ritardi multipli di 4, si può osservare un altro trend che scende molto lentamente, come se all'interno di un processo ve ne fosse innestato un altro.

2.6.2 Diagnostica delle serie SARIMA

L'ultima pagina dell'applicazione affronta la tematica della diagnostica del modello e dell'analisi dei residui. Nella forma è piuttosto simile alla corrispondente pagina nella sezione sulla procedura di Box e Jenkins per i processi ARIMA. L'utente (Figura 2.14) può generare una serie da un processo ARIMA stagionale, e poi cercare di adattare un modello sulla serie storica sopra generata. A questo punto, nel riquadro di tabelle in alto a sinistra compaiono coefficienti stimati e relativi standard error, le statistiche del test di Jarque e Bera. È presente anche il test di Pierce (Pierce 1978): esso viene usato per determinare la presenza di correlazione residua

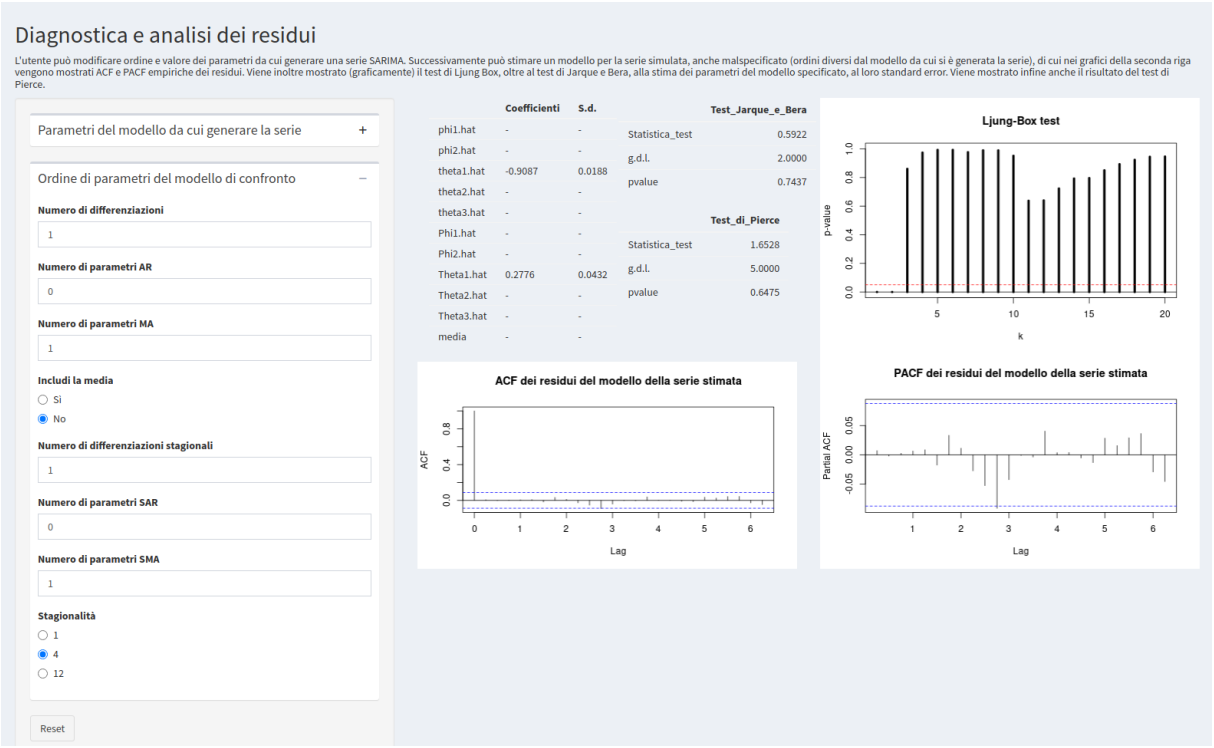


Figura 2.14: Quattordicesima pagina dell'applicazione: Stima dei parametri, diagnostica e analisi dei residui di una serie ARIMA stagionale.

stagionale. Si tratta di una modifica stagionale della statistica di Ljung e Box vista in precedenza. Quest'ultima, se non modificata, infatti rischia di sottovalutare i ritardi stagionali. Il test di Pierce nell'app è calcolato per i primi cinque ritardi stagionali. È calcolato anche il p-value della statistica, statistica che in assenza di correlazione si distribuisce come un χ^2_5. Per il resto, affianco è presente la visualizzazione grafica del test di Ljung e Box per i primi venti ritardi (per fare confronti tra il risultato del test di Pierce e quello di Ljung e Box). In basso ACF e PACF empiriche dalla serie dei residui del modello stimato.

Nell'esempio mostrato si è scelto di generare la serie dallo stesso modello airline della precedente sezione, compresi i valori dei parametri. Il modello proposto per la stima della serie derivata è stato correttamente specificato. Il risultato è una stima quasi coincidente con i valori dei veri parametri, mentre sia il test di Pierce che Ljung e Box che Jarque e Bera che l'ACF e PACF dei residui indicano un buon adattamento.

Capitolo 3

Osservazioni conclusive

La lunga spiegazione e descrizione del secondo capitolo sono indicative della vastità delle nozioni teoriche sui processi stocastici esistenti. Quanto esposto nell'applicazione implementata è poi in realtà soltanto una frazione, quella rappresentabile, di tutto questo materiale. Allo stesso modo, le note sulla funzione *shiny* nel primo capitolo sono soltanto un riassunto incompleto del vastissimo mondo delle funzioni ad essa collegate. Il senso secondario di questo laboratorio è stato dunque anche quello di coinvolgere due ambiti differenti, ma fortemente intrecciati tra loro: quello teorico-statistico e quello computazionale-informatico. I due aspetti, data l'evoluzione delle tecniche algoritmiche odierne, non possono rimanere incorrelati. Una statistica senza strumenti informatici non è infatti sostenibile con dei calcoli così complessi, così come lo strumento informatico, senza un'adeguata base teorica, rischia di generare analisi fuorvianti.
Si crede dunque che questo strumento possa essere un passo verso una costante integrazione tra teoria e informatica. Lo scopo principale poi di offrire un potenziamento didattico e di studio sembra raggiunto, almeno parzialmente. Certamente, alcuni argomenti sono stati affrontati in maniera puramente grafica o poco approfondita, ma si tratta in genere di concetti meno importanti o difficilmente rappresentabili nella maniera interattiva dell'app. Gli obiettivi di un successivo lavoro su questo materiale riguardano innanzi tutto la "saldatura" tra l'app implementata e quella simile di cui si è detto in precedenza sull'approccio tradizionale alle serie storiche. Successivamente sono necessari un miglioramento e l'ampliamento delle performance dell'app. Sarebbe inoltre utile includere nuovi argomenti e ulteriori generalizzazioni, come i modelli GARCH e ARCH per la varianza condizionata, al fine di offrire un dispositivo ancora più esauriente.

Bibliografia

Boshnakov, Georgi N. e Jamie Halliday (2024). *sarima: Simulation and Prediction with Seasonal ARIMA Models*. R package version 0.9.3. url: https://CRAN.R-project.org/package=sarima.

Box, George EP et al. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Chang, Winston e Barbara Borges Ribeiro (2021). *shinydashboard: Create Dashboards with 'Shiny'*. R package version 0.7.2. url: https://CRAN.R-project.org/package=shinydashboard.

Chang, Winston, Joe Cheng et al. (2024). *shiny: Web Application Framework for R*. R package version 1.8.1.1. url: https://CRAN.R-project.org/package=shiny.

Di Fonzo, Tommaso e Francesco Lisi (2005). *Serie storiche economiche. Analisi statistiche e applicazioni*. Carocci.

Diebold, Francis X e Roberto S Mariano (1995). "Comparing predictive accuracy". In: *Journal of Business and Economic Statistics* 13.3, pp. 253–263.

Hyndman, Rob et al. (2024). *forecast: Forecasting functions for time series and linear models*. R package version 8.23.0. url: https://pkg.robjhyndman.com/forecast/.

Jarque, Carlos M e Anil K Bera (1980). "Efficient tests for normality, homoskedasticity and serial independence of regression residuals". In: *Economics letters* 6.3, pp. 255–259.

Ljung, Greta M e George EP Box (1978). "On a measure of lack of fit in time series models". In: *Biometrika* 65.2, pp. 297–303.

Pierce, David A (1978). "Seasonal adjustment when both deterministic and stochastic seasonality are present". In: *Seasonal analysis of economic time series*. NBER, pp. 242–280.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. url: https://www.R-project.org/.

Stoffer, David e Nicky Poison (2024). *astsa: Applied Statistical Time Series Analysis*. R package version 2.1. url: https://CRAN.R-project.org/package=astsa.

Trapletti, Adrian e Kurt Hornik (2024). *tseries: Time Series Analysis and Computational Finance*. R package version 0.10-56. url: https://CRAN.R-project.org/package=tseries.

Vanderkam, Dan et al. (2018). *dygraphs: Interface to 'Dygraphs' Interactive Time Series Charting Library*. R package version 1.1.1.6. url: https://CRAN.R-project.org/package=dygraphs.