Università degli Studi di Padova

Dipartimento di Scienze Statistiche

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

Relazione Finale

Decomposizione di serie storiche con stagionalità complessa: metodi a confronto

Relatore: Prof.ssa Luisa Bisaglia

Dipartimento di Scienze Statistiche

Laureanda: Linda Pagnin

Matricola N. 1150952

Anno Accademico 2022/2023

Indice

Introduzione

La decomposizione delle serie storiche consiste nel suddividere una serie temporale in tre componenti principali: una componente di trend, una stagionale e una residua.

Molti dataset reali, però, possono contenere stagionalità multiple e quindi mostrare un andamento più complesso. Queste serie storiche prendono il nome di serie storiche con stagionalità multipla o complessa.

Esistono dei metodi per decomporre una serie temporale con stagionalità complessa; quelli che saranno trattati in questa tesi sono MSTL, BATS, TBATS e STR.

MSTL è l'acronimo di *Multiple Seasonal-Trend decomposition using Loess*. È un algoritmo di decomposizione accurato che riesce a catturare tutti gli schemi stagionali presenti in una serie storica (Bandara, Hyndman, Bergmeir, 2021).

STR (acronimo che sta per *Seasonal-Trend decomposition using Regression*) è un metodo per la decomposizione di serie stagionali che utilizza la regressione per decomporre una serie con stagionalità multipla (Dokumentov e Hyndman, 2021).

Il modello BATS è un metodo di *smoothing* esponenziale per serie storiche con stagionalità multipla. L'acronimo sta per B: Box-Cox *transformation* che consiste nella trasformazione di una variabile di-

pendente non Normale in modo che si avvicini, in quanto a forma, ad una distribuzione Normale, A: ARMA *errors*, T: *trend* e S: stagionalità (De Livera, Hyndman, Snyder, 2010).

TBATS è la forma trigonometrica (l'aggiunta della T nell'acronimo) del modello BATS e questo permette di modellare anche serie storiche che hanno una frequenza di stagionalità non intera (De Livera, Hyndman, Snyder, 2010).

Nei prossimi capitoli verrà mostrato come si comportano i vari metodi applicati ai dati provenienti dall'azienda Terna - Driving Energy. La serie storica presa in considerazione ha una durata di tre anni e riguarda l'andamento del fabbisogno totale di energia del sistema elettrico italiano. Nel primo capitolo verranno descritti i dati utilizzati e i metodi di decomposizione con tutte le loro caratteristiche, nel secondo sarà presentata l'analisi esplorativa, nel terzo i risultati sia in termini statistici che computazionali e, infine, le conclusioni.

Capitolo 1

Dati e metodi

In questo capitolo vengono presentati i dati e i metodi usati nella fase di analisi. Nello specifico si parte dalla descrizione dei dati utilizzati nello studio per poi passare alla rassegna delle specifiche scelte statistiche fatte per modellare la relazione presa in esame.

1.1 Dati

L'analisi è stata svolta sui dati Terna - Driving Energy, azienda che gestisce la rete di trasmissione nazionale italiana dell'elettricità in alta tensione. È stata presa in considerazone la serie storica multistagionale del fabbisogno totale del sistema elettrico italiano dal gennaio 2020 al dicembre 2022 (dati disponibili al sito Terna).

Il dataset iniziale includeva le rilevazioni fornite dall'azienda ogni 15 minuti e riferite a varie zone italiane (centro-sud, centro-nord, nord, sud, Calabria, Sicilia, Sardegna); in una fase preliminare dell'analisi i dati sono stati accorpati per ora e per l'intera Italia, quindi sommando tutte le varie zone in modo da sintetizzarle in un unico valore orario. La variabile osservata è riferita all'andamento del fabbisogno totale del sistema elettrico italiano.

Sono stati valutati tutti i modelli proposti per vedere quale sia il migliore sia in termini statistici che computazionali.

Il *software* usato è `R`, e le funzioni utilizzate sono disponibili nel pacchetto *forecast* (Hyndman et al., 2021, Forecast) e nel pacchetto *stR* (Dokumentov & Hyndman, 2018, stR).

1.2 Metodi

In seguito, vengono presentati nel dettaglio i metodi di decomposizione per serie storiche multistagionali citati nell'introduzione.

1.2.1 MSTL

Il metodo di decomposizione MSTL è un metodo di lisciamento che produce una decomposizione additiva della serie storica con stagionalità complessa. È un metodo completamente automatizzato per trattare le serie storiche multistagionali.

Essendo MSTL un'estensione di STL, se la serie storica è multistagionale, le componenti stagionali sono stimate usando in maniera iterativa quest'ultimo. È possibile che durante la decomposizione le componenti stagionali relative ad un ciclo stagionale inferiore vengano assorbite da un ciclo stagionale superiore (*nesting*). Inizialmente MSTL determina gli schemi stagionali presenti nella serie storica, successivamente la funzione di MSTL applica in maniera iterativa l'algoritmo di STL su ogni frequenza stagionale identificata, e infine, la componente di *trend* è calcolata usando l'ultima iterazione di STL (Bandara, Hyndman, Bergmeir, 2021).

Quindi l'equazione per il modello MSTL con uno schema stagionale multiplo è:

$$X_t = \hat{S}_t^1 + \hat{S}_t^2 + \ldots + \hat{S}_t^n + \hat{T}_t + \hat{R}_t$$

dove:

X_t : sono le osservazioni al tempo t

$S_t^1, \ldots, S_t^n$: sono le componenti stagionali

T_t : è la componente di trend

R_t : è la componente residuale

Per calcolare, invece, la parte residuale R_t della serie temporale multistagionale, la componente di *trend* viene sottratta dalla serie temporale destagionalizzata.

Inoltre, MSTL può essere usato per anche periodi stagionali non interi.

1.2.2 BATS

Il metodo di decomposizione additiva BATS restituisce un'equazione con schema stagionale multiplo.

È un modello lineare omoschedastico che incorpora la non-linearità integrando la trasformazione di Box-Cox e un ARMA stazionario (Naim, Mahara, Idrisi, 2018).

I parametri di questo modello sono $(\omega, \phi, p, q, m_1, \ldots, m_T)$ che indicano nell'ordine: il parametro per la trasformazione di Box-Cox $\omega \in [0, 1]$, che viene applicata quando $\omega \approx 1$, il parametro di lisciamento ϕ, p e q sono i parametri del processo ARMA e $m_1, \ldots, m_T$ le

multistagionalità.

L'equazione per questo modello è fatta di più componenti (Naim, Mahara, Idrisi, 2018):

$$y_t^{(\omega)} = \begin{cases} \frac{y_t^{\omega}-1}{\omega} & \omega \neq 0 \\ \log y_t & \omega = 0 \end{cases} \tag{1.1}$$

che identifica la trasformazione di Box - Cox con il rispettivo parametro ω al tempo t,

$$l_t = l_{t-1} + \phi b_{t-1} + \alpha d_t, \tag{1.2}$$

è la componente di livello,

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t, \tag{1.3}$$

la componente di trend,

$$s_t^{(i)} = s_{t-m_i}^{(i)} + \gamma_i d_t, \tag{1.4}$$

la i-esima componente stagionale, e infine

$$d_t = \sum_{i=1}^{p} \varphi_i d_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \tag{1.5}$$

che è il processo ARMA usato per modellare l'errore.

Nello specifico ogni parametro rappresenta:

$m_1, \ldots, m_T$: i periodi dei cicli stagionali

l_t: la componente di livello al tempo t

b : il trend di lungo periodo

b_t: trend di breve periodo al periodo t

$s_t^{(i)}$: i - esima componente stagionale al tempo t

d_t : processo ARMA(p, q)

ϵ_t: white noise Normale $\sim (0, \sigma^2)$, σ^2 *costante*

α, β, γ_i: parametri di lisciamento per i = 1,...,T

ϕ : parametro di lisciamento per il trend, poi affiancato dal parametro b citato sopra

φ_i, θ_i: coefficienti p, q del processo ARMA

Una limitazione di questo metodo è che non si adatta a stagionalità non intere e richiede un gran numero di parametri per la stima del modello. Infine, il valore iniziale si calcola sommando le stagionalità $m_1, \ldots, m_T$ del modello stimato (De Livera, Hyndman, Snyder, 2010).

1.2.3 TBATS

Il metodo TBATS è la forma trigonometrica del modello BATS basata sulle serie di Fourier. È un metodo innovativo e computazionalmente più veloce rispetto al metodo BATS. I parametri di questo nuovo modello sono $(\omega, \phi, p, q, \{m_1, k_1\}, \{m_2, k_2\}, \ldots, \{m_T, k_T\})$ che indicano nell'ordine: ω è il parametro per la trasformazione di Box-Cox, ϕ è il parametro di lisciamento, p e q sono i parametri del processo ARMA, le coppie $\{m_T, k_T\}$ identificano la stagionalità e il corrispondente termine della serie di Fourier.

L'equazione relativa al metodo TBATS varia rispetto al metodo BATS soltanto per la parte stagionale. Le formule del metodo BATS (1.1), (1.2), (1.3), (1.5) rimangono invariate, la (1.4) invece diventa

(De Livera, Hyndman, Snyder, 2010):

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$$

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \gamma_1^{(i)} d_t$$

$$s_{j,t}^{*(i)} = -s_{j,t-1} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t$$

dove:
$\gamma_1^{(i)}, \gamma_2^{(i)}$: sono i parametri di lisciamento
$s_{j,t}^{(i)}$: è il livello stocastico alla i-esima componente stagionale
$s_{j,t}^{*(i)}$: è la crescita stocastica nel livello della i-esima componente stagionale e descrive il cambiamento di questa nel tempo
k_i: sono il numero di armonizzazioni necessarie per la i-esima componente stagionale

TBATS può modellare anche frequenze stagionali non intere. Anche in questo metodo, come per BATS, è richiesta la stima di valori iniziali stagionali, che per il metodo TBATS si calcolano con la formula $2(k_1, \ldots, k_T)$ che prende in considerazione i coefficienti della serie di Fouerier.

1.2.4 STR

STR permette di fare analisi su serie storiche che hanno stagionalità multiple, componenti cicliche, covariate, schemi stagionali che possono non avere periodi interi e stagionalità con una topologia complessa (Dokumentov e Hyndman, 2021).

STR può essere usato per serie storiche orarie, giornaliere settimanali ecc..

L'equazione di decomposizione additiva della serie storica y_t è:

$$y_t = T_t + \sum_{i=1}^{I} S_t^{(i)} + \sum_{p=1}^{P} \phi_{p,t}\zeta_{t,p} + R_t$$

dove:

T_t : è il trend

$S_t^{(i)}$: sono componenti stagionali

$z_{p,t}$: sono covariate; il loro coefficiente è $\theta_{p,t}$ che può cambiare con il tempo e la stagionalità

R_t: sono i residui

La componente di *trend* riflette l'andamento medio dei dati dopo aver tenuto conto della stagionalità; la componente stagionale, invece, si assume segua uno schema ripetuto che cambia poco o rimane costante nel tempo. Infine, la componente residuale è tutto ciò che le componenti di *trend* e stagionalità non riescono a spiegare.

Capitolo 2

Analisi esplorativa

In questo capitolo verrà presentata l'analisi svolta con il *software* `R` sui dati Terna - Driving Energy dal gennaio 2020 a dicembre 2022.

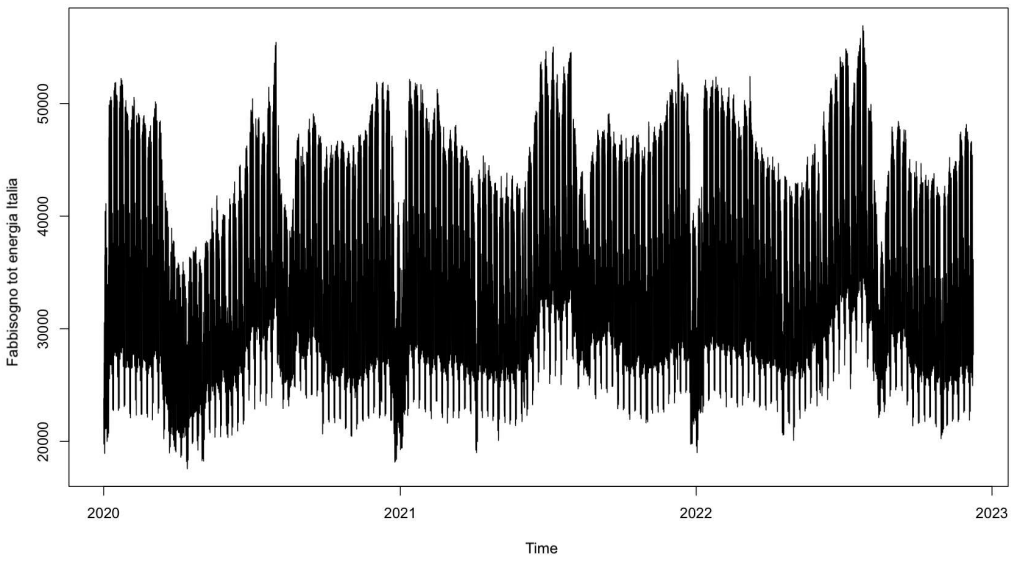

Figura 2.1: Dati Terna con rilevazioni orarie della domanda energetica in Italia su un periodo di 3 anni.

La Figura 2.1 rappresenta il grafico dei dati aggregati per ora. Già da questo grafico si nota il *pattern* della stagionalità annuale. Una volta che la serie storica sarà decomposta saranno visibili anche il *pattern* settimanale e quello giornaliero.

Per provare ad individuare le diverse stagionalità del dataset, sono stati presi in considerazione 15 giorni, dal 14 gennaio 2020 al 29 gennaio 2020, per un totale si 360 osservazioni (Figura 2.2).

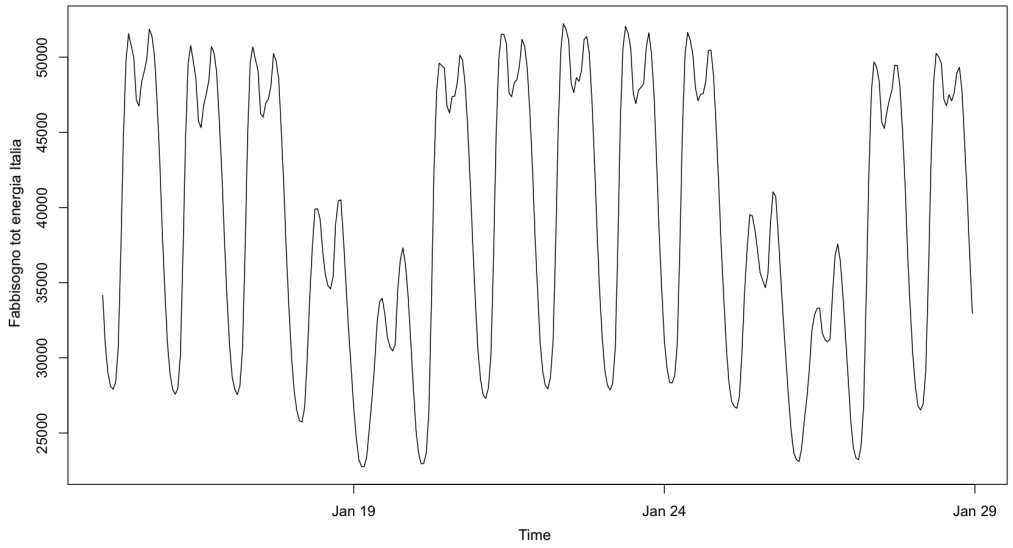

Figura 2.2: Dati Terna con rilevazioni orarie su un periodo di due settimane dal 14 gennaio al 29 gennaio 2020.

L'andamento stagionale giornaliero è evidente. Valutando successivamente il grafico dei dati su un periodo di 30 giorni (720 osservazioni) visibile in Figura 2.3, dal 14 gennaio al 14 febbraio 2020 si potrà vedere anche lo schema stagionale settimanale. In questo grafico la stagionalità giornaliera ha un ciclo che si ripete: le due linee rosse mostrano un periodo della durata di 7 giorni, e si nota come nelle settimane successive questo ciclo si reiteri in maniera ciclica.

Nei dati orari di Terna - Driving Energy sono quindi presenti tre diverse stagionalià: quella giornaliera, settimanale e annuale.

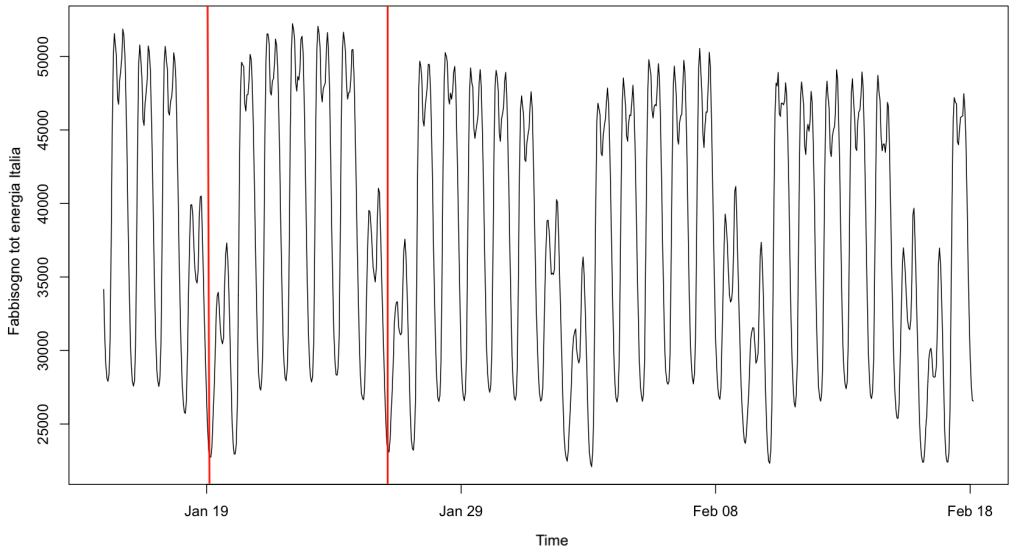

Figura 2.3: Dati Terna con rilevazioni orarie su un periodo di un mese dal 14 gennaio al 14 febbraio 2020.

Da questa breve analisi grafica, si procede con l'analisi vera e propria, quindi creando l'oggetto serie storica multistagionale. Questa serie classe `msts` ha come `seasonal.periods` il vettore c = (24(giornaliera), 168(settimanale) e 8766(annuale)) (dal sito di Rob J Hyndman: Stagionalità).

L'accorpamento dei dati per ora è stato eseguito principalmente per non dilungare ancora di più i tempi computazionali che si sono rivelati, per alcuni metodi, già abbastanza cospicui.

2.1 Metodo di valutazione

La precisione dei modelli stimati con i diversi metodi di decomposizione è stata valutata utilizzando l'RMSE (*Root mean squared error*) definito come:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_t - \hat{X}_t)^2},$$

dove:

X_t: è il vero valore della decomposizione al tempo t

$\hat{X}_t$: è la stima valore della decomposizione al tempo t

n: è il numero di osservazioni della serie storica

In aggiunta, verranno considerati anche i tempi computazionali come fattore per capire, anche sotto il punto di vista della rapidità di esecuzione, quale sia il metodo migliore.

2.1.1 MSTL

MSTL ha fornito buoni risultati rispetto agli altri metodi sia in termini di RMSE sia in termini computazionali: questa metodologia in pochi secondi restituisce i risultati. Gli altri metodi, invece, ci impiegano alcune ore, BATS addirittura delle giornate.

La funzione utilizzata per l'analisi di questo metodo è `mstl()` e la si trova all'interno del pacchetto *forecast* di `R` (Hyndman et al., 2021).

Dalla Figura 2.4 si può vedere che il *trend* ha una crescita regolare, liscia e omogenea il che significa che la stagionalità è stata decomposta nel suo totale.

Prendendo in considerazione solo la stagionalità `seasonal168`, che è la stagionalità settimanale, e `seasonal8766` che è quella annuale,

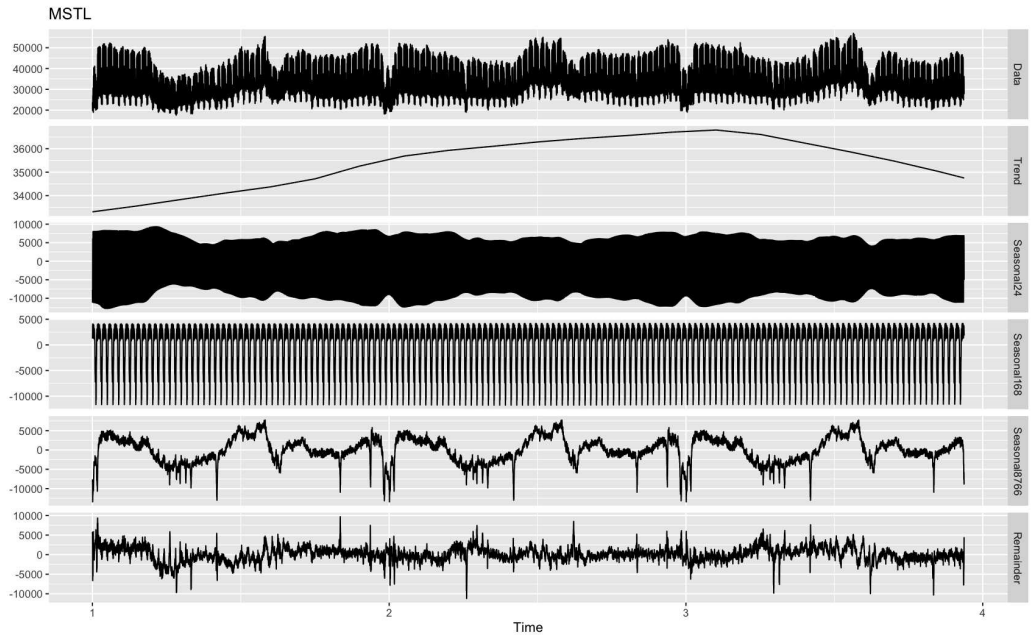

Figura 2.4: Decomposizione della domanda di energia oraria dell'Italia usando MSTL. In ordine dall'alto i pannelli mostrano i dati originali, il trend, le stagionalità giornaliera, settimanale, annuale e i residui.

si nota come in entrambe ci sia uno schema che si ripete: nella prima, `seasonal168`, ci sono picchi settimanali costanti, nella seconda, `seasonal8766`, il *pattern* nell'arco di ogni anno si reitera per quello successivo.

I residui sono omogenei tranne per qualche picco che indica che è ancora presente della stagionalità residua. Questo può essere giustificato dal fatto che quelle giornate fossero particolari dal punto di vista metereologico, quindi giornate calde o fredde oppure festività.

Inoltre, dai grafici di autocorrelazione globale (ACF) e parziale (PACF) in Figura 2.5, si nota che ogni coefficiente di autocorrelazione a uno o più ritardi è significativamente diverso da zero poichè ogni ritardo non è compreso nelle linee tratteggiate blu che indicano l'intervallo di confidenza con livello di confidenza $1-\alpha = 0.95$. Questo implica che i residui siano autocorrelati e ci sia quindi dipendenza tra

i valori assunti da una funzione nel suo dominio in ascissa.

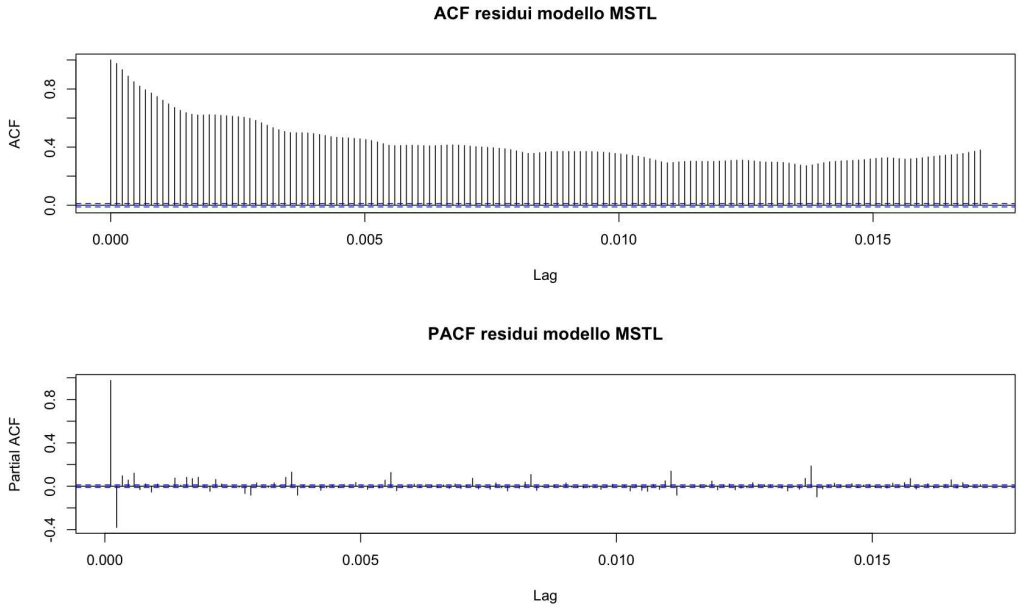

Figura 2.5: ACF e PACF residui del modello stimato con il metodo MSTL.

Il parametro `s.window` è stato impostato con "periodic" poiché nel grafico dei dati reali in Figura 2.1 non c'è un'evoluzione veloce dello schema stagionale della serie storica, ma è pressoché costante e ciclica (Bandara, Hyndman, Bergmeir, 2021).

2.1.2 BATS

BATS si è rivelato il peggior metodo, sia in termini di RMSE e soprattutto in termini computazionali. La funzione usata è `bats()` che è presente nel pacchetto *forecast* di `R` (Hyndman et al., 2021).

Il modello stimato con questo metodo è
BATS(0.00934, 0.8, {3, 2}, {24, 168, 8760}). Non è stata usata la trasformazione di Box-Cox poichè $\omega = 0.00934$ è vicino allo 0.

Il parametro di lisciamento è $\phi = 0.8$ e la parte residuale del modello è un processo ARMA(3, 2).

In Figura 2.6 viene rappresentato il grafico della decomposizione con questo metodo: dall'alto, come poi si vedrà per la decomposizione con il metodo TBATS, si trovano i dati osservati, il livello (valore medio per la durata della serie storica), la pendenza che è il cambiamento del livello per unità di tempo e, infine, le stagionalità giornaliera, settimanale e annuale.

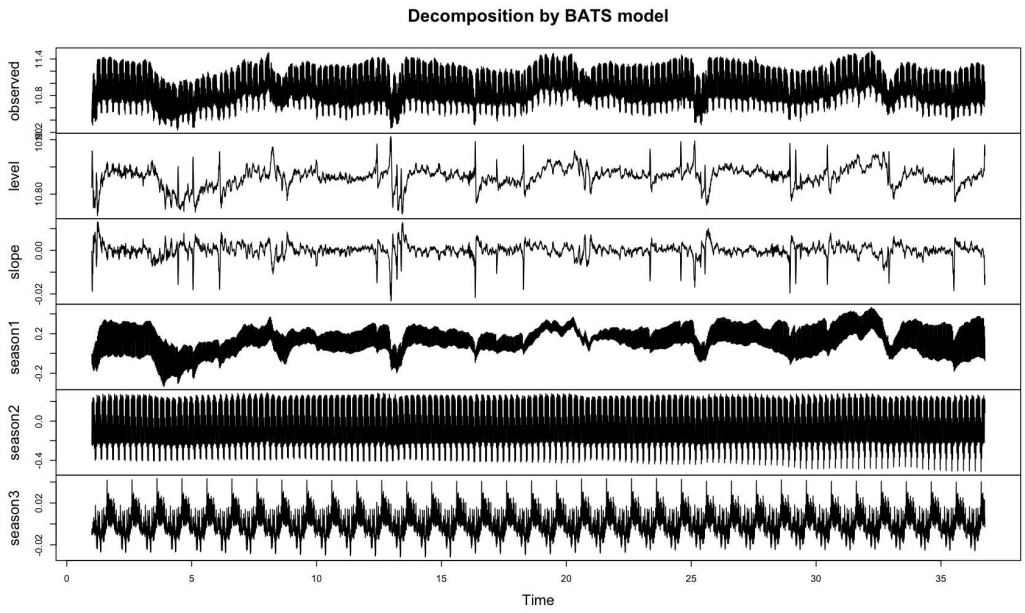

Figura 2.6: Decomposizione con il metodo BATS. In ordine dall'alto i pannelli mostrano i dati originali, il livello, la pendenza e le tre stagionalità: giornaliera, settimanale e annuale.

La Figura 2.7 mostra i residui del modello stimato con il metodo BATS. Come è evidente, ci sono molti più valori anomali rispetto a tutti gli altri modelli stimati che sono stati precedentemente presentati. Anche nella stima di questo modello, come si può vedere in Figura 2.8, i residui sono autocorrelati.

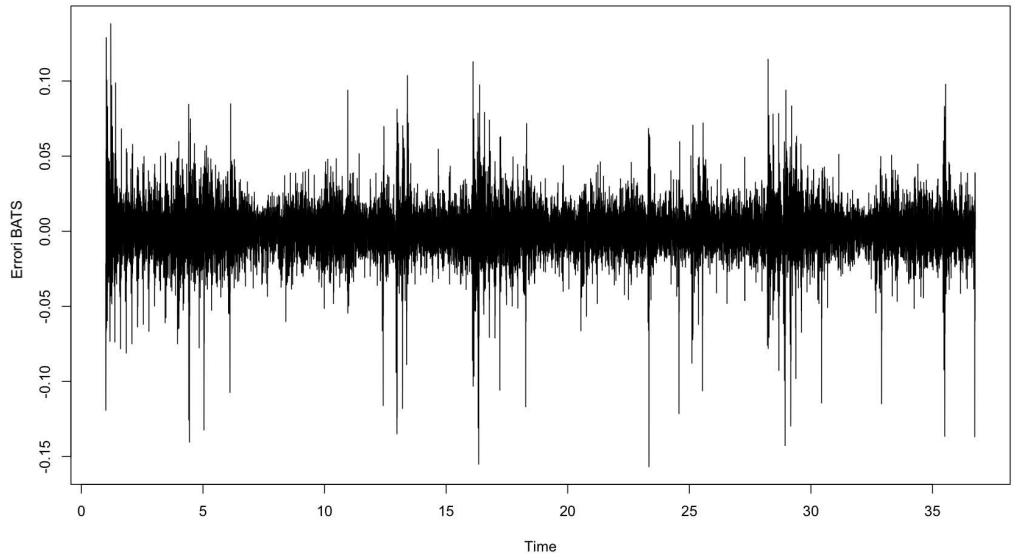

Figura 2.7: Residui del modello BATS

La stima del valore iniziale stagionale del modello stimato con il metodo BATS si calcola sommando le stagionalità $m_1, \ldots, m_T$ e cioè $24 + 168 + 8760 = 8952$. Nella sezione successiva, relativa al metodo TBATS, verrà mostrato come con questo secondo metodo l'*initial seasonal value* sia molto più basso (Naim et al., 2018).

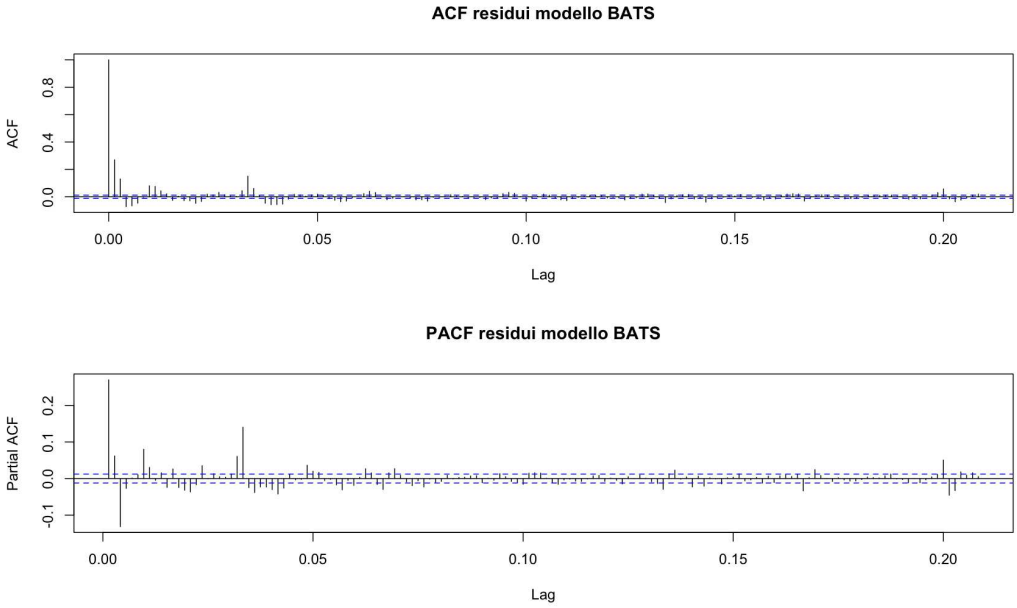

Figura 2.8: ACF e PACF dei residui del modello stimato con il metodo BATS.

2.1.3 TBATS

Rispetto alla sua forma non trigonometrica (BATS), TBATS ha fornito risultati migliori.

La funzione usata è `tbats()` all'interno del pacchetto *forecast* di `R` (Hyndman et al., 2021).

Nella Figura 2.9 si può notare che c'è forte stagionalità nelle componenti giornaliera e settimanale (`season1` e `season2`) rispetto alla stagionalità nella componente annuale (`season3`) che è più regolare.

Il modello stimato per questo metodo è un TBATS(0.934, {4, 1}, {24, 6}, {168,6}, {8766, 7}. Qui la trasformazione di Box-Cox è stata utilizzata poichè $\omega = 0.934$ è vicino a 1. I periodi stagionali sono accoppiati con i rispettivi termini della serie di Fourier. Infine, la componente dell'errore è stata modellata con un ARMA(4,1).

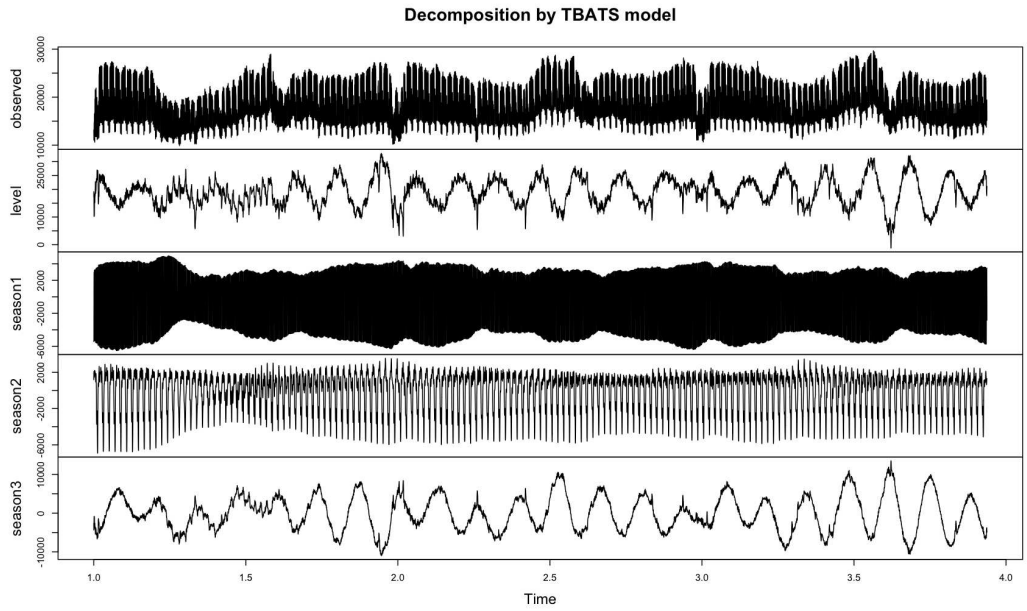

Figura 2.9: Grafico della decomposizione usando il metodo TBATS. In ordine dall'alto i pannelli mostrano i dati osservati, il livello e le tre stagionalità (giornaliera, settimanale e annuale).

La Figura 2.10, invece, mostra il grafico dei residui del modello TBATS. Si nota che i residui sono abbastanza omogenei eccetto per alcuni picchi che possono essere giustificati come per il modello MSTL: giornate molto calde o fredde oppure festività.

Nel grafico dell'ACF e PACF in Figura 2.11, si può notare l'autocorrelazione dei residui poichè sono sempre all'interno delle bande dell'intervallo di confidenza al 95%.

Per il modello TBATS il valore stagionale iniziale si calcola prendendo in considerazione i termini della serie di Fourier. L'equazione è $2(k_1, \ldots, k_T)$ che in questo caso diventa $2(6+6+7) = 38$ (Naim, Iram, mahara, Idrisi, 2018).

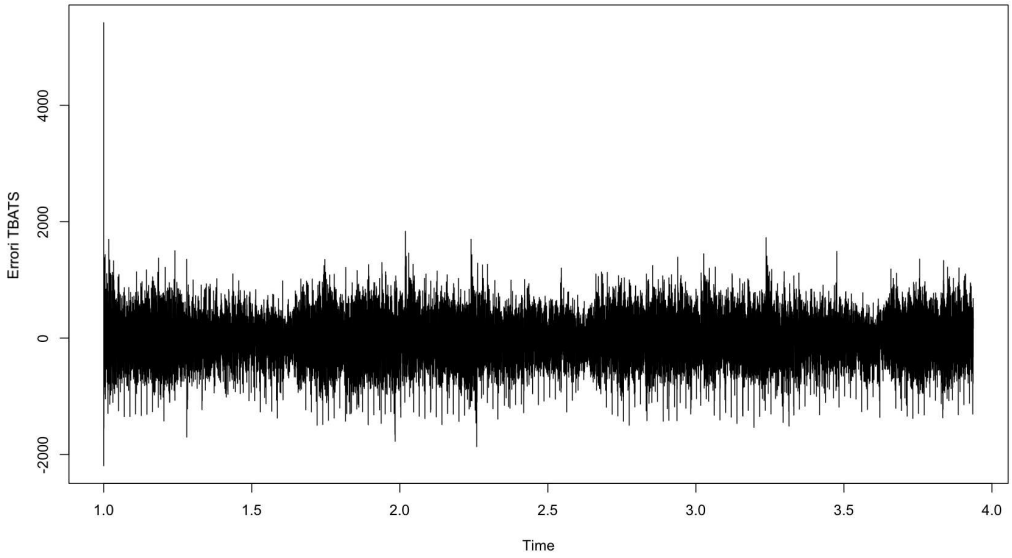

Figura 2.10: Grafico dei residui del modello TBATS.

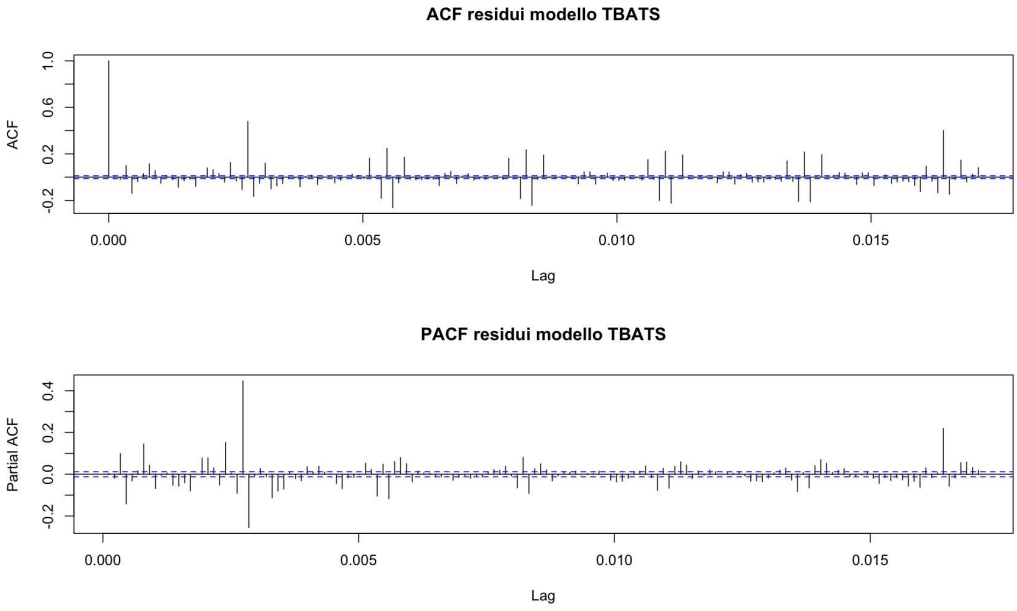

Figura 2.11: ACF e PACF dei residui del modello stimato con il metodo TBATS.

2.1.4 STR

Per questo metodo la funzione usata è stata `STR( )` all'interno del pacchetto *stR* (Dokumentov e Hyndman, 2018).

Il grafico in Figura 2.12 mostra la decomposizione svolta con questo metodo. Il parametro gapCV = 24 definisce la lunghezza della sequenza di valori saltati nella procedura di *cross validation*. È stato scelto questo valore poiché i dati sono orari e hanno un periodo di 24 osservazioni.

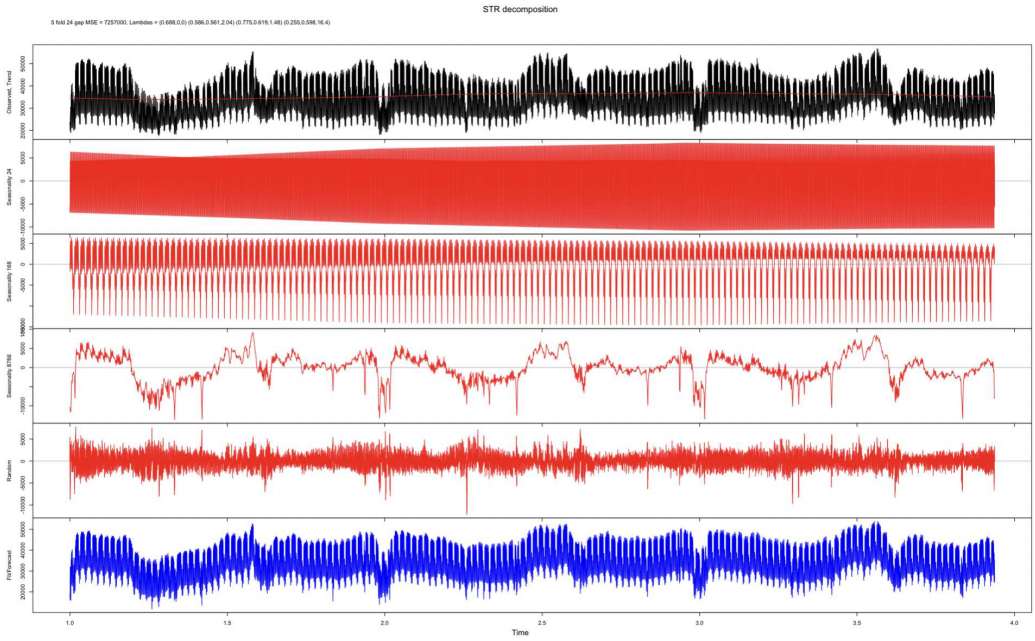

Figura 2.12: Grafico della decomposizione usando il metodo STR. Il contenuto di ogni panel dall'alto: dati osservati e trend (linea rossa), stagionalità giornaliera, settimanale, annuale, residui e le stime del modello per i dati osservati

Anche con questo metodo, se si guardano soltanto i panel con le componenti stagionali settimanale e annuale che sono più chiare rispetto a quella giornaliera, si nota che c'è forte stagionalità: nella prima c'è una continuità nei picchi e nella seconda si nota un pattern ricorrente

all'inizio e a metà di ogni anno che è molto simile alla decomposizione della stagionalità annuale con il metodo MSTL.

Per quanto riguarda i residui, dal correlogramma dell'ACF e PACF in Figura 2.13, si vede la ciclicità del legame di autocorrelazione, soprattutto nel grafico dell'ACF che dovuto al marcato fenomeno stagionale che influenza i dati.

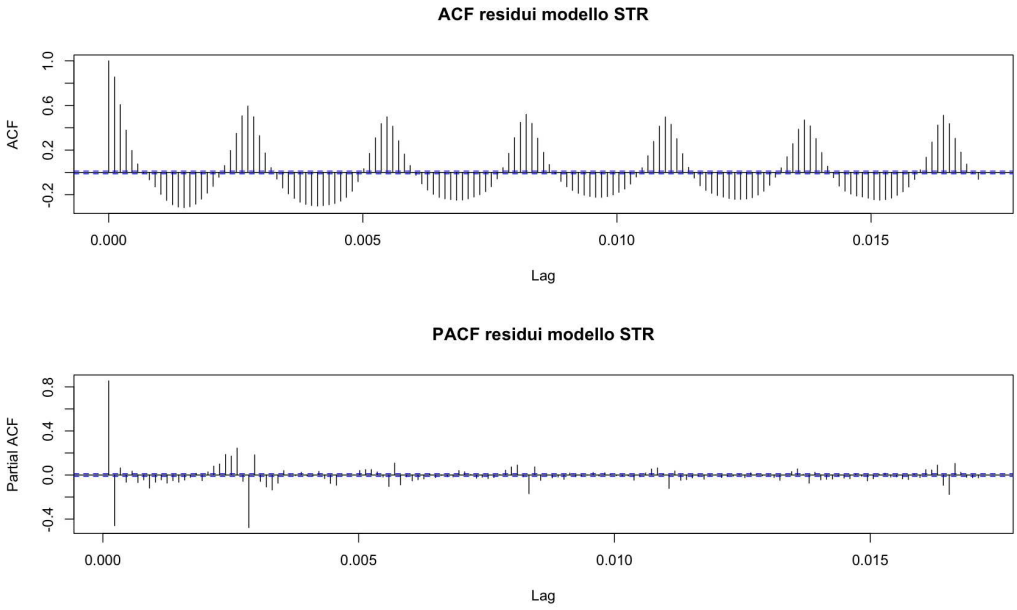

Figura 2.13: ACF e PACF dei residui del modello stimato con il metodo STR.

Capitolo 3

Risultati e discussione

Nel presente capitolo verranno presentati in dettaglio i risultati finali dell'analisi effettuata e una loro discussione, che copre sia l'aspetto statistico che quello computazionale.

Da un punto di vista statistico, si utilizza la radice dell'errore quadratico medio (RMSE).

Dall'altra parte, verrà data attenzione all'aspetto computazionale, concentrandosi sulle tempistiche di esecuzione per ogni specifica funzione.

3.1 Valutazione con RMSE

In questa analisi sono stati esaminati i metodi di decomposizione dei capitoli precedenti al fine di individuare quello più adatto.

La radice dell'errore quadratico medio o RMSE è stata calcolata con la funzione `accuracy()` del pacchetto *forecast* di `R` (Hyndman et al., 2021) che restituisce diverse misure sintetiche dell'accuratezza della previsione: ME, RMSE, MAE, MPE, MAPE, etc...

Nella Tabella 3.1 vengono mostrati gli RMSE della componente di *trend* e delle componenti stagionali per tutti i metodi utilizzati.

Tabella 3.1: RMSE per ogni componente stagionale e di trend per ogni metodo di decomposizione.

	RMSE				
Metodi	Trend	Giornaliero	Settimanale	Annuale	Residui
MSTL	1.618897	8.73939	1.991885	9.560292	251.8783
BATS	192.5589	597.9209	459.9205	767.2502	377.1803
TBATS	160.6412	467.6497	28.19781	307.4806	131.142
STR	1.195402	844.9334	312.207	10.11031	705.7451

Nel caso di questa analisi, dato che non viene fornito un *test set*, la funzione `accuracy()` produce solo misure di accuratezza del *training set* basate sulla formula $f["x"] - fitted[x]$. $f["x"]$ sono i vettori dei valori di *trend*, stagionalità giornaliera, settimanale, annuale e dei residui trovati con la decomposizione tramite i vari metodi; quindi dalla decomposizione con i diversi metodi si ottengono vettori di valori del *trend*, delle componenti stagionali e dei residui che sono differenti per ogni metodo. E, $fitted[x]$ è il vettore dei valori adattati trovati con la funzione `forecast()` (Hyndman et al., 2021) che viene applicata ai vettori dei valori ricavati dalla decomposizione. Quindi, facendo un esempio per comprendere meglio il funzionamento della funzione `accuracy()`, si prende in considerazione il metodo MSTL e la stagionalità giornaliera trovata con questo metodo; nella funzione `accuracy()` vengono inseriti i due oggetti $f["x"]$ e $fitted[x]$ che sono ripettivamente: il vettore dei valori trovati con il metodo MSTL relativo alla stagionalità giornaliera e il vettore dei valori adattati trovato con la funzione `forecast()` che ha in oggetto il vettore di valori relativo alla stagionalità giornaliera ricavato con il metodo MSTL, cioè $f["x"]$.

Dopo questa valutazione statistica basata sull'RMSE, si può concludere che il metodo che ha dato i risultati migliori sia MSTL poichè i valori dell'errore quadratico medio relativi ad ogni componente sono di gran lunga inferiori rispetto a quelli degli altri metodi.

3.2 Valutazione dei tempi computazionali

In relazione ai tempi computazionali per l'esecuzione delle singole funzioni, è stato evidenziato che MSTL è ancora il più efficiente tra tutti i metodi.

La Tabella 3.2 presenta, con un arrotondamento all'intero più vicino, i costi computazionali espressi in secondi per la compilazione delle funzioni relative a ciascun metodo, applicati alla serie storica multistagionale.

Come si può notare il metodo BATS è il meno efficiente, sia dal punto di vista dell'errore RMSE, sia dal punto di vista computazionale poiché la compilazione della funzione con stagionalità multipla utilizzando questo metodo ha richiesto tre giornate.

Tabella 3.2: Costi computazionali

Metodi	Tempo totale in secondi
MSTL	4
STR	1800
TBATS	43200
BATS	259200

Discussione dei risultati

Come accennato in precedenza, l'analisi dell'RMSE per le componenti di trend e stagionalità annuale hanno mostrato che il modello stimato con STR presenta un valore di RMSE simile a quello di MSTL (Tabella 3.1). Ciò è dovuto al fatto che la linea di trend trovata con il metodo STR, visibile in Figura 2.12, non presenta picchi improvvisi, ma segue un andamento costante e uniforme. Infatti, se si confrontano le componenti stagionali annuali dei due modelli dal punto di vista grafico, si osserva che hanno uno sviluppo molto simile.

Tuttavia, quando si analizzano i valori di RMSE relativi alle stagionalità giornaliera, settimanale o dei residui, il modello STR presenta valori molto più elevati. Inoltre, sempre nella Tabella 3.1, si evidenzia come il metodo MSTL abbia anche un RMSE più basso rispetto ai modelli BATS e TBATS, confermando la sua superiorità in termini di precisione statistica.

I metodi BATS e TBATS presentano nell'errore quadratico medio relativo alla componente di *trend* valori molto elevati; la spiegazione viene fornita dai grafici in Figura 2.6, relativo alla decomposizione con il metodo BATS e in Figura 2.9 che mostra la decomposizione con il metodo TBATS. Si può osservare che il pannello relativo al livello in entrambi i grafici, cioè il valore medio dei dati per tutta la durata della serie storica, non è liscio e omogeo come per il grafico di MSTL in Figura 2.4, ma ha molti picchi, e questo porta a calcolare dei valori adattati che sono peggiori e di conseguenza ad un RMSE più elevato.

La Tabella 3.2 fornisce ulteriori informazioni sulle prestazioni dei diversi metodi, evidenziando che il metodo MSTL è il migliore anche in termini di efficienza computazionale. I risultati mostrano come MSTL sia stato in grado di completare le operazioni in un tempo notevolmente

inferiore comparato agli altri metodi.

In sintesi, la presente analisi ha fornito un quadro chiaro della superiorità di MSTL in confronto agli altri modelli in termini di precisione statistica e di *performance* computazionale. Questi risultati indicano come l'utilizzo del metodo MSTL sia la scelta ottimale per questo tipo di applicazioni.

Conclusioni

L'analisi svolta sui dati Terna - Driving Energy ha confermato le aspettative: dopo aver analizzato tutti i metodi, MSTL risulta essere il più efficiente sotto ogni punto di vista. I costi computazionali sono irrisori e le misure degli errori molto più basse rispetto a tutti gli altri modelli.

STR potrebbe essere competitivo per MSTL relativamente ai tempi di esecuzione a livello computazionale, ma per quanto riguarda l'RMSE è molto più alto in diverse componenti del modello stimato.

Il metodo BATS, invece, risulta essere il meno performante sia dal punto di vista dell'errore RMSE che dal punto di vista computazionale. La compilazione della funzione con stagionalità multipla utilizzando questo metodo ha richiesto ben tre giorni, il che rappresenta un tempo estremamente elevato. Questo porta ad utilizzare questo metodo come ultima risorsa o addirittura scartarlo per la sua poca affidabilità.

TBATS, per quanto ci impieghi meno nella compilazione, non ha prodotto risultati soddisfacenti in quanto a precisione.

In conclusione, l'analisi effettuata ha dimostrato con chiarezza l'eccellente prestazione del metodo MSTL rispetto agli altri metodi esaminati, sia per quanto riguarda la precisione statistica che la performance computazionale. Questi risultati indicano senza ombra di dubbio come l'utilizzo del metodo MSTL sia la soluzione ideale per questo tipo di applicazioni. In altre parole, l'utilizzo di MSTL rappresenta la scelta

ottimale per garantire risultati affidabili e precisi, in tempi ragionevoli.

Appendice A

A.1 Comandi R analisi

A.1.1 Caricamento dati, pulizia e aggregazione oraria

```
#caricamento dati
library(readxl)
terna20 <- read_excel("Desktop/terna20.xlsx",
                      col_types = c("date", "numeric", "numeric",
                                    "text"))

library(readxl)
terna21 <- read_excel("Desktop/terna21.xlsx",
                      col_types = c("date", "numeric", "numeric",
                                    "text"))

library(readxl)
terna22 <- read_excel("Desktop/terna22.xlsx",
                      col_types = c("date", "numeric", "numeric",
                                    "text"))
#unione dei dati
ternatotale <- rbind(terna20, terna21, terna22)
#eliminazione omit
terna <- na.omit(ternatotale)
#creazione dataframe
terna <- data.frame(dataeora =terna$Date,
                    totalload=terna$`Total Load [MW]`,
                    fortotload =terna$`Forecast Total load [MW]`)

#dati aggregati da 15 minuti a orari
library("lubridate")
library("dplyr")
terna = terna %>%
  mutate(aggregato = floor_date(terna$dataeora, unit = "hour")) %>%
  group_by(aggregato) %>%
  summarise(loadtot = mean(totalload, na.rm = TRUE))

#funzione per creare l'oggetto mstl
```

```
totale2 <- msts(terna$loadtot,seasonal.periods = c(24, 168, 8766) )

plot(terna$aggregato, terna$loadtot, type = "l")
```

A.1.2 Modello MSTL

```
#funzione per creare l'oggetto mstl
totale2 <- msts(terna$loadtot,seasonal.periods = c(24, 168, 8766))

plot(terna$aggregato, terna$loadtot, type = "l")

#analisi con il metodo MSTL
library(forecast)
#funzione per creare l'oggetto classe mstl
mstltotale2<- mstl(totale2, s.window = "periodic")
plotternamstl = autoplot(mstltotale2, main = "MSTL")
library(fabletools)
#presa ogni colonna dell'oggetto mstl appena creato,
#applicata la funzione forecast e trovato l'RMSE con
#il comando accuracy() all'interno del pacchetto fabletools

#RMSE trend
trend2 <- mstltotale2[,2]
head(trend2)
trendforecast2 <- forecast(trend2)
accuracy(trendforecast2$fitted, trend2)

#RMSE giornaliero
totale24_2 <- mstltotale2[,3]
head(totale24)
totale24forecast2 <- forecast(totale24_2)
accuracy(totale24forecast2$fitted, totale24_2)#RMSE giornaliero
plot(totale24forecast2$residuals)

#RMSE settimanale
totale168_2 <- mstltotale2[,4]
totale168forecast2 <- forecast(totale168_2)
accuracy(totale168forecast2$fitted, totale168_2) #RMSE settimanale
plot(totale168forecast2$residuals)

#RMSE annuale
totale8766_2 <- mstltotale2[,5]
totale8766forecast2 <- forecast(totale8766_2)
accuracy(totale8766forecast2$fitted, totale8766_2) #annuale
plot(totale8766forecast2$residuals)

#RMSE residui
totaleremainder2 <- mstltotale2[,6]
totaleremainderforecast2 <- forecast(totaleremainder2)
accuracy(totaleremainderforecast2$fitted, totaleremainder2)
plot(totaleremainderforecast2$residuals)

#ACF e PACF
par(mfrow = c(2,1))
acf(totaleremainder2, 150, main = "ACF residui modello MSTL")
pacf(totaleremainder2, 150, main = "PACF residui modello MSTL")
```

A.1.3 Modello BATS

```
#analisi per il metodo BATS
#funzione per creare l'oggetto bats
ternabatstotale2 <- bats(totale2, use.parallel = TRUE, use.box.cox = TRUE)
library(generics)
library(fable)
library(generics)
head(tbats.components(ternabatstotale))
plot(ternabatstotale)
View(ternabatstotale)

#RMSE trend
batstrend <- forecast(tbats.components(ternabatstotale)[,'level'])
accuracy(batstrend)

#RMSE giornaliero
bats24 <- forecast(tbats.components(ternabatstotale)[,'season1'])
accuracy(bats24)

#RMSE settimanale
bats168 <- forecast(tbats.components(ternabatstotale)[,'season2'])
accuracy(bats168$fitted.values, )

#RMSE annuale
bats8766 <- forecast(tbats.components(ternabatstotale)[,'season3'])
accuracy(bats8766)

#RMSE residui
accuracy(forecast(ternabatstotale$errors))


#ACF e PACF
par(mfrow = c(2,1))
acf(ternabatstotale$errors, 150, main = "ACF residui modello BATS")
pacf(ternabatstotale$errors, 150, main = "PACF residui modello BATS")
```

A.1.4 Modello TBATS

```
#analisi con il metodo TBATS
library(forecast)
library(generics)
#funzione per creare l'oggetto classe tbats
tbatstotale2 <- tbats(totale2, use.parallel = TRUE)
head(tbats.components(tbatstotale2))
accuracy(tbatstotale)
plot(tbatstotale2)
View(tbatstotale2)

#RMSE trend
leveltbats2 <- forecast(tbats.components(tbatstotale2)[,'level'])
accuracy(leveltbats2$fitted,tbats.components(tbatstotale2)[,'level'])

#RMSE giornaliero
season24tbats2 <- forecast(tbats.components(tbatstotale2)[,'season1'])
```

```
accuracy(season24tbats$fitted,tbats.components(tbatstotale2)[,'season1'])

#RMSE settimanale
season168tbats2 <- forecast(tbats.components(tbatstotale2)[,'season2'])
accuracy(season168tbats, tbats.components(tbatstotale2)[,'season2'])

#RMSE annuale
season8766tbats2 <- forecast(tbats.components(tbatstotale2)[,'season3'])
accuracy(season720tbats, tbats.components(tbatstotale2)[,'season3'])

#RMSE residui
plot(forecast(tbatstotale2$errors))
accuracy(forecast(tbatstotale2$errors))
plot(tbatstotale2$errors)

#ACF e PACF
par(mfrow = c(2,1))
acf(tbatstotale2$errors, 150, main = "ACF residui modello TBATS")
pacf(tbatstotale2$errors, 150, main = "PACF residui modello TBATS")
```

A.1.5 Modello STR

```
#analisi con il metodo AutoSTR
library(stR)
library(graphics)
library(generics)
library(xts)
#funzione per creare l'oggetto classe STR
ternastr2 <- AutoSTR(totale2, gapCV = 24)
plot(ternastr2)
library(graphics)

head(components(ternastr2))
ternastrtrend2 <- components(ternastr2)[,"Trend"]
head(ternastrtrend2)
ternastrtrendfore2 <- forecast(ternastrtrend2)
#RMSE trend
accuracy(ternastrtrendfore2$fitted, components(ternastr2)[,"Trend"])


ternastr24_2 <- components(ternastr2)[,'Seasonality 24']
head(ternastr24)
ternastr24fore2 <- forecast(ternastr24_2)
#RMSE giornaliero
accuracy(ternastr24fore2$fitted, components(ternastr2)[,'Seasonality 24'])


ternastr168_2 <- components(ternastr2)[,'Seasonality 168']
head(ternastr168_2)
ternastr168fore2 <- forecast(ternastr168_2)
#RMSE settimanale
accuracy(ternastr168fore2$fitted, components(ternastr2)[,'Seasonality 168'])

ternastr8766_2 <- components(ternastr2)[,'Seasonality 8766']
head(ternastr8766_2)
ternastr8766fore2 <- forecast(ternastr8766_2)
#RMSE annuale
```

```
accuracy(ternastr8766fore2$fitted, components(ternastr2)[,'Seasonality 8766'])

ternastrrandom2 <- components(ternastr2)[,'Random']
head(ternastrrandom2)
ternastrrandomfore2 <- forecast(ternastrrandom2)

#RMSE residui del fit
accuracy(ternastrrandomfore2$fitted, components(ternastr2)[,'Random'])

#ACF e PACF
par(mfrow = c(2,1))
acf(ternastrrandom2, 150, main = "ACF residui modello STR")
pacf(ternastrrandom2, 150, main = "PACF residui modello STR")
```

Bibliografia e sitografia

[1] A. Dokumentov, R.J. Hyndman. "STR: Seasonal-trend decomposition using regression." arXiv preprint arXiv:2009.05894 (2020).

[2] Bandara, Kasun, R.J. Hyndman, and C. Bergmeir. "MSTL: a seasonal-trend decomposition algorithm for time series with multiple seasonal patterns." arXiv preprint arXiv:2107.13462 (2021).

[3] C. Bergmeir, R.J. Hyndman, J.M. Benítez. "Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation." International journal of forecasting 32.2 (2016): 303-312.

[4] De Livera, Alysha M., R.J. Hyndman, and R.D. Snyder. "Forecasting time series with complex seasonal patterns using exponential smoothing." Journal of the American statistical association 106.496 (2011): 1513-1527.

[5] I.Naim, T. Mahara, A.R. Idrisi. "Effective short-term forecasting for daily time series with complex seasonal patterns." Procedia computer science 132 (2018): 1832-1841.

[6] Modello TBATS per l'analisi di dati settimanali https://robjhyndman.com/hyndsight/forecasting-weekly-data/#tbats

[7] Pacchetto *forecast* https://cran.r-project.org/web/packages/forecast/forecast.pdf

[8] Pacchetto *STR* https://cran.r-project.org/web/packages/stR/stR.pdf

[9] R.B. Cleveland, W.S. Cleveland, J.E McRae, I.Terpenning. "STL: A seasonal-trend decomposition." J. Off. Stat 6.1 (1990): 3-73.

[10] R.J. Hyndman, Y. Khandakar. "Automatic time series forecasting: the forecast package for R." Journal of statistical software 27 (2008): 1-22.

[11] Sito Terna per il dowload dei dati https://www.terna.it/it/sistema-elettrico/transparency-report/download-centerx

[12] T. Proietti, D.J. Pedregal. "Seasonality in high frequency time series." Econometrics and Statistics (2022).