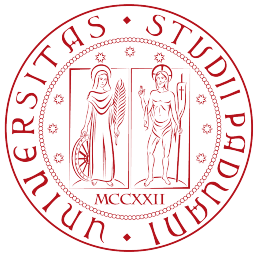


Università degli Studi di Padova

Dipartimento di Scienze Statistiche

Corso di Laurea Magistrale in
Scienze Statistiche

Riconciliazione contemporanea e temporale di previsioni di serie storiche in Python

Relatore:
Prof. Tommaso Di Fonzo

Laureando:
Alberto Maronilli
1236610

Anno Accademico 2021/2022

Indice

Capitolo 1

Introduzione

La previsione accurata di variabili macroeconomiche come il prodotto interno lordo, l'inflazione o la produzione industriale, è da sempre una delle tematiche più importanti nell'ambito della ricerca statistica. Generalmente, in questi contesti, che possono appartenere ad ambiti anche diversi da quello economico, la strada più percorsa per ottenere delle previsioni accurate consiste nel fare ricorso ad una grande varietà di modelli che utilizzano un vasto numero di predittori per la previsione di tali variabili.

Per molte di queste osservazioni è possibile anche percorrere una strada diversa: in molteplici casi la variabile di interesse può essere ottenuta anche come somma di altre variabili che a loro volta possono essere nuovamente disaggregate, ottenendo in questo modo delle particolari strutture gerarchiche, che in generale dovranno rispettare dei vincoli di somma. In questo senso, negli ultimi anni la letteratura scientifica ha conosciuto una vasta crescita nella trattazione di tecniche per aggiustare previsioni di serie storiche in modo da rispettare dei particolari vincoli di aggregazione gerarchica. Questi metodi, pur essendo stati sviluppati con l'intento iniziale di fornire delle previsioni allineate per garantire dei processi decisionali coerenti, si sono poi dimostrati anche capaci di incrementare significativamente l'accuratezza delle previsioni.

La tipologia di struttura gerarchica più approfondita in letteratura è quella detta contemporanea o cross sezionale che, ad ogni fissato istante di tempo, valuta la coerenza della gerarchia per tutte le variabili che appar-

tengono ad essa. È stato definito però anche un diverso tipo di gerarchie che possono anche essere ottenute a partire da una qualsiasi serie storica univariata ad una cadenza temporale fissa: aggregando temporalmente i valori delle serie storiche è possibile definire le cosiddette gerarchie temporali per le quali potrà essere valutata la coerenza dei livelli di aggregazione alle diverse cadenze temporali.

Soprattutto all'interno del linguaggio di programmazione R, celebre per il suo impiego in ambito statistico, con il passare degli anni sono stati implementati numerosi strumenti per permettere agli utenti di applicare alle previsioni di serie storiche diverse tecniche di riconciliazione. In particolare, nel 2020 è stato sviluppato il pacchetto FoReco (Girolimetto e Di Fonzo, 2022 [13]) che contiene al suo interno tutte le principali tecniche di riconciliazione contemporanea e temporale più note al giorno d'oggi.

Python è un linguaggio di programmazione sempre più utilizzato per l'analisi statistica. Però allo stato attuale tale linguaggio non dispone ancora di implementazioni complete nell'ambito della riconciliazione contemporanea e temporale di previsioni di serie storiche. Per questo motivo nel presente lavoro si è voluto sviluppare un pacchetto di funzioni che permetta l'applicazione delle tecniche di riconciliazione anche in ambiente Python, basandosi in particolare sul pacchetto FoReco.

Un altro fattore da considerare durante l'applicazione delle procedure di riconciliazione è la richiesta di elevate risorse computazionali per le gerarchie di più grandi dimensioni. Per questo motivo si è anche voluto valutare se il linguaggio Python, che generalmente risulta più efficiente di R, possa fornire un incremento in termini di prestazioni.

Il lavoro è strutturato come segue:

- Nel capitolo 2 viene definito il concetto di gerarchia contemporanea e, dopo aver stabilito la notazione, vengono presentati i diversi metodi di riconciliazione per le previsioni noti in letteratura: quelli classici, come il bottom up e il top down, e quelli moderni, basati sulla riconciliazione ottimale puntuale tramite modello di regressione lineare.

- Nel capitolo 3 viene presentato il concetto di gerarchia temporale,

formalizzando questo problema, e sono poi presentate tecniche di riconciliazione delle previsioni sviluppate in questo contesto.

- Nel capitolo 4 vengono presentati i principali strumenti disponibili su Internet per l'applicazione delle procedure di riconciliazione esposte precedentemente con particolare attenzione per il pacchetto FoReco.
- Nel capitolo 5, dopo il richiamo al linguaggio Python, viene introdotta la versione per questo linguaggio del pacchetto FoReco.
- Il capitolo 6 è dedicato all'analisi delle performance del pacchetto FoReco in Python rispetto alla sua controparte in R.
- Nel capitolo 7 viene presentata un'applicazione di FoReco in Python alla previsione dei flussi turistici australiani disaggregati per destinazione geografica e motivo del viaggio.

Capitolo 2

Serie Storiche Gerarchiche e raggruppate

Le serie storiche temporali possono essere disaggregate in base a vari attributi di interesse, che possono assumere anche una natura gerarchica. Uno degli esempi più semplici che si possono fare in questo contesto fa riferimento alla posizione geografica: i profitti totali di una multinazionale possono essere disaggregati nei profitti provenienti dai diversi paesi ed eventualmente ancora in quelli provenienti dai diversi punti vendita, a seconda che si sia interessati a una visione d'insieme della variabile o a sue caratteristiche più specifiche.

È possibile anche raggruppare le diverse variabili di base senza fare riferimento ad una struttura gerarchica univoca. Sempre ricorrendo all'esempio della multinazionale, si potrebbero dividere i profitti in base alla tipologia di prodotti venduti oltre che alla posizione geografica. Una struttura che considera contemporaneamente più di una singola gerarchia è detta raggruppata.

In questo contesto è possibile individuare quindi una struttura composta da diverse serie storiche vincolate linearmente tra loro per ogni componente a livello contemporaneo, generalmente tramite semplici somme.

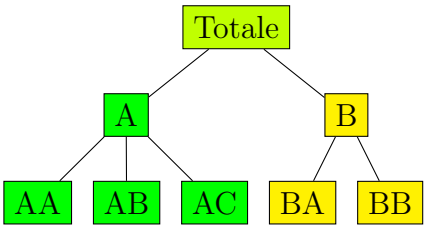


Figura 2.1: Diagramma ad albero di una struttura gerarchica contemporanea con 2 livelli

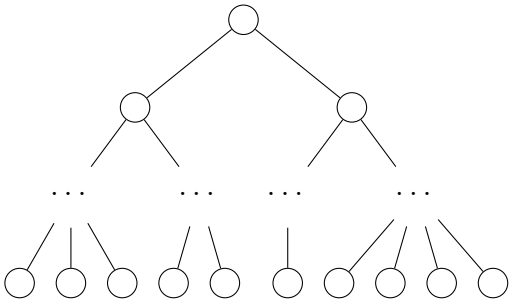


Figura 2.2: Diagramma ad albero di una generica gerarchia contemporanea

2.1 Notazione

Per introdurre la notazione da utilizzare nel contesto delle serie storiche gerarchiche facciamo ricorso al *diagramma ad albero*, uno strumento comodo per visualizzare una gerarchia quando si ha un numero ridotto di serie storiche.

Ad esempio, il diagramma ad albero in figura 2.1 mostra una struttura gerarchica a 2 livelli, all'interno della quale nel nodo più alto possiamo trovare il totale, cioè il livello più aggregato dei dati. Il totale è disaggregato in due serie al livello 1, che a loro volta sono divise rispettivamente in tre e due serie al livello più basso della gerarchia.

In questo esempio, il numero totale di serie nella gerarchia è $n = 8$, mentre il numero di serie del livello più basso n_b, chiamate *bottom time series* è pari a 5. Indichiamo poi con n_a il numero di serie date dall'aggregazione delle di altre serie, chiamate *upper time series* e in questo caso pari a $n - n_b = 3$. Facendo riferimento alla figura 2.2 in generale indichiamo con L_0 il livello più in alto relativo alla massima sintesi che si può ottenere. Questo livello conterrà sempre una sola serie storica, nell'esempio precedente "Totale", pari alla somma di tutte le serie del livello successivo.

I livelli L_j con $j = 1, 2, \ldots, (K-1)$ sono i vari livelli che indicano le classi intermedie della gerarchia, con una granularità di volta in volta più raffinata all'aumentare del numero del livello, l'unico livello intermedio nell'esempio precedente è L_1 contenente i nodi A e B.

L_K, infine, indica il livello più basso della gerarchia e contiene le *bottom time series*, cioè tutte quelle variabili che non possono essere ottenute come somma di altre.

Utilizziamo $y_{j,T}$ per denotare la t-esima osservazione della serie corrispondente al nodo j. Per esempio, in figura 2.1, $y_{A,t}$ denota la T-esima osservazione della serie corrispondente al nodo A del livello 1.

Ad ogni tempo t è possibile esprimere le *upper time series* come somma di *bottom time series* tramite un sistema di equazioni. Sempre facendo riferimento all'esempio in figura 2.1 il sistema prende questa forma:

$$\begin{cases} y_{Tot,t} = y_{AA,t} + y_{AB,t} + y_{AC,t} + y_{BA,t} + y_{BB,t} \\ y_{A,t} = y_{AA,t} + y_{AB,t} + y_{AC,t} \\ y_{B,t} = y_{BA,t} + y_{BB,t} \end{cases} . \tag{2.1}$$

2.1.1 Rappresentazione Strutturale

In generale, per esprimere i sistemi che definiscono le strutture gerarchiche e le eventuali operazioni che saranno svolte su di esse, risulta molto comodo fare ricorso ad una notazione matriciale. Per questo motivo viene presentata la *rappresentazione strutturale di una serie storica gerarchica*, introdotta da Hyndman et al. (2011 [20]).

Fissato un tempo t, indichiamo con y_t il vettore n-dimensionale contenente tutte le osservazioni a ogni livello della struttura gerarchica. Il vettore y_t può essere diviso in modo tale che $\mathbf{y}_t = [\mathbf{a}_t' \quad \mathbf{b}_t']'$ dove $\mathbf{a}_t = [a_{1,t} \ldots a_{j,t} \ldots a_{n_a,t}]'$ vettore $(n_a \times 1)$ e contenente le *upper time series* e $\mathbf{b}_t = [b_{1,t} \ldots b_{i,t} \ldots b_{n_b,t}]'$ vettore $(n_b \times 1)$ contenente le *bottom time series*.

Per esprimere i vincoli di aggregazione cross-sezionale è possibile fare ricorso ad una matrice $(n \times n_b)$ di somma contemporanea (*contemporaneous summing matrix*) $\mathbf{S}$, che ci permette di rappresentare la struttura gerarchica

come

$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t, \quad t = 1, \ldots, T \tag{2.2}$$

La matrice $\mathbf{S}$ può essere partizionata in due diversi blocchi, il primo contenente la matrice di aggregazione contemporanea $\mathbf{C}$ che andrà a sommare le *bottom time series* nelle *upper time series*, mentre il secondo è dato da una matrice identità $(n_b \times n_b)$. Quindi

$$\mathbf{S} = \begin{bmatrix} \mathbf{C}_{n_a \times n_b} \\ \mathbf{I}_{n_b \times n_b} \end{bmatrix} \tag{2.3}$$

e

$$\mathbf{a}_t = \mathbf{C}\mathbf{b}_t, \quad t = 1, \ldots, T. \tag{2.4}$$

È possibile esprimere i vincoli che caratterizzano le strutture gerarchiche anche tramite un sistema lineare omogeneo:

$$\mathbf{U}'\mathbf{y}_t = 0, \quad t = 1, \ldots, T, \tag{2.5}$$

dove 0 è un vettore di zeri di dimensione $(n_a \times 1)$ e $\mathbf{U}'$ è una matrice $(n_a \times n)$ che può assumere la forma:

$$\mathbf{U}' = \begin{bmatrix} \mathbf{I}_{n_a \times n_a} & -\mathbf{C}_{n_a \times n_b} \end{bmatrix}. \tag{2.6}$$

Per chiarire la natura di queste matrici, riprendiamo l'esempio del sistema 2.1. La matrice $\mathbf{S}$ ha la forma:

$$\mathbf{S} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{3 \times 5} \\ \mathbf{I}_5 \end{bmatrix}$$

La rappresentazione strutturale tramite l'equazione 2.2 diventa:

$$\begin{bmatrix} y_{Tot,t} \\ y_{A,t} \\ y_{B,t} \\ y_{AA,t} \\ y_{AB,t} \\ y_{AC,t} \\ y_{BA,t} \\ y_{BB,t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{AA,t} \\ y_{AB,t} \\ y_{AC,t} \\ y_{BA,t} \\ y_{BB,t} \end{bmatrix}.$$

Per il sistema lineare omogeneo 2.5, invece, otteniamo:

$$\begin{bmatrix} 1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 0 & 1 & 0 & -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} y_{Tot,t} \\ y_{A,t} \\ y_{B,t} \\ y_{AA,t} \\ y_{AB,t} \\ y_{AC,t} \\ y_{BA,t} \\ y_{BB,t} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Dal momento che i vincoli di aggregazione della struttura gerarchica sono validi ad ogni tempo t, è possibile esprimere le equazioni 2.2, 2.4 e 2.5 in forma compatta tramite la definizione delle due seguenti matrici:

$$\mathbf{B}_{(n_b \times T)} = \begin{bmatrix} b_{11} & \dots & b_{1t} & \dots & b_{1T} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ b_{i1} & \dots & b_{it} & \dots & b_{iT} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ b_{n_b 1} & \dots & b_{n_b t} & \dots & b_{n_b T} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_t & \dots & \mathbf{b}_T \end{bmatrix},$$

$$\mathbf{A}_{(n_a \times T)} = \begin{bmatrix} a_{11} & \dots & a_{1t} & \dots & a_{1T} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \dots & a_{it} & \dots & a_{iT} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n_a 1} & \dots & a_{n_a t} & \dots & a_{n_a T} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_t & \dots & \mathbf{a}_T \end{bmatrix},$$

$$\mathbf{Y}_{(n \times T)} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$$

che contengono rispettivamente le *bottom time series*, le *upper time series* e tutte le osservazioni della serie storica gerarchica ad ogni istante di tempo.

Per esprimere i vincoli forniti dalla struttura gerarchica ad ogni tempo possiamo, quindi, usare le formule:

$$\mathbf{Y} = \mathbf{SB}, \tag{2.7}$$

$$\mathbf{A} = \mathbf{CB}, \tag{2.8}$$

e

$$\mathbf{U}'\mathbf{Y} = 0_{n_a \times T}. \tag{2.9}$$

Per quanto riguarda le serie raggruppate, anche se non è possibile rappresentarle con un unico diagramma ad albero, un'opportuna definizione della matrice S ci permette di indicare tutti i vincoli lineari tramite sempre le formule 2.2, 2.4 e 2.5. Va tenuto presente che in questo tipo di gerarchie, i diversi diagrammi ad albero che le definiscono hanno in comune il livello più alto e le bottom time series, come è possibile vedere dall'esempio in figura 2.3.

In questo caso definendo i vettori delle *upper time series* e delle *bottom*

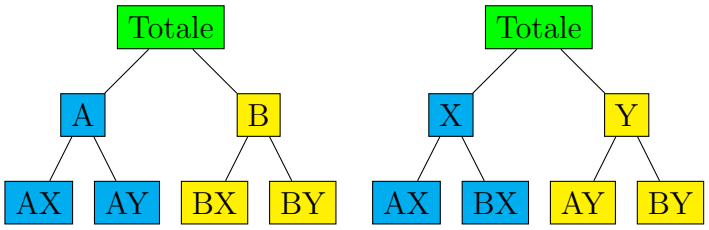


Figura 2.3: Rappresentazioni alternative di una gerarchia raggruppata a due livelli

time series come, rispettivamente

$$\mathbf{a} = \begin{bmatrix} y_{Tot,t} \\ y_{A,t} \\ y_{B,t} \\ y_{X,t} \\ y_{Y,t} \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{bmatrix}$$

è possibile esprimere tutte le relazioni della gerarchia raggruppata tramite la formula 2.2 in modo tale che:

$$\begin{bmatrix} y_{Tot,t} \\ y_{A,t} \\ y_{B,t} \\ y_{X,t} \\ y_{Y,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{bmatrix}.$$

2.2 Riconciliazione contemporanea di previsioni puntuali

Per ottenere le previsioni delle serie gerarchiche contemporanee è possibile ricorrere alle tecniche note nel campo dell'analisi delle serie storiche. Per esempio, in presenza di autocorrelazione nei dati, si può far ricorso ai mo-

delli ARIMA (Box e Jenkins, 1970 [5]), se si è a conoscenza di stagionalità o altre informazioni a priori sulle caratteristiche delle serie si può utilizzare la tecnica del lisciamento esponenziale ETS (Hyndman et al., 2002 [19]), in caso di correlazione tra le diverse serie si può far ricorso ai modelli VAR (Lütkepohl, 2015 [25]), altrimenti si possono utilizzare tecniche più complesse basate sul *machine learning* (Cheng et al., 2012 [7], Spiliotis et al., 2020 [37]). In queste circostanze nella quasi totalità dei casi quello che risulta è una previsione della struttura gerarchica nella quale i vincoli di aggregazione non sono rispettati. Per risolvere questo problema è possibile percorrere due diverse strade:

- Introdurre dei vincoli durante la procedura di modellazione per stimare delle previsioni.

- Affidarsi a delle tecniche di *post-forecasting* per far rispettare i vincoli aggiustando i valori delle previsioni di base.

Il ricorso a dei modelli che utilizzano vincoli, oltre ad essere molto onerosi dal punto di vista computazionale, non è sempre possibile dal momento che per serie storiche gerarchiche è spesso appropriato utilizzare tecniche diverse per la stima delle previsioni a diversi livelli di aggregazione. In generale quindi è preferibile far ricorso a delle tecniche di *post-forecasting*.

Indichiamo con T il numero di osservazioni per ciascuna serie, e con H l'orizzonte di previsione. Indichiamo con $\hat{\mathbf{y}}_h = \hat{\mathbf{y}}_T(h) = E[\mathbf{y}_{T+h}|I_T]$ le n previsioni con orizzonte di previsione h, dove I_T è l'informazione disponibile fino al tempo T. Definiamo con $\hat{\mathbf{A}}$ la matrice $(n_a \times H)$ che contiene tutte le previsioni di base dei livelli aggregati, con $\hat{\mathbf{B}}$ la matrice $(n_b \times H)$ contenente le previsioni relative alle *bottom time series* e con $\hat{\mathbf{Y}}$ la matrice $(n \times H)$ contenente le previsioni per l'intera gerarchia:

$$\hat{\mathbf{Y}}_{(n\times H)} = \begin{bmatrix} \hat{\mathbf{A}} \\ \hat{\mathbf{B}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{y}}_1 & \cdots & \hat{\mathbf{y}}_h & \cdots & \hat{\mathbf{y}}_H \end{bmatrix},$$

$$\hat{\mathbf{B}}_{(n_b \times H)} = \begin{bmatrix} \hat{b}_{11} & \dots & \hat{b}_{1h} & \dots & \hat{b}_{1H} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \hat{b}_{i1} & \dots & \hat{b}_{ih} & \dots & \hat{b}_{iH} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \hat{b}_{n_b 1} & \dots & \hat{b}_{n_b h} & \dots & \hat{b}_{n_b H} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{b}}_1 & \dots & \hat{\mathbf{b}}_h & \dots & \hat{\mathbf{b}}_H \end{bmatrix},$$

$$\hat{\mathbf{A}}_{(n_a \times H)} = \begin{bmatrix} \hat{a}_{11} & \dots & \hat{a}_{1h} & \dots & \hat{a}_{1H} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \hat{a}_{i1} & \dots & \hat{a}_{ih} & \dots & \hat{a}_{iH} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \hat{a}_{n_a 1} & \dots & \hat{a}_{n_a h} & \dots & \hat{a}_{n_a H} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{a}}_1 & \dots & \hat{\mathbf{a}}_h & \dots & \hat{\mathbf{a}}_H \end{bmatrix}.$$

Indichiamo, infine, con $\tilde{\mathbf{y}}_h$ il generico vettore delle previsioni riconciliate che rispettano i vincoli della struttura gerarchica al tempo h, e con $\tilde{\mathbf{Y}}$ la matrice $(n \times H)$ che contiene le previsioni riconciliate ad ogni orizzonte di previsione.

2.2.1 Metodi tradizionali di riconciliazione

Bottom-up

Un metodo semplice per generare previsioni coerenti è il metodo *bottom-up* (Orcutt et al., 1968 [30]), il quale fa uso delle sole *bottom time series* per ottenere le previsioni riconciliate dell'intera gerarchia. Utilizzando la notazione definita in sezione 2.1, le previsioni riconciliate utilizzando la procedura *bottom-up* sono calcolate premoltiplicando il vettore delle previsioni di base al livello *bottom* con la matrice di somma contemporanea:

$$\tilde{\mathbf{Y}} = \mathbf{S}\hat{\mathbf{B}}. \tag{2.10}$$

Un vantaggio di questo approccio è l'utilizzo delle previsioni più disaggregate in modo tale che nessuna informazione venga persa a causa dell'aggregazione. Tuttavia, vengono ignorate completamente le relazioni tra le serie, aspetto invece fondamentale di queste strutture. Inoltre, le previsioni

ottenute dalle *bottom time series* spesso risultano essere molto variabili e di scarsa qualità, a causa della notevole variabilità che caratterizza le serie 'più piccole'.

Top-down

L'approccio *top down* (Gross e Sohl, 1990 [14]) può essere visto come l'opposto di quanto accade nell'approccio *bottom-up*: vengono calcolate le previsioni relative solamente al livello più alto, che poi sono disaggregate tramite un vettore di proporzioni $\mathbf{p}_h = \begin{bmatrix} p_1 & \cdots & p_{n_b} \end{bmatrix}'$.

Quindi, fissato l'orizzionte di previsione h, possiamo scrivere il vettore delle previsioni riconciliate per il livello disaggregato come

$$\tilde{\mathbf{b}}_h = \mathbf{p}_h \hat{a}_{1,h}$$

e le previsioni riconciliate

$$\tilde{\mathbf{y}}_h = \mathbf{S}\mathbf{p}_h \tilde{a}_{1,h}. \tag{2.11}$$

Un ruolo fondamentale è ricoperto dalle proporzioni $\mathbf{p}_h$. In letteratura sono presenti tre principali tecniche per calcolarle:

- La media delle quote calcolate per ogni tempo disponibile (Gross e Sohl, 1990 [14]);
- Le quote calcolate utilizzando la media delle bottom time series e la media del totale (Gross e Sohl, 1990 [14]);
- Le quote costruite per ogni orizzonte temporale h a partire dalle previsioni di tutte le serie presenti nella gerarchia all'orizzonte temporale h (Athanasopoulos et al, 2009 [2]).

La tecnica *top-down* permette di produrre previsioni accurate anche quando le *bottom time series* presentano caratteristiche "scomode", come dati mancanti o un elevato numero di valori nulli.

È stato però dimostrato (Hyndman et al. 2011 [20]) che l'utilizzo dell'approccio *top-down* dà luogo a previsioni distorte per tutti i livelli della

gerarchia, anche se le previsioni di partenza non sono distorte. Inoltre, utilizzando le prime due tecniche presentate per ottenere il vettore di proporzioni, vengono perse informazioni importanti sulle diverse serie individuali, come stagionalità o altre particolari dinamiche temporali.

Middle-out

L'approccio *middle-out* (Athanasopoulos et al, 2009 [2]) risulta essere un compromesso tra gli approcci *bottom-up* e *top-down*. In questo caso, viene scelto un livello intermedio della struttura gerarchica per il quale sono ottenute delle previsioni; successivamente vengono ottenuti i livelli superiori tramite l'approccio *bottom-up* e quelli inferiori tramite l'approccio *top-down*. Seppure l'idea del *middle out* riduce gli svantaggi delle tecniche precedenti, essa tuttavia non li elimina completamente.

In conclusione, le tre tecniche presentate finora, pur risultando molto semplici da applicare, presentano delle importanti debolezze:

- sfruttano l'informazione solo di una parte delle serie disponibili;
- non considerano la struttura di correlazione della gerarchia, ignorando quindi la correlazione a ciascun livello;
- non permettono di ricavare facilmente degli intervalli di previsione.

2.2.2 Mapping e projection matrix

Volendo utilizzare tutte le osservazioni della struttura gerarchica per calcolare le previsioni riconciliate, invece che solamente una porzione di queste, risulta particolarmente comodo ampliare la notazione introducendo la *mapping matrix*, chiamata in questo modo perché "mappa" le previsioni di base nelle bottom time series riconciliate.

Tramite questa matrice, che indicheremo con $\mathbf{G}_h$, il generico vettore di previsioni riconciliate può essere scritto come

$$\tilde{\mathbf{y}}_h = \mathbf{S}\mathbf{G}_h\hat{\mathbf{y}}_h, \quad h = 1, \dots, H, \tag{2.12}$$

Da cui si deduce che $\tilde{\mathbf{b}}_h = \mathbf{G}_h\hat{\mathbf{y}}_h$. Nel caso in cui $\mathbf{G}_h = \mathbf{G}$, $h = 1, \ldots, H$, l'espressione 2.12 può essere espressa in forma compatta:

$$\tilde{\mathbf{Y}} = \mathbf{SG}\hat{\mathbf{Y}}. \tag{2.13}$$

Si può mostrare che entrambe le tecniche *bottom up* e *top down* possono essere espresse utilizzando delle *mapping matrix*. In particolare, se nella formula 2.13 si pone $\mathbf{G} = \begin{bmatrix} 0_{n_b \times n_a} & \mathbf{I}_{n_b} \end{bmatrix}$ si ottiene la riconciliazione *bottom up* come in formula 2.10, mentre se nella formula 2.12 si pone $\mathbf{G}_h = \begin{bmatrix} \mathbf{p}_h & 0_{n_b \times n-1} \end{bmatrix}$ si ottiene la riconciliazione top down come in formula 2.11.

Definiamo $\mathbf{M} = \mathbf{SG}$, allora l'equazione 2.13 diventa:

$$\tilde{\mathbf{Y}} = \mathbf{M}\hat{\mathbf{Y}}, \tag{2.14}$$

dove $\mathbf{M}$ è una matrice di proiezione (*projection matrix*), che ci permette di trasformare direttamente le previsioni di base nelle previsioni riconciliate.

Infine, per facilitare la spiegazione della teoria alla base della riconciliazione ottima puntuale che sarà presentata nel prossimo paragrafo, definiamo il concetto di spazio riconciliato $\mathfrak{s}$ proposto da Panagiotelis et al. (2021 [31]): un set di previsioni riconciliate h passi in avanti $\tilde{\mathbf{y}}_h$, generato utilizzando l'informazione fino al tempo T, è detto coerente se $\tilde{\mathbf{y}}_h \in \mathfrak{s}$, dove $\mathfrak{s}$ è un sottoinsieme di $\mathbb{R}^n$ in cui valgono i vincoli di aggregazione, espressi secondo la formula 2.5.

Per chiarire questo concetto è utile fare riferimento allo spazio tridimensionale riportato in figura 2.4.

Le tre coordinate dello spazio sono corrispondenti ai valori di una semplice gerarchia con tre serie: y_A, y_B, y_{Tot}. Le previsioni di base possono giacere in qualsiasi punto di $\mathbb{R}^3$, mentre le previsioni che rispettano la proprietà di coerenza possono giacere solamente nel piano riconciliato $\mathfrak{s} \subset \mathbb{R}^3$, dove è rispettata la condizione $y_{Tot} = y_A + y_B$.

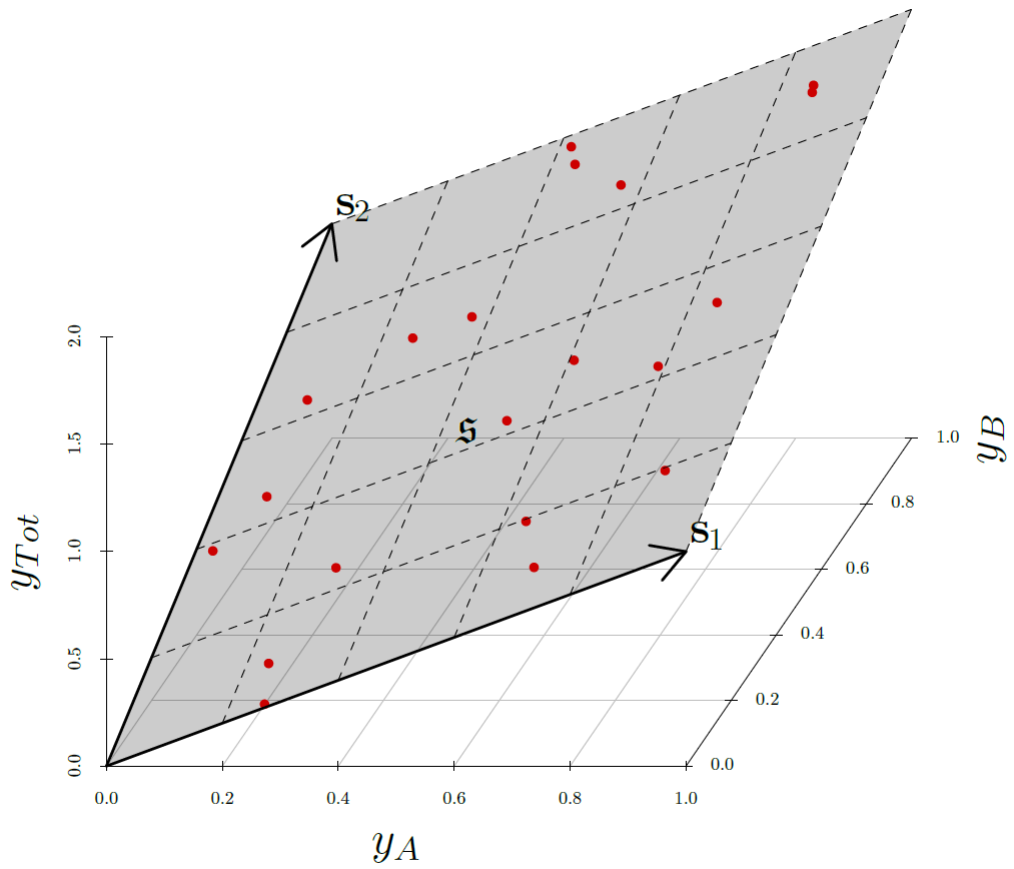


Figura 2.4: Rappresentazione di una gerarchia tridimensionale con $y_{Tot} = y_A + y_B$ tramite uno spazio tridimensionale. Il piano in grigio rappresenta lo spazio riconciliato all'interno del quale, con i punti in rosso, possiamo rappresentare dei possibili valori di previsioni riconciliate

2.2.3 Riconciliazione ottimale puntuale

Hyndman et al. (2011 [20]) hanno espresso il problema della riconciliazione della struttura gerarchica come un modello di regressione lineare generalizzato, valido per un fissato h, che include al suo interno i vincoli di aggregazione tramite la rappresentazione strutturale:

$$\tilde{\mathbf{y}}_h = \mathbf{S}\beta_h + \epsilon_h, \tag{2.15}$$

dove β_h è l'ignoto vettore di medie della *bottom time series* al tempo h da prevedere ed ϵ_h è il vettore casuale degli errori di riconciliazione con media

nulla e matrice di covarianza Σ_h. In questo caso, se la matrice Σ_h fosse nota e non singolare, l'espressione delle previsioni riconciliate utilizzando lo stimatore GLS di β_h sarebbe dato da

$$\tilde{\mathbf{y}}_h = \mathbf{S}(\mathbf{S}'\Sigma_h^{-1}\mathbf{S})^{-1}\mathbf{S}\Sigma_h^{-1}\hat{\mathbf{y}}_h.$$

Tuttavia, la matrice Σ_h, oltre a essere ignota, è spesso una matrice singolare per effetto dei vincoli di aggregazione sul vettore $\mathbf{y}_h$ e quindi, a meno di fare forti assunzioni riguardo alla matrice di covarianza, le stime riconciliate secondo questo procedimento non possono essere calcolate.

Per questi motivi, Wickramasuriya et al., (2019 [42]) propongono una soluzione alternativa che sfrutta i residui calcolati sulle previsioni di base. Indichiamo con $\mathbf{W}_h = Var(\hat{\mathbf{e}}_h|\mathbf{I}_T)$ la matrice di covarianza degli errori delle previsioni di base $\hat{\mathbf{e}}_h = \mathbf{y}_h - \hat{\mathbf{y}}_h$, data l'informazione fino al tempo T, allora Wickramasuriya et al. mostrano che

$$Var(\tilde{\mathbf{e}}_h|\mathbf{I}_T) = \mathbf{S}\mathbf{G}\mathbf{W}_h\mathbf{G}'\mathbf{S}',$$

dove $\tilde{\mathbf{e}} = \mathbf{y}_h - \tilde{\mathbf{y}}_h$ sono gli errori delle previsioni riconciliati e l'unica soluzione che minimizza la traccia ($minT$) della matrice $Var(\tilde{\mathbf{e}}_h|\mathbf{I}_T)$ è data da $\mathbf{G} = (\mathbf{S}'\mathbf{W}_h^{-1}\mathbf{S})^{-1}\mathbf{S}'\mathbf{W}_h{-}1$. Secondo questo procedimento (chiamato in letteratura 'approccio strutturale' o *structural approach*), le previsioni riconciliate saranno pari a

$$\tilde{\mathbf{y}}_h = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_h = \mathbf{S}(\mathbf{S}'\mathbf{W}_h^{-1}\mathbf{S})^{-1}\mathbf{S}'\mathbf{W}_h^{-1}\hat{\mathbf{y}}_h. \tag{2.16}$$

Servendoci del concetto di spazio riconciliato, è possibile anche riformulare il problema di riconciliazione ottimale puntuale nei termini di un problema di aggiustamento ai minimi quadrati vincolato. In particolare, come proposto da Di Fonzo e Girolimetto (2020 [9]) possiamo scrivere:

$$\hat{\mathbf{y}}_h = \mathbf{y}_h + \epsilon_h \quad s.t. \quad \mathbf{U}'\mathbf{y}_h = 0 \tag{2.17}$$

In questo approccio, chiamato 'di proiezione' o *projection approach* (van Erven e Cugliari, 2015 [11], Wickramasuriya et al., 2019 [42], Panagiotelis

et al., 2020 [31]), la soluzione del problema di stima è ottenuta "proiettando" nello spazio riconciliato $\mathfrak{s}$ le previsioni di base $\hat{\mathbf{y}}_h$ tramite una matrice di proiezione $\mathbf{M}$. Anche in questo caso per risolvere il problema quadratico linearmente vincolato che ci fornirà la soluzione, facciamo riferimento alla matrice $\mathbf{W}_h$, intesa come matrice metrica che induce la distanza di Mahalanobis. Otteniamo quindi che:

$$\tilde{\mathbf{y}}_h = \underset{\mathbf{y}_h}{\operatorname{argmin}}(\hat{\mathbf{y}}_h - \mathbf{y}_h)'\mathbf{W}_h^{-1}(\hat{\mathbf{y}}_h - \mathbf{y}_h) \quad s.t. \quad \mathbf{U}'\mathbf{y}_h = 0$$

che ha come soluzione:

$$\tilde{\mathbf{y}}_h = [\mathbf{I}_h - \mathbf{W}_h\mathbf{U}(\mathbf{U}'\mathbf{W}_h\mathbf{U})^{-1}\mathbf{U}']\hat{\mathbf{y}}_h = \mathbf{M}\hat{\mathbf{y}}_h, \tag{2.18}$$

dove $\mathbf{M} = [\mathbf{I}_h - \mathbf{W}_h\mathbf{U}(\mathbf{U}'\mathbf{W}_h\mathbf{U})^{-1}\mathbf{U}']$ è la matrice di proiezione. In Wickramasuriya et al. (2019 [42]) viene mostrata l'equivalenza tra le formulazioni 2.16 e 2.18. L'approccio di proiezione generalmente risulta più efficiente di quello strutturale dal momento che in 2.16 è necessario calcolare l'inversa della matrice $\mathbf{W}_h$, di dimensione $(n \times n)$, e della matrice $\mathbf{S}'\mathbf{W}_h^{-1}S$, di dimensione $(n_b \times n_b)$, mentre in 2.17 è sufficiente calcolare solamente l'inversa della matrice $\mathbf{U}'\mathbf{W}_h\mathbf{U}$, di dimensione $(n_a \times n_a)$.

Rimane da discutere la procedura per la stima di $\mathbf{W}_h$, nella pratica molto complessa, per questo motivo vengono riportate cinque diverse approssimazioni per il suo calcolo. In tutti i casi riportati viene sempre considerata una forma per la quale abbiamo $\mathbf{W}_h = k_h\mathbf{W}$ con k_h costante di proporzionalità che non è necessario stimare, in quanto può essere semplificata sia nella formulazione 2.16 che 2.18. Per indicare il nome delle approssimazioni della matrice $\mathbf{W}$, il prefisso $cs-$ a pedice, che sta per *cross-sectional*, indica che si sta considerando il caso di gerarchie contemporanee.

Abbiamo, quindi:

- $\hat{\mathbf{W}}_{cs-ols} = \mathbf{I}_n$

  Il caso più semplice risulta equivalente a utilizzare i minimi quadrati ordinari (OLS). Anche se questa soluzione è computazionalmente molto semplice, essa è ottimale solo se è presente equivarianza tra tutte

le serie della gerarchia, che invece hanno necessariamente delle scale dimensionali diverse.

- $\hat{\mathbf{W}}_{cs-struc} = diag(\mathbf{S}1_{n_b})$

  dove 1_{n_b} è un vettore unitario di dimensione n_b. Questa variante, chiamata *structural*, e proposta da Athanasopoulos et al. (2017 [3]) dipende solamente dalla struttura gerarchica e assume l'equivarianza solamente per le *bottom time series*, mentre per i livelli superiori la varianza di ogni nodo è pari al numero di *bottom time series* che somma. Per esempio, considerando la figura 2.1 avremo $\hat{\mathbf{W}}_{cs-struc} = diag(4, 3, 2, 1, 1, 1, 1, 1)$. Risulta utile soprattutto nel caso in cui non si abbiano a disposizione i residui.

- $\hat{\mathbf{W}}_{cs-sam} = \frac{1}{T}\sum_{t=1}^{T} \hat{e}_t \hat{e}_t'$

  dove $\hat{\mathbf{e}}_t = \mathbf{y}_t - \hat{\mathbf{y}}_t$ è il vettore dei residui in sample delle previsioni di base, di dimensione n. Anche se potrebbe sembrare la soluzione più intuitiva da utilizzare, e risulta molto semplice da ottenere, questa varianza è poco affidabile se $n > T$, o se T è di poco superiore ad n, per problemi di singolarità.

- $\hat{\mathbf{W}}_{cs-wls} = \mathbf{I}_n \bigodot \hat{\mathbf{W}}_{cs-sam}$

  dove $\bigodot$ indica il prodotto di Hadamard. Essenzialmente si ottiene una matrice che contiene sulla diagonale la varianza dei residui in-sample. Le previsioni di base vengono quindi scalate utilizzando i residui ottenendo una soluzione equivalente ai minimi quadrati pesati (WLS).

- $\hat{\mathbf{W}}_{cs-shr} = \lambda \hat{\mathbf{W}}_{cs-wls} + (1 - \lambda \hat{\mathbf{W}}_{cs-sam})$

  con $0 \leq \lambda \leq 1$. In questo caso gli elementi fuori dalla diagonale di $\hat{\mathbf{W}}_{cs-shr}$ sono "ristretti" verso lo 0 tramite il parametro di regolarizzazione (*shrinkage*) λ. Questa specificazione della matrice $\mathbf{W}$ ci permette di risolvere i problemi di singolarità che affliggono $\hat{\mathbf{W}}_{cs-sam}$ mantenendo l'informazione relativa alle correlazioni tra i residui. λ può essere calcolata tramite la formula (Ledoit e Wolf, 2004 [24], Schäfer

e Strimmer, 2005 [36]):

$$\lambda = \frac{\sum_{i \neq j} \hat{Var}(\hat{r}_{ij})}{\sum_{i \neq j} \hat{r}_{ij}^2} \tag{2.19}$$

dove $\hat{r}_{ij}^2$ è l'elemento di posto (i, j) della matrice di correlazione degli errori *in-sample*.

Capitolo 3

Gerarchia Temporale di una Serie Storica

Nell'analisi delle serie storiche temporali, generalmente si studiano variabili con frequenza diversa a seconda delle informazioni cheinteressa ottenere. In un ambito aziendale, per esempio, usualmente si fa ricorso a informazioni aggregate a bassa frequenza per previsioni strategiche di lungo termine, mentre decisioni operative a breve termine prediligono l'utilizzo di serie ad alta frequenza che facilitano l'individuazione di eventuali stagionalità o dinamiche di breve periodo.

In questo contesto, sempre con l'obiettivo di voler tener traccia contemporaneamente dell'andamento a breve, medio e lungo termine, è possibile adattare un approccio simile a quello usato per le strutture gerarchiche e raggruppate anche per la singola serie storica, che può essere aggregata temporalmente per produrre diverse serie storiche a frequenze più basse di quella di partenza. Per esempio, data una serie storica a cadenza mensile, tramite aggregazione temporale è possibile ottenere delle serie storiche bimestrali, trimestrali, quadrimestrali, semestrali e annuali, oppure da una serie storica a cadenza oraria possono essere ottenute le serie a cadenza giornaliera o settimanale.

Queste particolari strutture di dati vengono chiamate gerarchie temporali e, come mostrano Athanasopoulos et al. (2017 [3]), il loro utilizzo in un ambito di previsione, tramite delle opportune procedure di riconciliazione,

può fornire un'accuratezza maggiore rispetto alle previsioni convenzionali.

3.1 Notazione

Riprendendo la notazione di Athanasopoulos et al. (2017 [3]), indichiamo con m la più alta frequenza disponibile di una serie storica $\{x_t; t = 1, \dots, T\}$ e con $\mathcal{K} = \{k_p, k_{p-1}, \dots, k_1\}$ l'insieme dei p fattori di m, posti in ordine decrescente in modo che $k_p = m$ e $k_1 = 1$. Da ognuno di questi fattori sarà possibile, attraverso la somma non sovrapposta di k valori consecutivi, ottenere delle versioni temporalmente aggregate di x_t con dei periodi stagionali pari a $M_k = m/k$ con $k \in \mathcal{K}$. Indichiamo con N la lunghezza della serie osservata alla più bassa frequenza possibile e assumiamo infine, per disporre di dati regolari, che il numero delle osservazioni della serie a frequenza più alta sia un multiplo di m, quindi avremo che $T = N \cdot m$.

Definite queste quantità le serie aggregate possono essere riscritte nel seguente modo:

$$x_l^{[k]} = \sum_{t=(l-1)k+1}^{lk} x_t, \quad l = 1, \dots, M_k, \quad k \in \mathcal{K}.$$

Indichiamo con τ l'indice delle osservazioni al livello più aggregato, abbiamo quindi che $x_\tau^{[m]}, \tau = 1, \dots, N$, sarà la serie al livello di aggregazione temporale più elevato. Tramite l'indice τ sarà possibile rappresentare le diverse serie aggregate temporalmente con un vettore di dimensione M_k:

$$\mathbf{x}_\tau^{[k]} = \begin{bmatrix} x^{[k]}_{M_k(\tau-1)+1} & x^{[k]}_{M_k(\tau-1)+2} & \cdots & x^{[k]}_{M_k\tau} \end{bmatrix},$$

con $\tau = 1, \dots, N$ e $k \in \{k_{p-1}, \dots, k_2, 1\}$.

Possiamo anche raccogliere tutte le osservazioni disponibili per ogni τ nel vettore x_τ di dimensione $(k^* + m)$, con $k^* = \sum_{j=1}^{p-1} k_j$, in modo tale che

$$\mathbf{x}_\tau = \begin{bmatrix} x_\tau^{[m]} & \mathbf{x}_\tau^{[k_p-1]'} & \cdots & \mathbf{x}_\tau^{[k_2]'} & \mathbf{x}_\tau^{[1]'} \end{bmatrix}' = \begin{bmatrix} t'_{x_\tau} & \mathbf{x}_\tau^{[1]'} \end{bmatrix}'. \quad \tau = 1, \dots, N.$$

I valori di tutte le serie aggregate temporalmente sono quindi contenuti

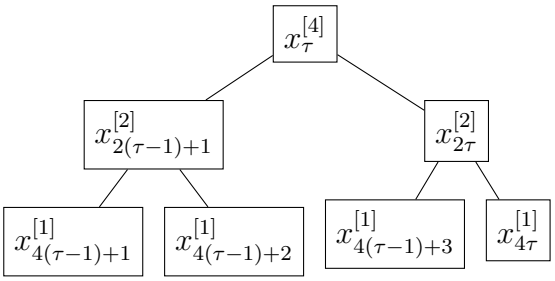


Figura 3.1: Diagramma ad albero per una serie storica trimestrale

nel vettore $t_{x_\tau} = \left[x_\tau^{[m]} \quad \mathbf{x}_\tau^{[k_p-1]'} \quad \cdots \quad \mathbf{x}_\tau^{[k_2]'}\right]'$ di dimensione k^*, mentre il vettore $\mathbf{x}_\tau^{[1]'}$, di dimensione m, contiene le osservazioni alla frequenza più alta per il tempo τ.

Data questa notazione possiamo utilizzare le rappresentazioni definite per le strutture gerarchiche anche per le gerarchie temporali. Per esempio, considerando una serie storica trimestrale che può essere aggregata in serie semestrali e annuali, riprendendo la formulazione 2.1, abbiamo che:

$$\begin{cases} x_\tau^{[4]} = x_{4(\tau-1)+1}^{[1]} + x_{4(\tau-1)+2}^{[1]} + x_{4(\tau-1)+3}^{[1]} + x_{4\tau}^{[1]} \\ x_{2(\tau-1)+1}^{[2]} = x_{4(\tau-1)+1}^{[1]} + x_{4(\tau-1)+2}^{[1]} \\ x_{2\tau}^{[2]} = x_{4(\tau-1)+3}^{[1]} + x_{4\tau}^{[1]} \end{cases} . \tag{3.1}$$

Tutto ciò essere rappresentato tramite il diagramma ad albero in figura 3.1

Analogamente al caso delle gerarchie contemporanee, anche per le gerarchie temporali è possibile esprimere i vincoli tramite opportune matrici usando una formulazione simile a quella delle formule 2.2, 2.4 e 2.5:

$$\mathbf{x}_\tau = \mathbf{R}_1 \mathbf{x}_\tau^{[1]}, \tag{3.2}$$

$$t_{x_\tau} = \mathbf{K}_1 \mathbf{x}_\tau^{[1]}, \tag{3.3}$$

$$\mathbf{Z}_1' \mathbf{x}_\tau = 0_{k^*}, \tag{3.4}$$

con $\tau = 1, \ldots, N$ e 0_r vettore di 0 di dimensione $(r \times 1)$. $\mathbf{R}_1$, di dimensione $[(k^* + m) \times m]$, è chiamata matrice di somma temporale (*temporal summing*

matrix), $\mathbf{K}_1$ è la matrice $(k^* \times m)$ di aggregazione temporale e $\mathbf{Z}_1$ è la matrice $[m \times (k^* + m)]$ dei vincoli di aggregazione (*zero costraints kernel matrix*). Le tre matrici sono la versione analoga al caso delle gerarchie contemporanee rispettivamente delle matrici $\mathbf{S}$, $\mathbf{C}$ e $\mathbf{U}$.

Per facilitare la creazione di queste matrici è anche possibile esprimerle nel seguente modo:

$$\mathbf{K}_1 = \begin{bmatrix} \mathbf{I}_{k_1} \otimes 1'_{k_p} \\ \mathbf{I}_{k_2} \otimes 1'_{k_{p-1}} \\ \vdots \\ \mathbf{I}_{k_{p-1}} \otimes 1'_{k_2} \end{bmatrix}, \quad \mathbf{R}_1 = \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{I}_m \end{bmatrix}, \quad \mathbf{Z}'_1 = \begin{bmatrix} \mathbf{I}_{k^*} & -\mathbf{K}_1 \end{bmatrix}. \tag{3.5}$$

con $\mathbf{I}_r$ matrice identità di dimensione $(r \times r)$, 1_r vettore di 1 di dimensione r e $\otimes$ prodotto di Kronecker.

Sempre facendo riferimento all'esempio definito dalla 3.1, le matrici appena presentate prenderanno la seguente forma:

$$\mathbf{K}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \mathbf{R}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{Z}_1 = \begin{bmatrix} 1 & 0 & 0 & -1 & -1 & -1 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & -1 \end{bmatrix}$$

Anche nel caso delle gerarchie temporali, non è sempre possibile rappresentare la struttura tramite un singolo diagramma ad albero. Per esempio, il caso già citato in cui da una serie storica a cadenza mensile vogliamo ottenere le serie bimestrali, trimestrali, quadrimestrali, semestrali e annuali deve essere rappresentato dai due diversi diagrammi ad albero in figura 3.2; sarà quindi necessario ricorrere a una struttura raggruppata.

Tenendo presente che in questo caso abbiamo $\mathcal{K} = \{12, 6, 4, 3, 2, 1\}$ tramite le formule 3.5 abbiamo che:

$$\mathbf{K}_1 = \begin{bmatrix} \mathbf{I}_1 \otimes 1'_{12} \\ \mathbf{I}_2 \otimes 1'_6 \\ \mathbf{I}_3 \otimes 1'_4 \\ \mathbf{I}_4 \otimes 1'_3 \\ \mathbf{I}_6 \otimes 1'_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Dalla quale è possibile ottenere le matrici $\mathbf{R}_1$ e $\mathbf{Z}_1$ secondo l'espressione 3.5. Per estendere la formulazione appena presentata all'intero periodo di osservazione di x_t, indichiamo con $\mathbf{x} = \begin{pmatrix} \mathbf{x}'_1 & \cdots & \mathbf{x}'_\tau & \cdots & x'_N \end{pmatrix}$ il vettore di dimensione $N \cdot (k^* + m)$ contenente tutte le osservazioni a qualsiasi frequenza. In questo caso possiamo definire i vincoli di aggregazione tramite la formula:

$$\mathbf{Z}'_N \mathbf{x} = 0_{Nk^*} \tag{3.6}$$

Dove $\mathbf{Z}'_N = \begin{bmatrix} \mathbf{I}_{Nk^*} & -\mathbf{K}_N \end{bmatrix}$ e $\mathbf{K}_1 = \begin{bmatrix} \mathbf{I}_{k_1 N} \otimes 1'_{k_p} \\ \mathbf{I}_{k_2 N} \otimes 1'_{k_{p-1}} \\ \vdots \\ \mathbf{I}_{k_{p-1} N} \otimes 1'_{k_2} \end{bmatrix}$

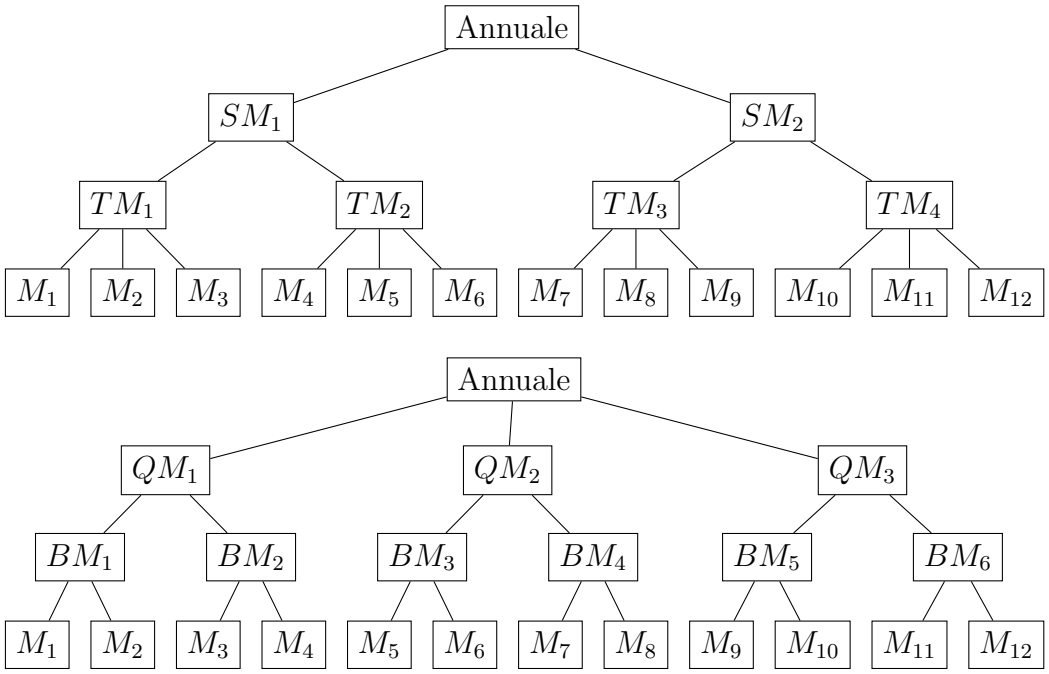


Figura 3.2: diagrammi ad albero necessari per rappresentare i possibili ordini di aggregazione di una serie storica a cadenza mensile

3.2 Riconciliazione temporale di previsioni puntuali

L'utilizzo di set di informazioni diverse, unito al fatto che serie di frequenze temporali diverse prediligono modelli diversi per la procedura di stima, ci portano ad ottenere anche in questo caso delle previsioni che nella maggior parte dei casi non rispetteranno i vincoli di aggregazione temporale alle diverse frequenze. Affidarsi a previsioni contrastanti potrebbe portare per esempio ad un processo decisionale che prevede un mercato stabile per il breve termine e un mercato in crescita per il lungo, in ambito aziendale questo può tramutarsi in sprechi o opportunità perse, ed eventuali costi aggiuntivi (Jeon et al., 2012 [23]).

Anche in questo contesto può essere quindi molto importante disporre di previsioni coerenti, in modo da descrivere andamenti a breve e a lungo termine non in contraddizione.

3.2.1 Notazione

In riferimento alla serie storica con la frequenza più bassa, indichiamo con H il massimo orizzonte di previsione e per ogni livello di aggregazione k generiamo $M_k H$ previsioni. Le previsioni di base possono essere raccolte, per ogni orizzonte $h = 1, \ldots, H$ in un vettore di dimensione $(k^* + m)$ definito nel seguente modo:

$$\hat{\mathbf{x}}_h = \begin{bmatrix} \hat{x}_h^{[m]} & \hat{\mathbf{x}}_h^{[k_p-1]'} & \cdots & \hat{\mathbf{x}}_h^{[k_2]'} & \hat{\mathbf{x}}_h^{[1]'} \end{bmatrix}' \tag{3.7}$$

Dove $\mathbf{x}_h^{[k]} = \begin{bmatrix} \hat{x}_{M_k(h-1)+1}^{[k]} & \hat{x}_{M_k(h-1)+2}^{[k]} & \cdots & \hat{x}_{M_k h}^{[k]} \end{bmatrix}'$ è il vettore di dimensione M_k contenente le previsioni al tempo h per l'ordine di aggregazione k. Il vettore $\hat{\mathbf{x}}_h$ non rispetterà i vincoli di aggregazione; quindi, come nel caso contemporaneo il nostro obiettivo sarà individuare il vettore dei valori riconciliati $\tilde{\mathbf{x}}_h$ tale che valga $\mathbf{Z}_1' \tilde{\mathbf{x}}_h = 0_{k^*}$ per ogni h.

3.2.2 Riconciliazione ottimale puntuale

Oltre ai metodi di riconciliazione tradizionali riportati nel capitolo 2.2.1 è possibile, come spiegano Athanasopoulos et al., (2017 [3]) estendere la procedura di riconciliazione ottimale puntuale di una gerarchia contemporanea al caso temporale.

In particolare, possiamo adattare la formula 2.16 per calcolare i valori riconciliati tramite l'approccio strutturale in modo tale che:

$$\tilde{\mathbf{x}}_h = \mathbf{R}_1 (\mathbf{R}_1' \Omega_h^{-1} \mathbf{R}_1)^{-1} \mathbf{R}_1' \Omega_h^{-1} \hat{\mathbf{x}}_h = \mathbf{R}_1 \mathbf{G} \hat{\mathbf{x}}_h. \tag{3.8}$$

Mentre per quanto riguarda l'approccio di proiezione, nel caso temporale avremo che:

$$\tilde{\mathbf{x}}_h = [\mathbf{I}_h - \Omega_h \mathbf{Z}_1 (\mathbf{Z}_1' \Omega_h \mathbf{Z})^{-1} Z_1'] \hat{\mathbf{x}}_h = \mathbf{M} \hat{\mathbf{x}}_h, \tag{3.9}$$

dove Ω_h è la matrice $[(k^* + m) \times (k^* + m)]$ di covarianza degli errori delle previsioni di base, analoga, nel caso contemporaneo, alla matrice $\mathbf{W}_h$. Per semplificare i calcoli viene considerata nuovamente un'approssimazione di Ω_h tale che $\Omega_h = k_h \Omega$.

Per definire i residui da utilizzare in alcune approssimazioni per Ω_h , indichiamo con $\hat{\mathbf{e}}_{\tau}^{[k]}$ il vettore di dimensione M_k dei residui *in sample* al tempo τ:

$$\hat{\mathbf{e}}_{\tau}^{[k]} = \mathbf{x}_{\tau}^{[k]} - \hat{\mathbf{x}}_{\tau}^{[k]}, \quad \tau = 1, \dots, N, \quad k \in \mathcal{K}$$

Per ogni ordine di aggregazione possiamo poi definire la matrice $(N \times M_k)$

$$\hat{\mathbf{E}}_{x}^{[k]} = \begin{bmatrix} \hat{\mathbf{e}}_{1}^{[k]} \\ \vdots \\ \hat{\mathbf{e}}_{\tau}^{[k]} \vdots \\ \hat{\mathbf{e}}_{N}^{[k]} \end{bmatrix}, \quad k \in \mathcal{K}.$$

Le p diverse matrici $\hat{\mathbf{E}}_{x}^{[k]}$ possono essere a loro volta aggregate nella matrice dei residui *in-sample* di dimensione $[N \times (k^* + m)]$:

$$\hat{\mathbf{E}}_{x} = \begin{bmatrix} \hat{\mathbf{E}}_{x}^{[m]} & \hat{\mathbf{E}}_{x}^{[k_p-1]} & \cdots & \hat{\mathbf{E}}_{x}^{[k_2]} & \hat{\mathbf{E}}_{x}^{[1]} \end{bmatrix}.$$

Di seguito riportiamo diverse espressioni per la definizione dell'approssimazione della matrice Ω proposte da Athanasopoulos et al. (2010 [3]), Hyndman e Kourentzes (2018 [22]), Nystrup et al. (2020 [27]). Il prefisso $t-$ indica che stiamo operando in un contesto di gerarchie temporali, abbiamo che:

- $\hat{\Omega}_{t-ols} = \mathbf{I}_{k^*+m}$

  Anche in questo caso è possibile ricorrere alla matrice identità per l'approssimazione di Ω ottenendo lo stimatore dei minimi quadrati ordinari (OLS). Tuttavia, come nel caso contemporaneo, l'assunzione di equivarianza nella gerarchia temporale è inverosimile e per questo un'approssimazione di questo tipo è sconsigliata.

- $\hat{\Omega}_{t-struc} = diag(\mathbf{R}_1 1_m)$

  con 1_m vettore unitario di dimensione m. Nel caso in cui non si disponga della matrice dei residui *in-sample*, una scelta più verosimile per Ω consiste, come nel caso contemporaneo, nella varinte *structural*

nella quale si assume l'equivarianza per gli errori solo all'interno di ciascun ordine di aggregazione.

- $\hat{\Omega}_{t-sam} = \frac{1}{N}(\hat{\mathbf{E}}_x)'\hat{\mathbf{E}}_x$

  Utilizzando la matrice di covarianza campionaria dei residui, è facile imbattersi in problemi di non invertibilità nel caso in cui $N \leq (k^*+m)$, questa scelta nell'approssimazione di Ω quindi non è sempre fattibile.

- $\hat{\Omega}_{t-shr} = \lambda(I_{k^*+m} \odot \hat{\Omega}_{t-sam}) + (1-\lambda)\hat{\Omega}_{t-sam}$

  Quando non è possibile utilizzare la matrice di covarianza campionaria dei residui, come nel caso contemporaneo è possibile risolvere il problema della non invertibilità della matrice facendo ricorso alla versione *shrinkage* di $\hat{\Omega}_{t-sam}$, calcolando lambda come in formula 2.19.

- $\hat{\Omega}_{t-wlsh} = \mathbf{I}_{k^*+m} \odot \hat{\Omega}_{t-sam}$

  Tramite questo stimatore si ignora la correlazione tra ed entro i livelli e vengono considerate solamente le $(k^* + m)$ diverse varianze stimate per ogni nodo.

- $\hat{\Omega}_{t-wlsv} = \begin{bmatrix} (\sigma^{[m]})^2 & 0 & \cdots & 0 \\ 0 & (\sigma^{[k_{p-1}]})^2\mathbf{I}_{M_{k_{p-1}}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (\sigma^{[1]})^2\mathbf{I}_{M_1} \end{bmatrix}$

  Continuando ad ignorare gli elementi all'infuori della diagonale della matrice di correlazione, in questo caso vengono calcolate solamente le p diverse varianze degli errori $(\sigma^{[k]})^2$ stimate per ogni ordine di aggregazione $k \in \mathcal{K}$.

Nelle seguenti proposte per l'approssimazione della matrice Ω (Nystrup et al., 2020 [27]), si ignora ancora la correlazione tra i diversi ordini di aggregazione ma si considera l'autocorrelazione entro ognuno di questi.

- $\hat{\Omega}_{t-acov} = \begin{bmatrix} \hat{\Omega}^{[m]} & 0 & \cdots & 0 & 0 \\ 0 & \hat{\Omega}^{[k_{p-1}]} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \hat{\Omega}^{[k_2]} & 0 \\ 0 & 0 & \cdots & 0 & \hat{\Omega}^{[1]} \end{bmatrix}$

  le matrici $\hat{\Omega}^{[k]}$, di dimensione $(M_k \times M_k)$ sono le matrici di covarianza campionaria dei residui per l'ordine di aggregazione k e vengono calcolate come

$$\hat{\Omega}^{[k]} = \frac{1}{N}\sum_{\tau=1}^{N} \hat{\mathbf{e}}_{\tau}^{[k]}(\hat{\mathbf{e}}_{\tau}^{[k]})' = \frac{1}{N}(\hat{\mathbf{E}}_{x}^{[k]})'\hat{\mathbf{E}}_{x}^{[k]}, \quad k \in \mathcal{K}$$

  Per ricorrere a questa matrice è necessario avere $N > m$, altrimenti non potrà essere invertita.

- $\hat{\Omega}_{t-strar} = \hat{\Omega}_{t-struc}^{\frac{1}{2}} \Gamma \hat{\Omega}_{t-struc}^{\frac{1}{2}}$

  dove Γ è una matrice $[(k^* + m) \times (k^* + m)]$ composta da p diversi blocchi definita nel seguente modo

$$\Gamma = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \Gamma^{k_p - 1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Gamma^{[1]} \end{bmatrix} \tag{3.10}$$

  E per le diverse matrici $\Gamma^{[k]}$ si considera la matrice di autocorrelazione di un processo AR(1) del corrispondente ordine di aggregazione, creata a partire dalla stima dei coefficienti di autocorrelazione del primo ordine dei residui $\rho_{[k]}$ per ciascun livello $k \in \mathcal{K}$:

$$\Gamma^{[k]} = 1, \quad \Gamma^{[k]} = \begin{bmatrix} 1 & \rho_{[k]} & \cdots & \rho_{[k]}^{M_k - 1} \\ \rho_{[k]} & 1 & \cdots & \rho_{[k]}^{M_k - 2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{[k]}^{M_k - 1} & \rho_{[k]}^{M_k - 2} & \cdots & 1 \end{bmatrix}, \quad k \in \mathcal{K}.$$

Dal momento che queste matrici di covarianza vengono scalate per i pesi ottenuti dalla struttura gerarchica, questa approssimazione viene chiamata *structural* AR(1).

- $\hat{\Omega}_{t-sar1} = \hat{\Omega}^{\frac{1}{2}}_{t-wlsv} \Gamma \hat{\Omega}^{\frac{1}{2}}_{t-wlsv}$

  Con Γ definita come 3.10. Come nel caso *structural* AR(1) si assume una struttura auto-regressiva di ordine 1 per le serie all'interno di ogni ordine di aggregazione, ma viene scalata tramite le stime delle varianze di ogni nodo.

- $\hat{\Omega}_{t-har1} = \hat{\Omega}^{\frac{1}{2}}_{t-wlsh} \Gamma \hat{\Omega}^{\frac{1}{2}}_{t-wlsh}$

  Analogo ai due casi precedenti, ma lo *scaling* è ottenuto tramite le varianze stimate di ogni ordine di aggregazione.

Capitolo 4

Strumenti per la riconciliazione disponibili

Per risolvere il problema della riconciliazione di previsioni di serie storiche gerarchiche sono stati sviluppati diversi pacchetti, soprattutto per il software R, disponibili gratuitamente per il download sulla piattaforma CRAN. Basati su questi ultimi sono stati poi sviluppati alcuni strumenti anche all'interno del linguaggio Python, che saranno descritti nel capitolo successivo.

4.1 Pacchetti per la riconciliazione in R

Il pacchetto più diffuso per l'applicazione delle tecniche di riconciliazione contemporanea di previsioni di serie storiche gerarchiche e raggruppate attualmente è ancora `hts` (Hyndman et al., 2021 [21]) che, pubblicato il 22 marzo 2010, è stato anche il primo pacchetto di funzioni specializzato nel trattamento di questa tematica. Nel corso degli anni a questo pacchetto sono state aggiunte diverse funzionalità, tra le quali la riconciliazione non negativa, ma dal 2020 non è solo parzialmente supportato in quanto gli autori hanno spostato la loro attenzione sullo sviluppo del pacchetto `fable` (O'Hara-Wild et al., 2021 [29]). Quest'ultimo punta ad inglobare tutte le procedure presenti in `hts`, sviluppandole in un flusso di lavoro coerente con il *tidyverse* (Wickham et al., 2022 [41]).

Sempre nell'ambito della riconciliazione contemporanea è disponibile anche il pacchetto `gtop` (Cugliari e Van Erven, 2015 [8]) che si concentra sull'approccio di proiezione. Questo pacchetto però presenta molti problemi nella gestione di gerarchie con molte serie storiche e offre una ristretta varietà di proposte per l'approssimazione della matrice di autocovarianza degli errori delle previsioni riconciliate $\mathbf{W}$.

Per quanto riguarda la riconciliazione temporale invece è disponibile il pacchetto `thief` (Hyndman e Kourentzes, 2018 [22]) che implementa i metodi descritti in Athanasopoulos et al. (2017 [3]).

4.2 Il pacchetto FoReco

Il pacchetto `FoReco` (Girolimetto e Di Fonzo, 2022 [13]) è stato pubblicato per la prima volta con il nome di `forec` il 1° ottobre 2020 da Daniele Girolimetto e Tommaso Di Fonzo. Da quella data è continuamente aggiornato con nuove funzionalità e ottimizzazioni e allo stato attuale rappresenta la collezione di funzioni più completa disponibile nell'ambito della riconciliazione di previsioni di serie storiche gerarchiche.

La caratteristica principale di FoReco è che a differenza di altri pacchetti dedicati alla riconciliazione presenti in R come `hts` e `thief`, fortemente legati alle tecniche con le quali sono ottenute le previsioni di base, in `FoReco` le procedure per la riconciliazione sono applicate in maniera completamente indipendente da queste.

Il pacchetto rende disponibili tecniche classiche per la riconciliazione (*bottom-up* e *top-down*) e le tecniche più moderne basate sulla riconciliazione ottimale puntuale viste nei capitoli precedenti.

Oltre ad essere uno strumento unificato per la riconciliazione contemporanea e temporale, in `FoReco` è stata implementata per la prima volta anche la riconciliazione cross-temporale, che agisce contemporaneamente sulla gerarchia temporale e contemporanea di un insieme di serie storiche, argomento non trattato in questa tesi.

All'interno di `FoReco` la rappresentazione utilizzata di default per risolvere i problemi di riconciliazione è quella di proiezione, ma volendo è

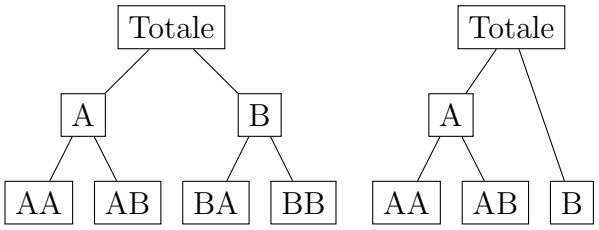


Figura 4.1: Esempio di gerarchia bilanciata (a sinistra) e non bilanciata (a destra)

possibile fare ricorso anche alla rappresentazione strutturale.

In questo pacchetto, per poter gestire efficacemente gerarchie di grandi dimensioni, si fa anche ricorso all'utilizzo di matrici sparse tramite il pacchetto di R `Matrix` (Bates e Maechler, 2022 [4]).

Due aspetti della riconciliazione sui quali `FoReco` presta molta attenzione, che lo distinguono rispetto agli altri pacchetti che trattano questa tematica, sono la gestione delle gerarchie sbilanciate e la riconciliazione non negativa.

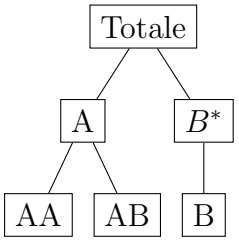
4.2.1 Gerarchie bilanciate e sbilanciate

Nel contesto della riconciliazione contemporanea è possibile distinguere tra due diversi tipi di gerarchie, quelle bilanciate e quelle non bilanciate. Una gerarchia è detta bilanciata se ogni nodo dei livelli intermedi presenta almeno due legami con i nodi del livello inferiore, in caso contrario la gerarchia è detta non bilanciata.

In molte circostanze (come nel pacchetto di R `hts`) nel momento della definizione della matrice $\mathbf{C}$ si decide di bilanciare artificiosamente una gerarchia non bilanciata aggiungendo dei nodi in corrispondenza dei livelli che non presentano nodi legati alle bottom time series. Nel caso dell'esempio in figura 4.1 (a destra), la matrice $\mathbf{S}$ e la corrispondente gerarchia bilanciata

artificiosamente prendono le seguenti forme:

$$\mathbf{S} = \begin{bmatrix} \mathbf{C} \\ \mathbf{I}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$



Come si può notare, la terza e la sesta riga della matrice $\mathbf{S}$ per questa gerarchia sono identiche. Questo espediente comporta diversi problemi e inconsistenze nel calcolo delle previsioni riconciliate. Innanzitutto, a seconda delle assunzioni che si scelgono per la definizione della matrice di covarianza degli errori $\mathbf{W}$ per la serie B saranno calcolate due varianze diverse anche se si sta facendo riferimento ad uno stesso elemento, che potrebbero influire anche sul calcolo delle varianze di altre serie. Per lo stesso motivo, la matrice $\mathbf{W}$ potrebbe risultare più complicata da invertire nel caso si faccia ricorso ad approcci che potrebbero avere questo tipo di problemi. Infine, soprattutto nel caso di grandi gerarchie non bilanciate, il calcolo delle serie riconciliate comporterà più calcoli di quanti ne siano effettivamente necessari, appesantendo l'esecuzione del codice sia con maggiore occupazione di memoria che con un aumento dei tempi di calcolo.

Per questo, nel caso di gerarchie non bilanciate in FoReco viene corretta la matrice $\mathbf{C}$ in modo da eliminare le righe ridondanti che verrebbero a crearsi nella matrice S come riportato di seguito:

$$\tilde{\mathbf{C}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \quad \tilde{\mathbf{S}} = \begin{bmatrix} \tilde{\mathbf{C}} \\ \mathbf{I}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

4.2.2 Riconciliazione non negativa

Le procedure di riconciliazione, soprattutto nel caso in cui si ha a che fare con *bottom time series* che presentano molti zeri, possono portare a previsioni riconciliate negative anche per fenomeni che non dovrebbero presentare questi valori, come per esempio nel caso di serie che indicano conteggi.

Potrebbe quindi essere necessario avere a disposizione delle previsioni che rispettino il vincolo della non negatività, oltre ai vincoli dovuti alla struttura gerarchica. Bisogna tenere presente che l'imposizione dei vincoli di non negatività delle previsioni non punta principalmente a migliorare l'accuratezza, ma si pone l'obiettivo di fornire valori utilizzabili per prendere decisioni allineate ad ogni livello e dotate di un significato operativo (Wickramasuriya et al., 2020 [43]).

Una soluzione semplice per risolvere il problema della non negatività è porre uguali a zero tutti i valori riconciliati negativi e successivamente sommare il nuovo vettore di *bottom time series* tramite la matrice di aggregazione contemporanea; tuttavia, questa procedura, pur risultando molto semplice da implementare, non ha nessun supporto teorico che la giustifichi.

Per risolvere il problema della riconciliazione non negativa Wickramasuya et al. (2020 [43]) hanno proposto diversi algoritmi, tra i quali quello di block principal pivoting implementato poi nel pacchetto `hts`, che si basa sull'idea di trasformare il problema dei minimi quadrati vincolati da disuguaglianze in una sequenza di problemi con vincoli di uguaglianza.

`FoReco` fa invece ricorso al pacchetto `OSQP` (Stellato et al., 2021 [38]) che implementa un risolutore di problemi di programmazione quadratica basato sul metodo della direzione alternata dei moltiplicatori di Lagrange (Boyd et al., 2011 [6]). L'algoritmo si presta molto bene alla risoluzione del problema di riconciliazione non negativa in quanto, oltre ad essere molto robusto, non impone requisiti come l'indipendenza lineare dei vincoli. In `OSQP`, per ottenere una soluzione accurata, è implementato anche uno step di solution polishing, che però richiede la risoluzione di un sistema lineare aggiuntivo e in alcuni casi può dare luogo ad un aumento dei tempi di calcolo.

Considerando l'approccio di proiezione, il problema definito a 2.18, fis-

sato $h = 1$ e introducendo i vincoli di non negatività, può essere riscritto come:

$$\min_{\mathbf{y}} \frac{1}{2}\mathbf{y}'\mathbf{W}^{-1}y - \hat{\mathbf{y}}\mathbf{W}^{-1}\mathbf{y} \quad s.t. \quad \mathbf{U}'\mathbf{y} = 0, \quad \mathbf{y} > 0, \tag{4.1}$$

dove $\hat{\mathbf{y}}$ è il vettore $n \times 1$ delle previsioni di base, $\mathbf{W}$ è la matrice di covarianza definita positiva $(n \times n)$ e $\mathbf{U}'$ è la matrice $(n_a \times n)$ dei vincoli di aggregazione contemporanea.

La forma dei problemi di minimo che `OSQP` richiede in ingresso è la seguente:

$$\min_{x} \frac{1}{2}x'\mathbf{P}x + q'x \quad s.t. \quad l \leq \mathbf{A}x \leq u. \tag{4.2}$$

Per ottenere il problema nella forma 4.1 è quindi necessario porre:

$$\mathbf{P} = \mathbf{W}^{-1}, \quad q' = -\hat{y}\mathbf{W}^{-1}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{U} \\ \mathbf{I}_n \end{bmatrix}_{(n_a+n)\times n},$$

$$l = 0_{(n_a+n)\times 1}, \quad u = \begin{bmatrix} 0_{n_a \times 1} \\ +\infty_{n\times 1} \end{bmatrix}_{(n_a+n_b)\times 1}$$

Considerando gerarchie usuali nelle quali le upper time series sono ottenute come combinazione lineare con coefficienti positivi delle *bottom time series*, è possibile prendere alcuni accorgimenti modificando le matrici definite precedentemente per eliminare alcuni vincoli ridondanti e migliorare quindi l'efficienza dell'algoritmo di riconciliazione. Per esempio, nel caso usuale di aggregazione data da somme, considerando la gerarchia più semplice con $(n_b = 2)$ e $(n_a = 1)$, abbiamo:

$$\mathbf{A} = \begin{bmatrix} \mathbf{U} \\ \mathbf{I}_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y \\ y_1 \\ y_2 \end{bmatrix}.$$

Sostituendo nei vincoli in 4.2 si ottiene:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} y - y_1 - y_2 \\ y \\ y_1 \\ y_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ +\infty \\ +\infty \\ +\infty \end{bmatrix}$$

La condizione su y riportata nella seconda riga della disequazione precedente risulterà quindi inutile, in quanto la positività di y è già assicurata dalle condizioni richieste alle due *bottom time series*.

Generalizzando, i vincoli che dobbiamo considerare per la risoluzione del problema possono quindi essere ristretti a:

$$\begin{cases} \mathbf{U}'\mathbf{y} = 0 \\ \mathbf{y}_b > 0 \end{cases}.$$

Dove $\mathbf{y}_b$ è il vettore delle previsioni delle *bottom time series*.

Possiamo sistemare la definizione della matrice $\mathbf{A}$ e dei vettori u e l, per considerare solamente i vincoli strettamente necessari nel seguente modo:

$$\mathbf{A} = \begin{bmatrix} \mathbf{U} \\ \begin{bmatrix} 0_{n_a} & \mathbf{I}_{n_b} \end{bmatrix} \end{bmatrix}_{n \times n}, \quad l = \begin{bmatrix} 0_{n_a \times 1} \\ 0_{n_b \times 1} \end{bmatrix} = 0_{n \times 1}, \quad u = \begin{bmatrix} 0_{n_a \times 1} \\ +\infty_{n_b \times 1} \end{bmatrix}_{n \times 1}.$$

Considerando gli accorgimenti definiti nel caso contemporaneo, è semplice l'estensione del problema al caso temporale:

$$\min_{\mathbf{y}} \frac{1}{2}\mathbf{y}'\Omega^{-1}\mathbf{y} - \hat{\mathbf{y}}\Omega^{-1}\mathbf{y} \quad s.t. \quad \mathbf{Z}'\mathbf{y} = 0, \quad \mathbf{y} > 0, \tag{4.3}$$

Dove Ω è la matrice di covarianza definita positiva ($(k^* + m) \times (k^* + m)$), $\hat{\mathbf{y}}$ è il vettore $(k^* + m) \times 1)$ delle previsioni di base e $\mathbf{Z}'$ è la matrice ($k^* \times (k^* + m)$) dei vincoli di aggregazione temporale. In questo caso per ottenere la formulazione eliminando le condizioni ridondanti poniamo:

$$\mathbf{P} = \Omega^{-1}, \quad q' = -\hat{\mathbf{y}}\Omega^{-1}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{U} \\ \begin{bmatrix} 0_{k^*} & \mathbf{I}_m \end{bmatrix} \end{bmatrix}_{(k^*+m) \times (k^*+m)},$$

$$l = 0_{(k^*+m)\times 1}, \quad u = \begin{bmatrix} 0_{k^*\times 1} \\ +\infty_{m\times 1} \end{bmatrix}_{(k^*+m)\times 1}.$$

In `FoReco` inoltre è possibile anche impostare manualmente i valori da utilizzare nei vettori l e u per soddisfare dei vincoli diversi dalla semplice non negatività.

Capitolo 5

Il pacchetto FoReco in Python

Il lavoro principale di questa tesi è consistito nel trasferire tutto quello che fa `FoReco` in R per le gerarchie contemporanee e temporali nel linguaggio Python, per valutare se in questo nuovo ambiente le performance del pacchetto possono migliorare in termini di efficienza, e per mettere a disposizione della comunità di Python delle tecniche, quelle della riconciliazione, ancora poco diffuse in questo linguaggio. Dove non specificato, nel seguito con `FoReco` si farà sempre riferimento alla versione di Python del pacchetto.

5.1 Il linguaggio Python

L'origine di Python risale al 1989, quando Guido Von Rossum iniziò a ideare quello che doveva essere un successore di ABC, un linguaggio di programmazione destinato quasi unicamente all'insegnamento e alla prototipazione. Lo sviluppo pluridecennale di Python, grazie soprattutto alla sua numerosissima - e affiatatissima – community, nel tempo ha portato a un'estensione delle sue potenzialità in aree ben più ampie di quelle per cui ABC era stato creato (Vanners 2003 [39]). Al giorno d'oggi, infatti, questo linguaggio è utilizzato in ormai quasi tutti gli ambiti dell'informatica: sviluppo di siti o applicazioni Web e desktop, realizzazione di interfacce grafiche, amministrazione di sistema, calcolo scientifico e numerico, gestione di database, machine learning, data science e molto altro.

Una delle motivazioni che hanno garantito a Python il successo di cui gode ora è quella della semplicità e della chiarezza. Infatti, la stessa grammatica del linguaggio cerca di enfatizzare la leggibilità del codice tramite un estensivo uso dell'indentazione. Inoltre, spesso si fa anche ricorso direttamente all'uso di specifiche parole in inglese piuttosto che ai simboli per specificare le istruzioni da dare al linguaggio.

5.1.1 Caratteristiche tecniche

Python è un linguaggio di programmazione di alto livello caratterizzato dall'adozione di diversi paradigmi di programmazione. Nativamente supporta la programmazione orientata agli oggetti, la programmazione strutturale e la programmazione funzionale, ma tramite estensioni possono essere supportati anche diversi altri paradigmi. È un linguaggio di programmazione dinamico e quindi non necessita di dichiarare le variabili e il tipo di queste durante la scrittura del codice.

Per quanto riguarda la gestione della memoria, fa ricorso ad un *garbage collector*, per liberare automaticamente la memoria allocata che non viene più utilizzata, e al *reference counting*, per evitare di occupare più spazio del necessario quando vengono creati diversi oggetti uguali (Python Software foundation 2022 [34]).

Un limite importante per questo linguaggio di programmazione allo stato attuale è costituito dal *Global Interpreter Lock* (GIL), un *mutex*, che limita l'esecuzione del codice ad un singolo thread. Questa soluzione, pur permettendo una gestione della memoria sicura necessaria al reference counting, limita fortemente le potenzialità delle macchine con processori *multi-core* dove viene eseguito del codice intensivo per la CPU (Hastings, 2015 [16]). Tuttavia, in alcuni contesti e tramite particolari espedienti il GIL può essere aggirato o "rilasciato" per permettere ai programmi di Python di sfruttare tutti i core della macchina.

Un altro importante pilastro della filosofia di Python è il ristretto numero di funzioni native offerte dal linguaggio (Peters 2004 [33]). Una delle idee alla base di Python, infatti, è che piuttosto di appesantire il core del programma, vengano importate solamente le funzioni strettamente necessa-

rie tramite dei file esterni detti moduli. Nel caso di problemi più complessi è possibile importare nel codice anche dei pacchetti o delle librerie, termini con i quali in Python si intendono delle collezioni di diversi moduli.

Per questo motivo il linguaggio di Python di base non si presta molto bene all'analisi statistica, ma con il passare degli anni sono stati implementati un vastissimo numero di pacchetti e librerie specializzati in questo ambito. Nonostante questo, per l'analisi statistica in generale Python non ha ancora le stesse disponibilità di R, ma in alcuni campi, come per esempio quello del deep learning, Python allo stato attuale è il linguaggio di programmazione più utilizzato.

5.2 Serie storiche gerarchiche in Python

Per quanto riguarda le serie storiche gerarchiche nell'ambito di Python, allo stato attuale c'è ancora una disponibilità molto limitata di pacchetti che trattano questo argomento.

Il lavoro più citato per quanto riguarda la previsione e la riconciliazione di serie storiche contemporanee, pur essendo attualmente ancora nelle fasi iniziali del suo sviluppo, è `scikit-hts` (Mazzaferro et al., 2021 [26]). Basandosi su Hyndman e Athanasopoulos (2018 [17]) questo pacchetto fornisce un'implementazione in Python delle tecniche di previsione per serie storiche gerarchiche e raggruppate, dalla fase di stima fino alla riconciliazione.

Un altro pacchetto disponibile in Python per la riconciliazione di serie storiche è `pyhts` (Zang, 2021 [44]), che si propone come una trasposizione in Python del pacchetto di R `hts`. Non si tratta però di una trasposizione completa, in quanto alcune funzionalità di `hts`, come la riconciliazione non negativa, non sono implementate.

È da citare anche il pacchetto `htsprophet` (Rooney, 2021 [35]) che, basandosi sempre su Hyndman e Athanasopoulos (2018 [17]), e servendosi della procedura di previsione prophet (Facebook Inc., 2022 [12]), permette di ottenere delle previsioni di serie storiche gerarchiche riconciliate ma non di riconciliare previsioni di serie storiche da fonti esterne al pacchetto.

Per quanto riguarda invece la riconciliazione di gerarchie temporali, per quello che ci è noto, l'unico pacchetto presente in Python che implementa queste procedure è `TimeHierarchy` (Donadio, 2021 [10]), che tuttavia risulta privo di documentazione.

5.3 Pacchetti esterni utilizzati in FoReco

Come è già stato anticipato, dal momento che Python offre nativamente una disponibilità estremamente ridotta di funzioni riguardanti il calcolo scientifico, per lo sviluppo di `FoReco` in Python è stato necessario fare ricorso a tre diversi pacchetti esterni di funzioni.

5.3.1 Pacchetti obbligatori

Per funzionare correttamente in Python, FoReco necessita dei seguenti pacchetti:

- `NumPy`

  `NumPy`, acronimo di *Numerical Python*, è una libreria che fornisce l'oggetto `ndarray`, che consiste in un array multidimensionale, vari oggetti derivati da questo e un assortimento di routine per le operazioni su questi array (Harris et al., 2020 [15]). In `FoReco` il pacchetto è stato utilizzato principalmente per la creazione delle matrici e per alcuni dei calcoli matriciali necessari alla riconciliazione. Una caratteristica apprezzabile di `NumPy` è che per molte delle sue funzioni attua quello che viene chiamato *GIL release*. Gran parte delle funzioni di `NumPy`, compreso il semplice prodotto tra matrici, sono infatti scritte in C, linguaggio che permette il *multithreading*. Quello che succede quando viene chiamata una di queste funzioni, è che durante l'esecuzione il GIL viene "rilasciato" e si permette l'accesso in C a tutti i diversi core nella macchina, velocizzando significativamente i tempi di esecuzione della funzione (AA. VV., 2015 [1]).

- `SciPy`

Un'altra libreria ampiamente utilizzata in `FoReco` è `SciPy` (Virtanen et al., 2020 [40]), una raccolta di algoritmi matematici, funzioni e oggetti basata su `NumPy`. Di questa libreria è stato utilizzato principalmente il pacchetto `sparse`, che implementa le matrici sparse, garantendo una sostanziale riduzione della memoria impiegata dal programma per eseguire il codice. Come `NumPy` anche `SciPy`, per molte delle operazioni tra le matrici sparse, rilascia il GIL.

- `OSQP`

  Come per la versione R di `FoReco`, anche in Python si è fatto ricorso al risolutore `OSQP` per risolvere il problema della riconciliazione non negativa. Il risolutore, infatti, oltre per il linguaggio R è disponibile anche in Python.

5.3.2 Pacchetti opzionali

Per il corretto funzionamento della riconciliazione in `FoReco` non è necessario che i seguenti pacchetti siano importati, ma per facilitare la gestione e visualizzazione dei risultati ed incrementare le performance, `FoReco` può ricorrere anche alle due seguenti librerie.

- `Pandas`

  Pandas è un pacchetto di python che fornisce alcune strutture di dati per rendere agevole la gestione di dati strutturati, e si pone l'obiettivo di diventare lo strumento di analisi e manipolazione dei dati open source più potente e flessibile disponibile in qualsiasi linguaggio (The pandas development team, 2022 [32]). Sono presenti in `FoReco` due varianti delle principali funzioni del pacchetto che fanno ricorso all'oggetto `pandas.dataframe` per facilitare l'interpretazione delle variabili restituite dalla procedura di riconciliazione.

- `Multiprocessing`

  Per migliorare l'efficienza della riconciliazione negativa quando si ha a che fare con diversi orizzonti di previsione, è stato implementato anche

il *multiprocessing*, tramite il pacchetto omonimo, per permettere a diverse istanze del risolutore `OSQP` di essere risolte contemporaneamente usando i diversi core disponibili nella macchina.

5.4 Esempio di utilizzo e confronto dei risultati tra FoReco in Python e in R

Per mostrare le funzionalità di `FoReco` in Python e la coerenza con i risultati prodotti in R, forniamo degli esempi utilizzando i dataset già citati nella documentazione di `FoReco` in R.

Per quanto riguarda la riconciliazione contemporanea facciamo quindi ricorso al dataset *htseg1* del pacchetto di R hts, che contiene dieci osservazioni di una gerarchia composta da 8 serie storiche divise in 2 livelli gerarchici, analoga a quella riportata in figura 1 del capitolo 1. Tramite la funzione auto.arima del pacchetto forecast otteniamo delle previsioni di base che verranno riconciliate con la versione R di FoReco. I risultati di questi calcoli verranno poi importati in Python, insieme alle previsioni di base che saranno riconciliate nuovamente con la versione Python di FoReco.

```
library(hts)
library(FoReco)
data <- allts(htseg1)
C <- smatrix(htseg1)[1:3,]
n <- NCOL(data)

# Previsioni di base fino a 10 orizzonti di previsione
base <- list()
for (i in 1:n) {
  base[[i]] <- forecast(auto.arima(data[,i]))
}

# Matrice delle previsioni di base (8 x n)
BASE <-NULL
for (i in 1:n) {
  BASE <- cbind(BASE, base[[i]]$mean)
}
```

```r
colnames(BASE) <- colnames(data)

# Matrice dei residui
res <- NULL
for(i in 1:n) res <- cbind(res,base[[i]]$residuals)
colnames(res) <- colnames(data)


## Utilizzo del pacchetto FoReco
# ols
Y_FoRecoR_ols <- htsrec(BASE, C=C, comb="ols")$recf
# struc
Y_FoRecoR_struc <- htsrec(BASE, C=C, comb="struc")$recf
# wls
Y_FoRecoR_wls <- htsrec(BASE, C=C, comb="wls", res=res)$recf
# shr
Y_FoRecoR_shr <- htsrec(BASE, C=C, comb="shr", res=res)$recf

# export delle previsioni di base
write.csv(BASE,"BASE.csv", row.names = TRUE)
write.csv(C,"C.csv", row.names = FALSE)
write.csv(res,"res.csv", row.names = TRUE)

# export delle previsioni riconciliate
write.csv(Y_FoRecoR_ols,"Y_FoRecoR_ols.csv", row.names =
    TRUE)
write.csv(Y_FoRecoR_struc,"Y_FoRecoR_struc.csv", row.names =
     TRUE)
write.csv(Y_FoRecoR_wls,"Y_FoRecoR_wls.csv", row.names =
    TRUE)
write.csv(Y_FoRecoR_shr,"Y_FoRecoR_shr.csv", row.names =
    TRUE)
```

Listing 5.1: Codice R per ricavare le previsioni di base del dataset htseg1, riconciliarle tramite FoReco ed esportare i dati.

```python
# importazione delle librerie
import pandas as pd
import numpy as np
from FoReco.htsrec import pddf_htsrec

```

```
# Caricamento dataframe di base
base = pd.read_csv("BASE.csv", index_col=0)
res = pd.read_csv("res.csv", index_col=0).to_numpy()
C = pd.read_csv("C.csv").to_numpy()

# Utilizzo del pacchetto FoReco in Python tramite pandas
Precf_ols=pddf_htsrec(base, "ols", C,)
Precf_struc=pddf_htsrec(base, "struc", C)
Precf_sam=pddf_htsrec(base, "sam", C, res)
Precf_shr=pddf_htsrec(base, "shr", C, res)
Precf_wls=pddf_htsrec(base, "wls", C, res)

# Importazione delle previsioni riconciliate con FoReco in R
Rrecf_ols = pd.read_csv("Y_FoRecoR_ols.csv", index_col=0)
Rrecf_struc = pd.read_csv("Y_FoRecoR_struc.csv", index_col
    =0)
Rrecf_sam = pd.read_csv("Y_FoRecoR_sam.csv", index_col=0)
Rrecf_shr = pd.read_csv("Y_FoRecoR_shr.csv", index_col=0)
Rrecf_wls = pd.read_csv("Y_FoRecoR_wls.csv", index_col=0)

# Confronto delle previsioni riconciliate

np.abs(Precf_ols.to_numpy()-Rrecf_ols.to_numpy()).sum()
# 2.589928271845565e-12

np.abs(Precf_struc.to_numpy()-Rrecf_struc.to_numpy()).sum()
# 2.76578759894619e-12

np.abs(Precf_shr.to_numpy()-Rrecf_shr.to_numpy()).sum()
# 2.0108359422010835e-12

np.abs(Precf_wls.to_numpy()-Rrecf_wls.to_numpy()).sum()
# 2.0108359422010835e-12
```

Listing 5.2: Codice Python per ottenere le previsioni riconciliate tramite FoReco, e confronto con i risultati ottenuti in R.

Per la riconciliazione nel contesto delle gerarchie temporali forniamo un esempio utilizzando la variabile *Total emergency admissions* del dataset *AE-demand* presente in `thief` che contiene il numero settimanale dei ricoverati negli ospedali del Regno Unito dal gennaio 2011 al dicembre 2014. Il codi-

ce seguente esegue una procedura analoga a quanto visto per le gerarchie contemporanee per quanto riguarda il caso delle gerarchie temporali.

```r
library(hts)
library(FoReco)
data <- allts(htseg1)
C <- smatrix(htseg1)[1:3,]
n <- NCOL(data)

# Previsioni di base fino a 10 orizzonti di previsione
base <- list()
for (i in 1:n) {
  base[[i]] <- forecast(auto.arima(data[,i]))
}

# Matrice delle previsioni di base (8 x n)
BASE <-NULL
for (i in 1:n) {
  BASE <- cbind(BASE, base[[i]]$mean)
}
colnames(BASE) <- colnames(data)

# Matrice dei residui
res <- NULL
for(i in 1:n) res <- cbind(res,base[[i]]$residuals)
colnames(res) <- colnames(data)


## Utilizzo del pacchetto FoReco
# ols
Y_FoRecoR_ols <- htsrec(BASE, C=C, comb="ols")$recf
# struc
Y_FoRecoR_struc <- htsrec(BASE, C=C, comb="struc")$recf
# wls
Y_FoRecoR_wls <- htsrec(BASE, C=C, comb="wls", res=res)$recf
# shr
Y_FoRecoR_shr <- htsrec(BASE, C=C, comb="shr", res=res)$recf
# sam
Y_FoRecoR_sam <- htsrec(BASE, C=C, comb="sam", res=res)$recf

# export delle previsioni di base
```

```r
write.csv(BASE,"BASE.csv", row.names = TRUE)
write.csv(C,"C.csv", row.names = FALSE)
write.csv(res,"res.csv", row.names = TRUE)

# export delle previsioni riconciliate
write.csv(Y_FoRecoR_ols,"Y_FoRecoR_ols.csv", row.names =
    TRUE)
write.csv(Y_FoRecoR_struc,"Y_FoRecoR_struc.csv", row.names =
     TRUE)
write.csv(Y_FoRecoR_wls,"Y_FoRecoR_wls.csv", row.names =
    TRUE)
write.csv(Y_FoRecoR_sam,"Y_FoRecoR_sam.csv", row.names =
    TRUE)
write.csv(Y_FoRecoR_shr,"Y_FoRecoR_shr.csv", row.names =
    TRUE)
```

Listing 5.3: Codice R per ricavare le previsioni di base del dataset AEdemand, riconciliarle tramite FoReco ed esportare i dati.

```python
# importazione delle librerie
import pandas as pd
import numpy as np
from FoReco.thfrec import pddf_thfrec

# Caricamento dataframe di base
base = pd.read_csv("base_vec.csv", index_col=0)
res = pd.read_csv("res.csv", index_col=0).to_numpy()

# Utilizzo del pacchetto FoReco in Python tramite pandas
Precf_ols=pddf_thfrec(base, m = 52, comb="ols", keep="recf")
Precf_struc=pddf_thfrec(base, m = 52, comb="struc", keep="
    recf")
Precf_shr=pddf_thfrec(base, m = 52, comb="shr", res=res,
    keep="recf")
Precf_wlsv=pddf_thfrec(base, m = 52, comb="wlsv", res=res,
    keep="recf")

# Importo le previsioni riconciliate con FoReco in R
Rrecf_ols = pd.read_csv("y_FoRecoR_ols.csv", index_col=0)
Rrecf_struc = pd.read_csv("y_FoRecoR_struc.csv", index_col
    =0)
```

```
Rrecf_shr = pd.read_csv("y_FoRecoR_shr.csv", index_col=0)
Rrecf_wlsv = pd.read_csv("y_FoRecoR_wlsv.csv", index_col=0)

# Controllo delle previsioni riconciliate

np.abs(Precf_ols.to_numpy()-Rrecf_ols.to_numpy().ravel()).
    sum()
# 8.44266878630151e-11

np.abs(Precf_struc.to_numpy()-Rrecf_struc.to_numpy().ravel()
    ).sum()
# 1.0084022505907342e-10

np.abs(Precf_shr.to_numpy()-Rrecf_shr.to_numpy().ravel()).
    sum()
# 9.181633231492015e-11

np.abs(Precf_wlsv.to_numpy()-Rrecf_wlsv.to_numpy().ravel()).
    sum()
# 8.894573966244934e-11
```

Listing 5.4: Codice Python per ottenere le previsioni riconciliate tramite FoReco, e confronto con i risultati ottenuti in R.

Capitolo 6

Valutazione delle performace

6.1 Confronto dell'efficienza tra Python e R

Per confrontare l'efficienza dell'esecuzione del codice in Python e in R si è fatto ricorso ad un esperimento Monte Carlo simile a quanto proposto da Wickramasuriya et al. (2019 [42]). In questa procedura vengono costruite diverse gerarchie contemporanee di dimensioni variabili che sono poi riconciliate tramite l'approccio strutturale visto nel capitolo 2.

Gli esperimenti sono stati effettuati su una macchina Windows dotata di un processore Ryzen 5 3600 Esa-core da 3.60 Ghz e 16 GB di memoria RAM.

Per questo esperimento vengono generate 12 diverse gerarchie, con un numero di livelli K che va da uno, per la più semplice, a dodici per la più complessa. Per la gerarchia con un singolo livello sono considerate tre diverse serie al livello disaggregato, a partire da questa sono poi ottenute progressivamente le gerarchie con più livelli aggiungendo ogni volta tre o quattro nodi a ognuna delle bottom time series nella gerarchia del livello precedente.

Le previsioni di base per la serie più aggregata delle diverse gerarchie sono ottenute simulando un valore da una distribuzione uniforme nell'intervallo $(1.5e^K, 2e^K)$ dove K è il numero di livelli della gerarchia. Questo valore è poi disaggregato tramite un vettore di proporzioni che sommano ad uno in quelli che vengono considerati come i valori delle bottom time

series. Il vettore delle proporzioni è ottenuto simulando dei valori da una distribuzione Gamma con parametri di forma e di scala pari a 2, che sono poi normalizzati in modo che sommino ad uno. Moltiplicando infine la matrice di aggregazione contemporanea delle gerarchie per il vettore delle *bottom time series* sono ottenute le serie dei livelli intermedi. Per rendere la struttura gerarchica inconsistente rispetto ai vincoli di aggregazione viene aggiunto un disturbo casuale tramite dei valori generati da una distribuzione uniforme nell'intervallo $(-\overline{\mathbf{y}}_{n_b}, +\overline{\mathbf{y}}_{n_b})$ dove $\overline{\mathbf{y}}_{n_b}$ è il valore medio delle bottom time series della gerarchia. Nel caso in cui dopo l'aggiunta dal disturbo casuale un valore risulta negativo esso viene impostato uguale a zero per assicurare che tutte le previsioni di base nella gerarchia siano strettamente non negative. Se dopo una procedura di riconciliazione non sono stati ottenuti valori negativi, viene aggiunto dell'ulteriore disturbo casuale alle previsioni di base finché non è registrato almeno un valore riconciliato negativo. Per ogni struttura gerarchica, 6 di questi set di previsioni di base vengono associati ad altrettanti orizzonti di previsione. L'intera procedura è ripetuta per 50 volte per le strutture gerarchiche fino ad 8 livelli e per 10 volte in quelle con 9 livelli o più. Nella tabella 6.1 è riportata nel dettaglio la struttura di ogni gerarchia e il numero medio di valori riconciliati negativi.

Nella tabella 6.2 sono riportati i tempi medi di calcolo, in secondi, necessari a FoReco in R e in Python con le impostazioni di default per completare la procedura di riconciliazione tramite l'approccio di proiezione con e senza vincoli di non negatività per ogni struttura gerarchica. Le colonne G% indicano la differenza percentuale dei tempi di calcolo calcolata come $\frac{x_R - x_{Python}}{x_R}\%$, dove x è il tempo in secondi.

Per quanto riguarda la riconciliazione senza vincoli di negatività, sono registrati dei guadagni delle tempistiche fino al 75% nelle gerarchie più piccole mentre a per le gerarchie con 7 livelli o più, il guadagno di efficienza si riduce progressivamente fino ad essere molto importante per le gerarchie più grandi. Per quanto riguarda la riconciliazione con vincoli di non negatività si presenta una situazione simile, con buoni guadagni delle performance nelle gerarchie più piccole e un peggioramento molto significativo, ma meno grave rispetto alla riconciliazione senza vincoli, per le gerarchie più grandi.

			Orizzonte di previsione					
Livelli	n	n_b	***1***	***2***	***3***	***4***	***5***	***6***
1	4	3	1,1	1,04	1,08	1,08	1,02	1,1
			28%	*26%*	*27%*	*27%*	*26%*	*28%*
2	14	10	1,24	1,42	1,34	1,28	1,4	1,26
			9%	10%	10%	9%	10%	9%
3	49	35	2,46	2,62	2,34	2,56	2,68	2,54
			5%	5%	5%	5%	5%	5%
4	171	122	8,86	8,16	8,68	8,94	8,92	8,38
			5%	5%	5%	5%	5%	5%
5	598	427	29,92	29,54	29,5	29,06	30,7	30,16
			5%	5%	5%	5%	5%	5%
6	2092	1494	107,24	106,2	105,4	104,84	106,12	108,24
			5%	5%	5%	5%	5%	5%
7	7321	5229	373,34	372,86	369,22	371,88	368,7	372,32
			5%	5%	5%	5%	5%	5%
8	25622	18301	1300,86	1307,68	1309,52	1301,16	1307,26	1299,02
			5%	5%	5%	5%	5%	5%
9	89675	64053	4582,6	4580,7	4563,5	4555,9	4591,5	4579,5
			5%	5%	5%	5%	5%	5%
10	249808	160133	10664,5	10681,6	10710	10603,5	10673,7	10669,6
			4%	4%	4%	4%	4%	4%
11	650141	400333	26575,5	26576,2	26537,1	26621,1	26518,6	26633,5
			4%	4%	4%	4%	4%	4%
12	1650974	1000833	66372,7	66454	66390,9	66538,1	66414,2	66330,6
			4%	4%	4%	4%	4%	4%

Tabella 6.1: Struttura di ogni gerarchia generata e numero medio di valori riconciliati negativi per ogni orizzonte di previsione, sotto ad ogni valore è indicata la percentuale di valori negativi sul totale.

		Senza vincoli di non negatività			Con vincoli di non negatività		
Livelli	**n**	***Python***	***R***	***G%***	***Python***	***R***	***G%***
1	*4*	0,001855	0,007454	75,1%	0,003036	0,015303	80,2%
2	*14*	0,001861	0,006923	73,1%	0,003173	0,014934	78,8%
3	*49*	0,001772	0,006463	72,6%	0,003583	0,014974	76,1%
4	*171*	0,001876	0,006659	71,8%	0,005632	0,016899	66,7%
5	*598*	0,002199	0,007193	69,4%	0,013259	0,024734	46,4%
6	*2092*	0,003707	0,009502	61,0%	0,044307	0,052932	16,3%
7	*7321*	0,011577	0,019805	41,5%	0,169748	0,16929	-0,3%
8	*25622*	0,068896	0,067161	-2,6%	0,711763	0,642328	-10,8%
9	*89675*	0,605751	0,236065	-156,6%	3,167012	2,594077	-22,1%
10	*249808*	6,635604	0,799343	-730,1%	14,13467	8,115703	-74,2%
11	*650141*	47,64766	2,481989	-1819,7%	69,16047	24,20644	-185,7%
12	*1650974*	322,2523	7,156756	-4402,8%	378,4229	64,82713	-483,7%

Tabella 6.2: Tempi di calcolo medi, in secondi, necessari a FoReco in Python e in R per riconciliare le previsioni di base con e senza vincoli di non negatività

6.2 Problematiche riscontrate in Python e possibili soluzioni

Andando ad approfondire le motivazioni per le quali è stata riscontrata una così importante perdita di efficienza per la versione Python di `FoReco` rispetto ad R, si è appurato che la quasi totalità del tempo di esecuzione del codice nelle gerarchie più grandi è impiegato dall'operazione di risoluzione del sistema lineare necessario al calcolo di $(\mathbf{U}'\mathbf{W}_h\mathbf{U})^{-1})\mathbf{U}'\hat{y}_h$ nella formula 2.18. Si sono quindi cercate delle soluzioni alternative per completarne il calcolo in delle tempistiche almeno paragonabili a quelle di R.

Nella versione Python di FoReco la procedura di risoluzione dei sistemi lineari è svolta grazie alla funzione `scipy.sparse.linalg.splu` tramite la quale viene eseguita la decomposizione LU, fattorizzazione utilizzata anche nella versione originale di `FoReco`. Questa, tra le diverse fattorizzazioni note in letteratura, nel contesto dell'inversione di matrici sparse è la preferibile, in quanto a differenza di altre fattorizzazioni che producono matrici dense, è possibile individuare una permutazione della matrice di partenza in modo che le matrici risultanti della decomposizione risultino anch'esse sparse (Nelson, 2020 [28]).

Per la risoluzione dei sistemi lineari in un contesto di sparsità è possibile seguire anche un approccio diverso dalla fattorizzazione e utilizzare dei metodi iterativi che fanno ricorso unicamente a prodotti tra matrici e vettori. In questo ambito, la libreria `scipy` fornisce le seguenti funzioni che sono state implementate in `FoReco`:

- `scipy.sparse.linalg.gmres(A,b)`: Utilizza il metodo dei residui minimi generalizzati per risolvere il sistema $\mathbf{A}x = b$

- `scipy.sparse.linalg.cg(A,b)`: Utilizza il metodo del gradiente coniugato per risolvere il sistema $\mathbf{A}x = b$

Entrambi i metodi non richiedono particolari condizioni alla matrice $\mathbf{A}$; tuttavia, nel caso di matrici non definite positive, non è assicurata la convergenza per il metodo del gradiente coniugato.

Un fattore da valutare nel contesto dei metodi iterativi, ma molto dipendente come efficacia ed efficienza dal campo di applicazione, è quello del *precondizionamento*, che consiste nel fornire alle procedure una matrice "comoda" da utilizzare al primo step delle iterazioni. In quest'ottica `scipy` mette a disposizione la funzione `scipy.sparse.linalg.spilu(A,b)` che calcola una decomposizione LU incompleta, comoda per ottenere un buon precondizionatore. Il calcolo della decomposizione LU incompleta nel nostro contesto ha prodotto delle tempistiche molto simili a quelle della decomposizione LU completa e per questo motivo, dal momento che non sono state riscontrate problematiche per ottenere la convergenza dei metodi iterativi anche senza il precondizionamento, non si è fatto ricorso a questo elemento.

Un'altra caratteristica comoda dei metodi iterativi è la possibilità di indicare manualmente una particolare "tolleranza" per la convergenza, sacrificando l'accuratezza della soluzione in favore di una maggiore efficienza. Utilizzando come metro di paragone la differenza delle soluzioni ottenute per i valori riconciliati rispetto alle stesse soluzioni ottenute con FoReco in R sono state considerati due diversi valori per il parametro `tol`, di default pari a `1e-05`, che indica la tolleranza per la convergenza delle soluzioni: il primo, indicato con HA (*high accuracy*) e pari a `1e-12`, fornisce un'accuratezza paragonabile alla funzione che risolve il sistema lineare tramite la fattorizzazione LU; il secondo, indicato con LA(*low accuracy*) e pari a `1e-04`, invece fornisce un'accuratezza ridotta in favore di una maggiore efficienza.

Come si può vedere in tabelle 6.3 l'utilizzo dei metodi iterativi fornisce un importante guadagno di efficienza rispetto alla fattorizzazione LU, in particolare il metodo del gradiente coniugato risulta sempre più efficiente anche quando si richiede un'accuratezza elevata.

Si può notare però che Il tempo di esecuzione della riconciliazione quando vengono utilizzati i metodi iterativi aumenta linearmente con il numero degli orizzonti di previsione, in quanto è necessario ripetere il calcolo della soluzione del sistema lineare per ogni vettore di previsioni da riconciliare. Questo non accade con i metodi diretti che essenzialmente possono fornire l'inversa della matrice $\mathbf{U}'\mathbf{W}_h\mathbf{U}$, con la quale è possibile risolvere tutti

n	*Risolutore*	*Orizzonti di previsione*					
		1	2	3	4	5	6
171	splu	0.0018	0.0019	0.0019	0.0019	0.0019	0.0019
	cgLA	0.0021	0.0026	0.0030	0.0034	0.0038	0.0042
	cgHA	0.0023	0.0030	0.0037	0.0044	0.0050	0.0056
	gmresLA	0.0020	0.0025	0.0028	0.0032	0.0035	0.0038
	gmresHA	0.0021	0.0026	0.0031	0.0035	0.0039	0.0044
598	splu	0.0021	0.0022	0.0022	0.0022	0.0022	0.0022
	cgLA	0.0025	0.0032	0.0039	0.0046	0.0052	0.0058
	cgHA	0.0032	0.0046	0.0059	0.0071	0.0084	0.0097
	gmresLA	0.0024	0.0030	0.0035	0.0040	0.0045	0.0050
	gmresHA	0.0033	0.0048	0.0063	0.0076	0.0090	0.0104
2092	splu	0.0036	0.0038	0.0037	0.0037	0.0037	0.0038
	cgLA	0.0038	0.0052	0.0064	0.0077	0.0090	0.0102
	cgHA	0.0055	0.0086	0.0116	0.0146	0.0176	0.0206
	gmresLA	0.0036	0.0049	0.0060	0.0071	0.0082	0.0093
	gmresHA	0.0062	0.0100	0.0137	0.0174	0.0211	0.0247
7231	splu	0.0111	0.0116	0.0116	0.0116	0.0118	0.0118
	cgLA	0.0091	0.0127	0.0158	0.0190	0.0221	0.0256
	cgHA	0.0143	0.0230	0.0313	0.0396	0.0480	0.0567
	gmresLA	0.0099	0.0145	0.0186	0.0227	0.0268	0.0310
	gmresHA	0.0216	0.0380	0.0544	0.0704	0.0864	0.1003
25622	splu	0.0672	0.0683	0.0689	0.0694	0.0694	0.0704
	cgLA	0.0319	0.0443	0.0561	0.0684	0.0807	0.0936
	cgHA	0.0558	0.0933	0.1286	0.1643	0.2013	0.2393
	gmresLA	0.0475	0.0749	0.1016	0.1282	0.1544	0.1816
	gmresHA	0.1340	0.2463	0.3539	0.4684	0.5764	0.6895
89675	splu	0.5992	0.6067	0.6044	0.6056	0.6093	0.6094
	cgLA	0.1875	0.3065	0.4184	0.5670	0.6887	0.8141
	cgHA	0.4633	0.8561	1.2051	1.4918	2.0033	2.3032
	gmresLA	0.7031	1.4144	2.0513	2.6834	3.3725	3.9979
	gmresHA	3.2325	6.2403	9.1411	12.2879	15.2320	18.1421
249808	splu	6.6094	6.6256	6.6310	6.6316	6.6580	6.6581
	cgLA	0.8340	1.4525	2.0858	2.7052	3.3198	3.9363
	cgHA	2.4013	4.5634	6.6919	8.8721	11.0237	13.1796
	gmresLA	4.6595	9.2353	13.3739	17.7821	22.2131	26.6543
	gmresHA	22.8208	42.7932	67.5658	84.3751	105.4181	126.6810
650141	splu	47.5784	47.4740	47.4795	47.4742	48.3802	47.4997
	cgLA	3.4127	6.2766	9.1228	11.9405	14.7920	17.7900
	cgHA	11.0120	23.1462	34.1045	42.2907	52.6528	63.0667
	gmresLA	32.5188	66.9248	96.4228	128.4803	160.5361	193.9918
	gmresHA	166.6801	330.8386	496.5140	671.1352	840.4435	999.3929
1650974	splu	323.2731	323.3808	323.0799	321.4506	320.9258	321.4038
	cgLA	17.0949	28.8414	41.6500	54.7607	68.4760	81.4999
	cgHA	51.1987	100.7665	150.0561	198.8079	247.2734	294.3112

Tabella 6.3: Tempi di calcolo medi, in secondi, necessari al calcolo della riconciliazione utilizzando diversi risolutori lineari

n	*Numero Core*	*Orizzonti di previsione(h)*					
		1	*2*	*3*	*4*	*5*	*6*
25622	1	0.0319	0.0443	0.0561	0.0684	0.0807	0.0936
	h	0.0314	1.0879	1.1069	1.1453	1.2522	1.2793
89675	1	0.1875	0.3065	0.4184	0.5670	0.6887	0.8141
	h	0.1930	1.2629	1.2800	1.4086	1.5387	1.6775
249808	1	0.8340	1.4525	2.0858	2.7052	3.3198	3.9363
	h	0.8268	2.1927	2.4849	2.8298	3.2403	3.7472
650141	1	3.4127	6.2766	9.1228	11.9405	14.7920	17.7900
	h	3.4152	5.4675	6.8491	8.6183	10.7806	13.1975
1650974	1	14.0749	28.8414	41.6500	54.7607	68.4760	81.4999
	h	15.6376	24.2451	31.6373	40.3349	52.4509	64.4624

Tabella 6.4: Tempi necessari al calcolo della riconciliazione utilizzando il metodo del gradiente coniugato con o senza multiprocessing

i sistemi lineari tramite una semplice moltiplicazione tra matrici, per questo motivo tramite la fattorizzazione LU il tempo di calcolo dipende solo marginalmente dal numero di orizzonti di previsione da riconciliare.

Per risolvere parzialmente questo problema, dal momento che la soluzione di ogni sistema lineare avviene indipendentemente dagli altri, può tornare utile la parallelizzazione dei processi per risolvere diversi sistemi lineari contemporaneamente.

Come si può vedere dalla tabella 6.4, fare ricorso alla parallelizzazione per le gerarchie più piccole non risulta svantaggioso in quanto il peso computazionale dell'operazione di inizializzazione dei diversi processi è significativo, ma per quanto riguarda le gerarchie più grandi si ha un buon guadagno di efficienza.

È possibile fare ricorso alla parallelizzazione anche durante l'utilizzo del risolutore `OSQP`, che, come i metodi iterativi, ripete la procedura di risoluzione del problema di ottimizzazione quadratica indipendentemente per ogni orizzonte di previsione.

Capitolo 7

Previsione dei flussi turistici australiani

Per verificare le performance delle procedure di riconciliazione implementate si è fatto ricorso al dataset proveniente dal progetto governativo australiano '*National visitor survey*' che misura i flussi turistici domestici in termini di arrivi e di notti trascorse nelle strutture turistiche. I dati sono stati raccolti tramite interviste telefoniche da quasi 120000 cittadini australiani di età non inferiore a 15 anni.

7.1 Descrizione del dataset

Il dataset consiste in 555 serie storiche mensili che partono dal gennaio 1998 e arrivano al dicembre 2016. Queste serie definiscono una gerarchia raggruppata nella quale il totale indica il numero mensile di notti spese dai cittadini australiani lontani da casa. Il valore totale è poi disaggregato nelle restanti serie tramite posizione geografica e motivazioni del viaggio.

Per quanto riguarda le motivazioni del viaggio sono considerate quattro diverse categorie: vacanze, visite ad amici e parenti (indicato nei grafici con VAP), lavoro e altro. Invece, dal punto di vista geografico e amministrativo, l'Australia è divisa in 76 regioni che sono raggruppate in 27 diverse zone che sono poi a loro volta raggruppate in 7 stati. Considerando queste due

diverse divisioni, la gerarchia raggruppata avrà quindi un totale di sette diversi livelli.

In tabella 7.1 è riportata la struttura gerarchica della suddivisione geografica dell'Australia. Dalla tabella possiamo notare che sono presenti sei diverse zone turistiche che contengono al loro interno una sola regione. Per questo motivo la gerarchia può essere definita come sbilanciata e nella procedura di modellazione si farà quindi ricorso alle soluzioni proposte nel capitolo 4.2.1. In particolare, verranno escluse dalla gerarchia un totale di 30 serie storiche ridondanti.

Le diverse serie storiche mensili della gerarchia presentano differenze molto significative in termini di scala, trend e stagionalità. Per quasi tutte le serie più aggregate (Figura 7.1) possiamo notare una forte componente stagionale con un picco in corrispondenza del mese di gennaio di ogni anno, durante il quale si ha il periodo delle vacanze estive in Australia. Si può infatti notare anche che in quel mese le notti trascorse in strutture turistiche per motivi lavorativi presentano un picco verso il basso. La componente stagionale risulta sempre meno netta valutando i livelli più disaggregati della gerarchia (Figura 7.2) che risultano, com'è da attendersi, molto più variabili delle serie aggregate. Possiamo poi notare anche molte serie storiche con diversi valori pari a zero, sia per le bottom time series, che per alcune serie a livelli più aggregati come il tipo di viaggio per le diverse zone e le diverse regioni.

Dal momento che i dati a disposizione sono a frequenza mensile, risulta più comodo, per valutare eventuali trend, aggregare i valori mensili ad una frequenza annuale. Facendo questo possiamo vedere come le serie storiche dei livelli aggregati (Figura 7.3) presentano degli andamenti maggiormente regolari. In particolare notiamo che le serie storiche delle notti trascorse per motivi del viaggio sono più stabili mentre quelle relative al totale o ai diversi stati hanno degli andamenti di natura diversa, ma comunque identificabili. Per quanto riguarda invece le serie più disaggregate (Figura 7.4) possiamo notare una variabilità ridotta rispetto ai dati mensili. Risulta comunque difficile individuare particolari trend e l'andamento si dimostra anche in questo caso molto irregolare.

Serie	Nome	Label
Total		
1	Australia	Total
Stati		
2	New South Wales (NSW)	A
3	Victoria (VIC)	B
4	Queensland (QLD)	C
5	South Australia (SA)	D
6	Western Australia (WA)	E
7	Tasmania (TAS)	F
8	Northern Territory (NT)	G
Zone		
9	Metro NSW	AA
10	Nth Coast NSW	AB
11	**Sth Coast NSW**	**AC**
12	Sth NSW	AD
13	Nth NSW	AE
14	**ACT**	**AF**
15	Metro VIC	BA
16	**West Coast VIC**	**BB**
17	East Coast VIC	BC
18	Nth East VIC	BD
19	Nth West VIC	BE
20	Metro QLD	CA
21	Central Coast QLD	CB
22	Nth Coast QLD	CC
23	Inland QLD	CD
24	Metro SA	DA
25	Sth Coast SA	DB
26	Inland SA	DC
27	West Coast SA	DD
28	West CoastWA	EA
29	**Nth WA**	**EB**
30	**SthWA**	**EC**
31	**Sth TAS**	**FA**
32	Nth East TAS	FB
33	Nth West TAS	FC
34	Nth Coast NT	GA
35	Central NT	GB
Regioni		
36	Sydney	AAA
37	Central Coast	AAB
38	Hunter	ABA
39	North Coast NSW	ABB
40	Northern Rivers Tropical NSW	ABC
41	South Coast	ACA
42	Snowy Mountains	ADA
43	Capital Country	ADB
44	The Murray	ADC
45	Riverina	ADD
46	Central NSW	AEA
47	New England North West	AEB
48	Outback NSW	AEC
49	Blue Mountains	AED
50	Canberra	AFA
51	Melbourne	BAA
52	Peninsula	BAB
53	Geelong	BAC
54	Western	BBA
Continuo delle regioni		
55	Lakes	BCA
56	Gippsland	BCB
57	Phillip Island	BCC
58	Central Murray	BDA
59	Goulburn	BDB
60	High Country	BDC
61	Melbourne East	BDD
62	Upper Yarra	BDE
63	MurrayEast	BDF
64	Wimmera+Mallee	BEA
65	Western Grampians	BEB
66	Bendigo Loddon	BEC
67	Macedon	BED
68	Spa Country	BEE
69	Ballarat	BEF
70	Central Highlands	BEG
71	Gold Coast	CAA
72	Brisbane	CAB
73	Sunshine Coast	CAC
74	Central Queensland	CBA
75	Bundaberg	CBB
76	Fraser Coast	CBC
77	Mackay	CBD
78	Whitsundays	CCA
79	Northern	CCB
80	Tropical North Queensland	CCC
81	Darling Downs	CDA
82	Outback	CDB
83	Adelaide	DAA
84	Barossa	DAB
85	Adelaide Hills	DAC
86	Limestone Coast	DBA
87	Fleurieu Peninsula	DBB
88	Kangaroo Island	DBC
89	Murraylands	DCA
90	Riverland	DCB
91	Clare Valley	DCC
92	Flinders Range and Outback	DCD
93	Eyre Peninsula	DDA
94	Yorke Peninsula	DDB
95	Australia's Coral Coast	EAA
96	Experience Perth	EAB
97	Australia's SouthWest	EAC
98	Australia's North West	EBA
99	Australia's Golden Outback	ECA
100	Hobart and the South	FAA
101	East Coast	FBA
102	Launceston, Tamar and the North	FBB
103	North West	FCA
104	WildernessWest	FCB
105	Darwin	GAA
106	Kakadu Arnhem	GAB
107	Katherine Daly	GAC
108	Barkly	GBA
109	Lasseter	GBB
110	Alice Springs	GBC
111	MacDonnell	GBD

Tabella 7.1: Suddivisione geografica dell'Australia in stati, zone e regioni (Tratta da Wickramasuriya et al., 2019 [42]). In grassetto sono state evidenziate le zone formate da una sola regione

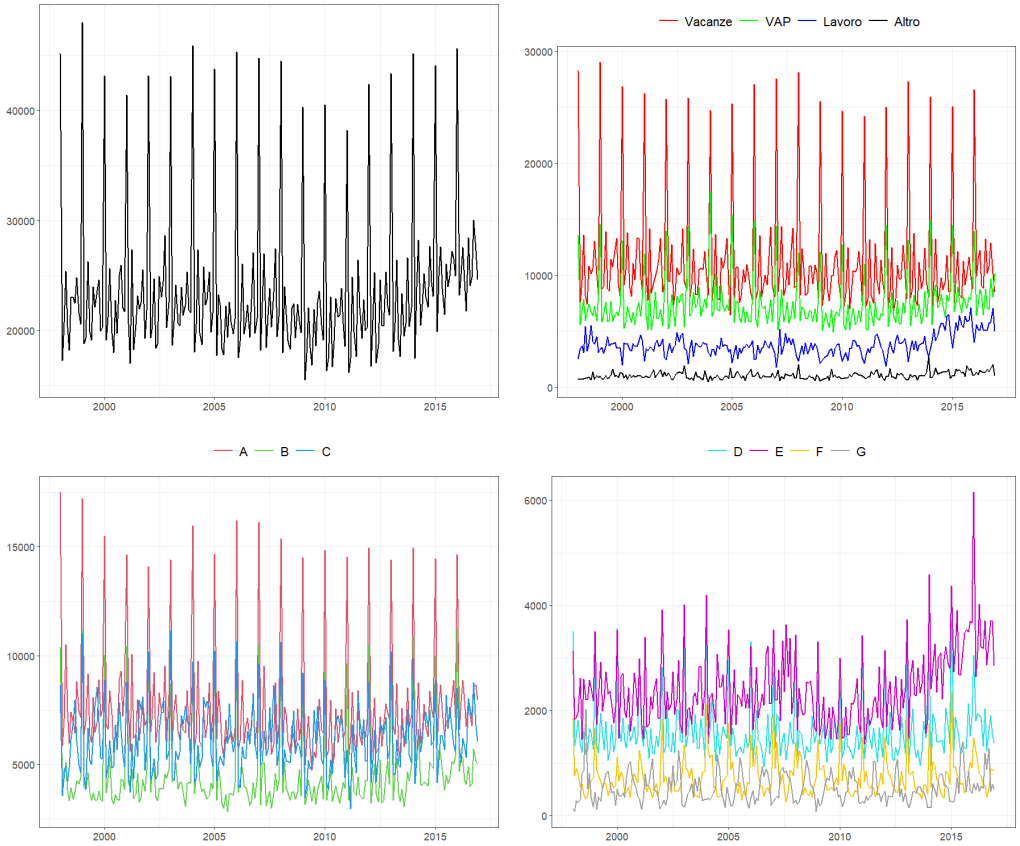


Figura 7.1: Andamento mensile delle notti trascorse nelle strutture turistiche australiane. In alto a sinistra abbiamo il totale, in alto a destra abbiamo una divisione delle serie storiche per motivo del viaggio, in basso per gli stati.

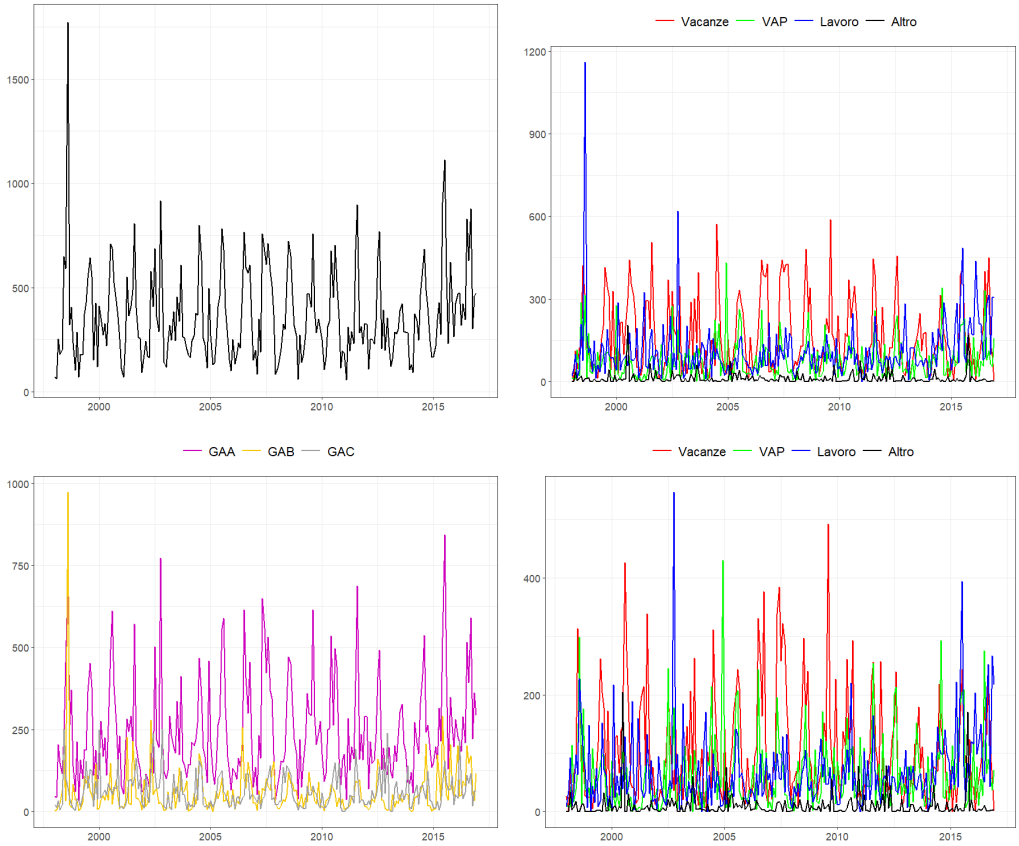


Figura 7.2: Andamento mensile delle notti trascorse nelle strutture turistiche australiane. In alto a sinistra abbiamo il totale, in alto a destra abbiamo una divisione delle serie storiche per motivo del viaggio, in basso per gli stati.

Figura 7.3: Andamento annuale delle notti trascorse nelle strutture turistiche australiane. In alto a sinistra abbiamo il totale, in alto a destra abbiamo una divisione delle serie storiche per motivo del viaggio, in basso per gli stati.

Figura 7.4: Andamento annuale delle notti trascorse nelle strutture turistiche australiane per la zona "Nth Coast NT"(GA) secondo diverse variabili di interesse. In alto a sinistra abbiamo l'andamento totale del numero di notti trascorse nelle strutture turistiche, in alto a destra abbiamo la divisione per motivo del viaggio, in basso a sinistra abbiamo la divisione per le regioni, e in basso a destra abbiamo la divisione per i motivi del viaggio per la regione "Darwin"(GAA).

7.2 Calcolo delle previsioni riconciliate

L'esperimento di previsione considerato per l'applicazione delle tecniche di riconciliazione contemporanea è di tipo *rolling window*. In particolare, come *training set* viene utilizzata una finestra mobile di ampiezza fissa pari a 144 mesi con un passo di un mese. Dal momento che le osservazioni dal gennaio 1998 al dicembre 2016 forniscono un totale di 228 valori mensili e che si è deciso di fissare un orizzonte di previsione pari ad un anno, avremo a disposizione 73 origini di previsioni diverse. Le previsioni di base sono sempre ottenute adattando un modello ARIMA alle diverse serie storiche tramite la funzione `auto.arima` del pacchetto `Forecast` di R (Hyndman et al., 2020 [18]) con le impostazioni di default.

Alle previsioni così ottenute sono state applicate tutte le tecniche di riconciliazione per le previsioni contemporanee che offre il pacchetto `FoReco`, ad eccezione della tecnica SAM, che non può essere utilizzata, in quanto il numero di osservazioni a disposizione (228) è di gran lunga inferiore a quello delle serie che compongono la gerarchia (525).

Per valutare invece la riconciliazione di gerarchie temporali, il dataset viene aggregato temporalmente al fine di ottenere i dataset anche con le serie a frequenza bimestrale, trimestrale, quadrimestrale, semestrale e annuale. In questo caso, per facilitare la stima nei modelli nelle serie alle frequenze più basse si è deciso di fare ricorso ad una *rolling window* con ampiezza fissa maggiore pari a 168 mesi, con passo di un mese, ottenendo in questo caso 49 origini di previsioni diverse. In questo modo i modelli per le previsioni annuali saranno adattati ad un campione di 14 osservazioni. Fissando ancora ad un anno l'orizzonte di previsione, per ogni origine di previsione saranno stimati quindi 12 previsioni per le serie mensili, 6 per quelle bimestrali, 4 per le trimestrali, 3 per le quadrimestrali, 2 per le semestrali e 1 previsione per i valori annuali.

Anche in questo caso saranno applicate tutte le tecniche di riconciliazione temporale offerte `FoReco` ad eccezione dell'approccio SAM, in quanto le osservazioni annuali a disposizione (19) sono inferiori al numero di nodi di ogni gerarchia temporale (28).

Nelle tabelle che seguono sono indicate come bts e uts sempre le serie

relative alla gerarchia come esposta nel capitolo 7.1. Tuttavia deve essere tenuto presente che nel caso delle gerarchie temporali le *bottom time series* sono per definizione sempre le serie mensili, mentre le *upper time series* sono quelle alle altre frequenze. In questo senso nell'analisi delle gerarchie temporali la distinzione nei grafici tra uts e bts è volta solamente a distinguere le performance delle tecniche di riconciliazione temporale per serie di partenza con caratteristiche diverse. Inoltre, dal momento che la riconciliazione avviene solamente rispetto alla dimensione temporale, in questo caso i vincoli contemporanei non sono rispettati.

7.2.1 Riconciliazione delle previsioni negative

Vista la natura dei dati da prevedere, soprattutto per le serie ai livelli più disaggregati, è legittimo aspettarsi diverse previsioni con valori negativi. All'atto pratico effettivamente questi rendono complicata l'analisi economica di un fenomeno, come le notti passate in strutture turistiche, nel quale un valore negativo, oltre a non poter essere osservato, non è di facile interpretazione.

Per risolvere questo problema possono essere seguite due diverse strade: fare ricorso a delle tecniche di previsione che rispettino anche i vincoli di non negatività, oppure utilizzare le tecniche di riconciliazione presentate nel capitolo 4.5 che assicurano dei risultati non negativi.

Valutando queste due strade bisogna tenere conto, però, che le tecniche di riconciliazione possono portare a dei risultati negativi anche per previsioni di base positive. Per esempio, nel caso in analisi, per la previsione della gerarchia contemporanea sono state osservate una media di 1.6 previsioni di base negative per ogni 3600 previsioni. Dopo aver applicato le tecniche di riconciliazione il numero di valori negativi sale significativamente ottenendo una media di 35.6 valori negativi ogni 3600 previsioni. Per questo motivo si è deciso di fare ricorso alle soluzioni per la riconciliazione non negativa e durante l'analisi si è valutata l'accuratezza delle previsioni riconciliate anche in base a queste tecniche.

7.3 Analisi dei risultati

Per l'analisi dell'accuratezza delle previsioni ottenute dalle procedure di riconciliazione contemporanea e temporale, riprendendo quanto proposto da Wickramasuya et al. (2020 [43]), si è fatto ricorso al rapporto tra il RMSE medio calcolato per le previsioni riconciliate e la stessa quantità calcolata rispetto alle previsioni di base (*RMSE ratio*). In particolare, dei valori maggiori di uno indicheranno quindi un peggioramento delle performance rispetto alle previsioni di base, mentre un valore minore di 1 indicherà un miglioramento. Nelle tabelle le sigle per indicare le procedure corrispondono a quelle indicate nei capitoli 2.2.3 e 3.2.2, mentre per quanto riguarda le procedure di non negatività, con OSQP si fa riferimento alla procedura che utilizza il pacchetto omonimo, mentre con SNTZ (set negative to zero) si fa riferimento alla procedura che pone uguali a zero i valori riconciliati negativi e successivamente ripete l'aggregazione.

Nella riconciliazione della gerarchia contemporanea possiamo notare come le procedure di riconciliazione producano nella quasi totalità dei casi un incremento dell'accuratezza fatta eccezione per la procedura bottom up che, basandosi unicamente sulle previsioni di bottom time series nel nostro caso molto variabili, ha delle performance peggiori rispetto a quelle delle previsioni di base. Valutando invece le altre procedure di riconciliazione vediamo come la procedura shr fornisce in tutti i casi un'accuratezza di gran lunga migliore rispetto alle altre, probabilmente dovuta al fatto che tra le serie è presente una forte correlazione che deve essere considerata. Apprezzabili anche le performance della procedura struc che, soprattutto per le previsioni ad un passo in avanti, fornisce un buon incremento dell'accuratezza dei risultati anche senza fare ricorso ai residui ottenuti dalla stima del modello usato per calcolare le previsioni di base.

Possiamo anche vedere come l'utilizzo di tecniche di non negatività, oltre che a facilitare l'interpretazione dei risultati, nella maggior parte dei casi garantisce anche un incremento dell'accuratezza rispetto alle procedure di riconciliazione nelle quali non si è considerato questo vincolo.

Tabella 7.2: RMSE ratio per le previsioni riconciliate secondo la gerarchia contemporanea 12 passi in avanti con relativa rappresentazione grafica

Non negatività	***Procedura***	***Livello della gerarchia***			
		Medio	***tot***	***uts***	***bts***
Nessuna	*Base*	1	1	1	1
	bu	1.015	1	1.016	1
	ols	0.986	0.998	0.986	0.998
	struc	0.98	0.99	0.98	0.99
	wls	0.994	0.982	0.995	0.982
	shr	0.721	0.774	**0.719**	0.774
Osqp	*bu*	1.015	1	1.016	1
	ols	0.986	0.998	0.986	0.998
	struc	0.98	0.99	0.98	0.99
	wls	0.994	0.982	0.995	0.982
	shr	0.723	**0.773**	0.721	**0.773**
Sntz	*bu*	1.015	1	1.016	1
	ols	0.978	0.996	0.978	0.996
	struc	0.979	0.989	0.979	0.989
	wls	0.994	0.982	0.995	0.982
	shr	**0.72**	0.774	**0.719**	0.774

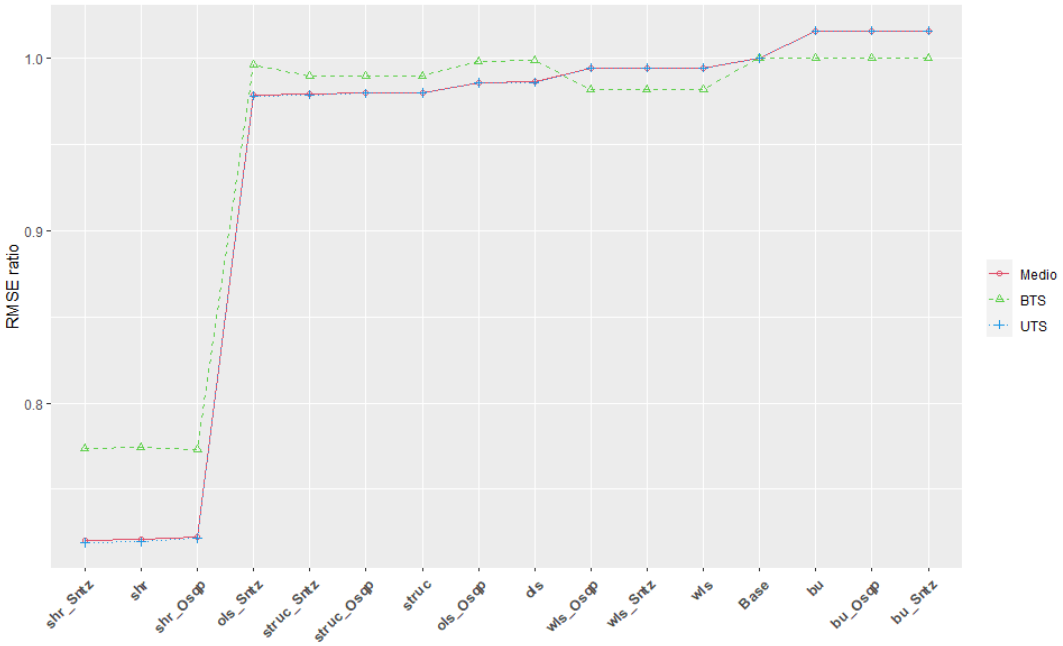
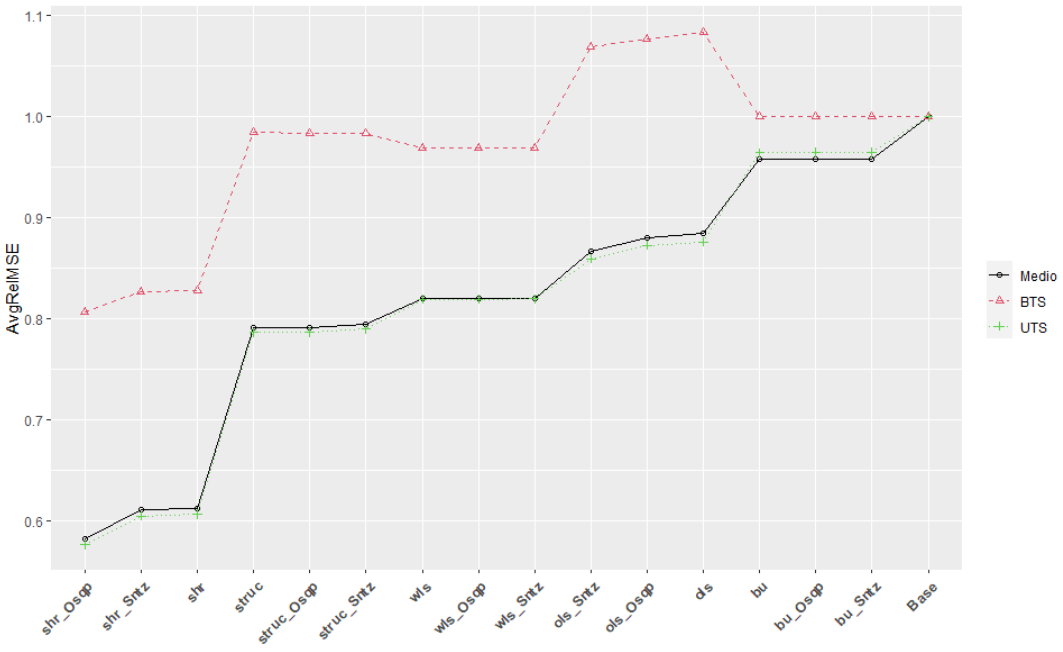


Tabella 7.3: RMSE ratio per le previsioni riconciliate secondo la gerarchia contemporanea 1 passo in avanti con relativa rappresentazione grafica

Non negativita	***Procedura***	***Livello della gerarchia***			
		Medio	***tot***	***uts***	***bts***
Nessuna	*Base*	1	1	1	1
	bu	0.958	1	0.965	1
	ols	0.884	1.084	0.876	1.084
	struc	0.792	0.985	0.787	0.985
	wls	0.82	0.969	0.82	0.969
	shr	0.613	0.828	0.607	0.828
Osqp	*bu*	0.958	1	0.965	1
	ols	0.881	1.077	0.873	1.077
	struc	0.792	0.983	0.787	0.983
	wls	0.82	0.969	0.82	0.969
	shr	**0.581**	**0.806**	**0.575**	**0.806**
Sntz	*bu*	0.958	1	0.965	1
	ols	0.868	1.069	0.859	1.069
	struc	0.795	0.984	0.79	0.984
	wls	0.82	0.969	0.82	0.969
	shr	0.611	0.827	0.605	0.827



Passando all'analisi delle gerarchie temporali, possiamo vedere come anche in questo caso le procedure di riconciliazione, sempre ad eccezione della *bottom up*, producono in media un miglioramento dell'accuratezza delle previsioni, con la procedura shr che nella grande maggioranza dei casi porta le performance migliori. Questo però non vale per tutte le serie a tutti i livelli di aggregazione temporale, in quanto l'accuratezza delle previsioni riconciliate delle serie semestrali per le upper time series della gerarchia risulta sempre più bassa rispetto alle previsioni di base. Anche in questo caso, possiamo vedere come l'introduzione dei vincoli di non negatività non compromette l'accuratezza delle previsioni ma anzi in molti casi la migliora. Deve essere segnalato che però per diversi orizzonti di previsione la procedura OSQP non è riuscita a raggiungere la convergenza.

non negatività	***Procedura***	***Livello della gerarchia***						
		Totale	*Annuali*	*Semestr.*	*Quadrim.*	*Trimestr.*	*Bimens.*	*Mensili*
Nessuna	*Base*	1	1	**1**	1	1	1	1
	bu	1.2845	1.5185	2.3726	1.4939	1.1646	1.026	1
	ols	0.8972	1.0263	1.3052	0.9455	0.7569	0.7612	0.8737
	struc	1.0078	1.159	1.6342	1.0885	0.8623	0.8344	0.9096
	wlsv	0.8351	0.985	1.1439	0.8529	0.6923	0.7112	0.848
	wlsh	0.6996	0.9264	1.1161	0.764	0.5751	0.5517	0.5828
	acov	0.6846	1.0426	1.209	0.7511	0.5686	0.4966	0.5037
	strar1	1.0109	1.195	1.6202	1.0903	0.8597	0.8302	0.9075
	sar1	0.8377	1.0007	1.1374	0.8554	0.6935	0.7094	0.8476
	har1	0.6926	0.9322	1.0988	0.7555	0.5672	0.5424	0.5718
	shr	**0.5684**	0.9828	1.0128	**0.633**	**0.4716**	0.4023	0.393
osqp	*bu*	1.2845	1.5184	2.3726	1.4939	1.1646	1.026	1
	ols	0.8972	1.0263	1.3052	0.9455	0.7569	0.7612	0.8737
	struc	1.0078	1.159	1.6342	1.0885	0.8623	0.8344	0.9096
	*wlsv**	0.8344	0.9805	1.1493	0.8525	0.6902	0.7127	0.8497
	*wlsh**	0.7002	**0.9204**	1.1228	0.7656	0.5793	0.5555	0.5888
	*acov**	0.6671	0.944	1.1503	0.766	0.5913	0.5116	0.5245
	strar1	1.0109	1.195	1.6202	1.0903	0.8597	0.8302	0.9075
	*sar1**	0.8381	0.9947	1.1375	0.862	0.6933	0.7094	0.8483
	*har1**	0.6933	0.9257	1.1075	0.7594	0.5723	0.5482	0.5803
	*shr**	0.5732	0.9838	1.0162	0.6506	0.4812	0.4104	0.3968
sntz	*bu*	1.2845	1.5184	2.3726	1.4939	1.1646	1.026	1
	ols	0.8972	1.0263	1.3052	0.9454	0.7569	0.7612	0.8737
	struc	1.0078	1.159	1.6343	1.0885	0.8623	0.8344	0.9096
	wlsv	0.8351	0.985	1.1438	0.8529	0.6923	0.7112	0.848
	wlsh	0.6997	0.9269	1.1165	0.7642	0.5752	0.5517	0.5828
	acov	0.6844	1.0482	1.21	0.7495	0.5683	0.4956	0.5027
	strar1	1.0109	1.195	1.6202	1.0903	0.8597	0.8302	0.9075
	sar1	0.8377	1.0007	1.1374	0.8553	0.6935	0.7094	0.8476
	har1	0.6927	0.9327	1.0993	0.7557	0.5672	0.5424	0.5717
	shr	**0.5684**	0.9836	1.0133	0.6332	**0.4716**	**0.4022**	**0.3929**

Tabella 7.4: RMSE ratio per le previsioni riconciliate temporalmente per tutte le serie della gerarchia contemporanea

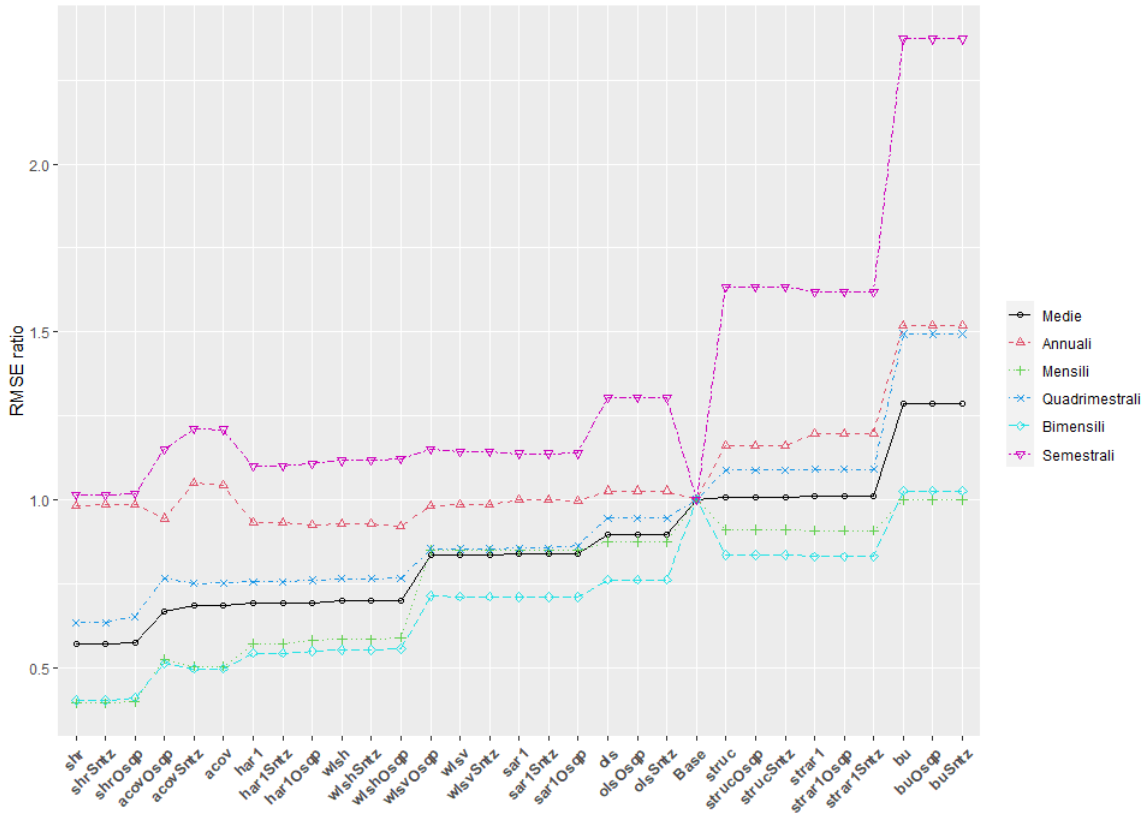


Figura 7.5: rappresentazione grafica dell'RMSE ratio per le previsioni riconciliate temporalmente per tutte le serie della gerarchia contemporanea

non negatività	***Procedura***	***Livello della gerarchia***						
		Totale	*Annuali*	*Semestr.*	*Quadrim.*	*Trimestr.*	*Bimens.*	*Mensile*
Nessuna	Base	1	1	**1**	1	1	1	1
	bu	1.2889	1.5383	2.4391	1.5104	1.169	1.0245	1
	ols	0.8965	1.028	1.3253	0.9486	0.7538	0.7578	0.8731
	struc	1.0095	1.1662	1.6712	1.0967	0.8615	0.8321	0.9095
	wlsv	0.8332	0.9865	1.1548	0.853	0.6877	0.707	0.8471
	wlsh	0.695	0.9244	1.1261	0.7623	0.5689	0.5455	0.5789
	acov	0.6796	1.0476	1.2219	0.7471	0.563	0.4897	0.4992
	strar1	1.0125	1.202	1.6551	1.0985	0.8588	0.8279	0.9074
	sar1	0.8359	1.0026	1.1475	0.8557	0.6891	0.7053	0.8467
	har1	0.6879	0.9301	1.1074	0.7535	0.5608	0.5361	0.5677
	shr	0.5622	0.9904	1.0198	**0.6296**	**0.465**	**0.3952**	**0.388**
osqp	bu	1.2889	1.5383	2.439	1.5104	1.169	1.0245	1
	ols	0.8965	1.028	1.3253	0.9486	0.7538	0.7578	0.8731
	struc	1.0095	1.1662	1.6712	1.0967	0.8615	0.8321	0.9095
	wlsv*	0.8324	0.9819	1.1605	0.8527	0.6857	0.7085	0.8488
	wlsh*	0.6955	**0.9177**	1.1329	0.7638	0.573	0.5493	0.585
	acov*	0.6649	0.9766	1.1872	0.7595	0.5783	0.5018	0.5238
	strar1	1.0125	1.202	1.6551	1.0985	0.8588	0.8279	0.9074
	sar1*	0.8363	0.9962	1.1476	0.8624	0.6888	0.7052	0.8474
	har1*	0.6885	0.9227	1.1163	0.7574	0.5658	0.5419	0.5764
	shr*	**0.561**	0.9916	1.0207	0.6334	0.466	0.3978	0.3921
sntz	bu	1.2889	1.5383	2.439	1.5104	1.169	1.0245	1
	ols	0.8965	1.0279	1.3253	0.9486	0.7538	0.7578	0.8731
	struc	1.0095	1.1662	1.6712	1.0967	0.8615	0.8321	0.9095
	wlsv	0.8332	0.9865	1.1548	0.853	0.6877	0.707	0.8471
	wlsh	0.695	0.9244	1.1261	0.7623	0.5689	0.5455	0.5789
	acov	0.6797	1.0478	1.2219	0.7471	0.563	0.4896	0.4991
	strar1	1.0125	1.202	1.6551	1.0985	0.8588	0.8279	0.9074
	sar1	0.8359	1.0026	1.1475	0.8557	0.6891	0.7053	0.8467
	har1	0.6879	0.9301	1.1074	0.7535	0.5608	0.5361	0.5677
	shr	0.5622	0.9905	1.0198	**0.6296**	**0.465**	**0.3952**	**0.388**

Tabella 7.5: RMSE ratio per le previsioni riconciliate temporalmente per le *upper time series* della gerarchia contemporanea

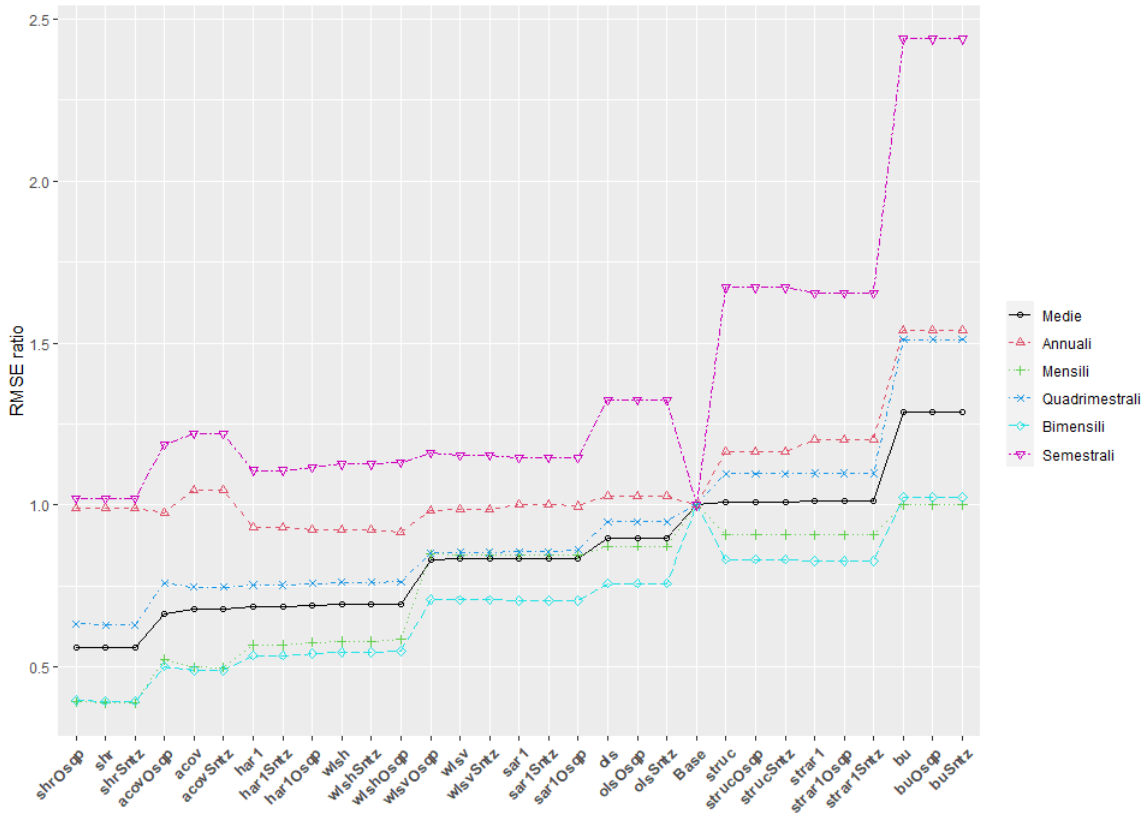


Figura 7.6: Rappresentazione grafica dell'RMSE ratio per le previsioni riconciliate temporalmente per le *upper time series* della gerarchia contemporanea

non negatività	***Procedura***	***Livello della gerarchia***						
		Totale	*Annuali*	*Semestr.*	*Quadrim.*	*Trimestr.*	*Bimens.*	*Mensile*
Nessuna	*Base*	1	1	1	1	1	1	1
	bu	1.1272	1.0232	1.3965	1.203	1.1442	1.0974	1
	ols	0.9212	0.927	1.023	0.9264	0.8874	0.8959	0.9021
	struc	0.9479	0.915	1.0861	0.9625	0.9268	0.9274	0.9157
	wlsv	0.9007	0.9147	0.9965	0.8984	0.863	0.8765	0.8905
	wlsh	0.8531	0.9109	1.0001	0.8641	0.8235	0.8023	0.7576
	acov	0.8368	0.9165	1.0202	0.8528	0.7947	0.7818	0.6926
	strar1	0.9494	0.9367	1.0845	0.9606	0.9254	0.9236	0.9136
	sar1	0.8982	0.9158	0.9926	0.8947	0.8601	0.8727	0.8887
	har1	0.8486	**0.911**	0.9951	0.8581	0.8185	0.7963	0.752
	shr	0.7765	0.9248	**0.9581**	0.78	0.7378	0.6865	0.6114
osqp	*bu*	1.1269	1.0228	1.3957	1.2027	1.1439	1.0971	0.9999
	ols	0.9211	0.927	1.0231	0.9264	0.8873	0.8958	0.9019
	struc	0.9478	0.915	1.0861	0.9625	0.9268	0.9273	0.9156
	wlsv	0.9005	0.9146	0.9964	0.8983	0.8628	0.8764	0.8903
	*Wlsh**	0.8557	0.9186	1.0053	0.8653	0.8253	0.805	0.7585
	*acov**	0.7879	0.914	0.963	0.8124	0.7478	0.7089	0.6364
	strar1	0.9494	0.9367	1.0846	0.9606	0.9253	0.9235	0.9135
	sar1	0.8981	0.9157	0.9926	0.8946	0.8599	0.8725	0.8885
	har1	0.8503	0.9179	0.9998	0.8599	0.8193	0.7963	0.7508
	*Shr**	**0.7727**	0.9243	0.9626	**0.7773**	**0.7327**	**0.6837**	**0.6047**
sntz	*bu*	1.1269	1.0228	1.3957	1.2027	1.1439	1.0971	0.9999
	ols	0.9211	0.9269	1.0232	0.9264	0.8873	0.8958	0.902
	struc	0.9478	0.9149	1.0862	0.9625	0.9268	0.9273	0.9156
	wlsv	0.9005	0.9146	0.9965	0.8983	0.8628	0.8764	0.8903
	wlsh	0.855	0.9181	1.005	0.8659	0.8244	0.8026	0.7566
	acov	0.8289	0.9923	1.0297	0.8348	0.7842	0.7386	0.6602
	strar1	0.9494	0.9366	1.0846	0.9607	0.9253	0.9235	0.9135
	sar1	0.8981	0.9157	0.9926	0.8946	0.8599	0.8725	0.8886
	har1	0.8507	0.919	1.0005	0.8602	0.8195	0.7965	0.7509
	shr	0.7779	0.9357	0.9639	0.7814	0.7378	0.6851	0.6084

Tabella 7.6: RMSE ratio per le previsioni riconciliate temporalmente per le *upper time series* della gerarchia contemporanea

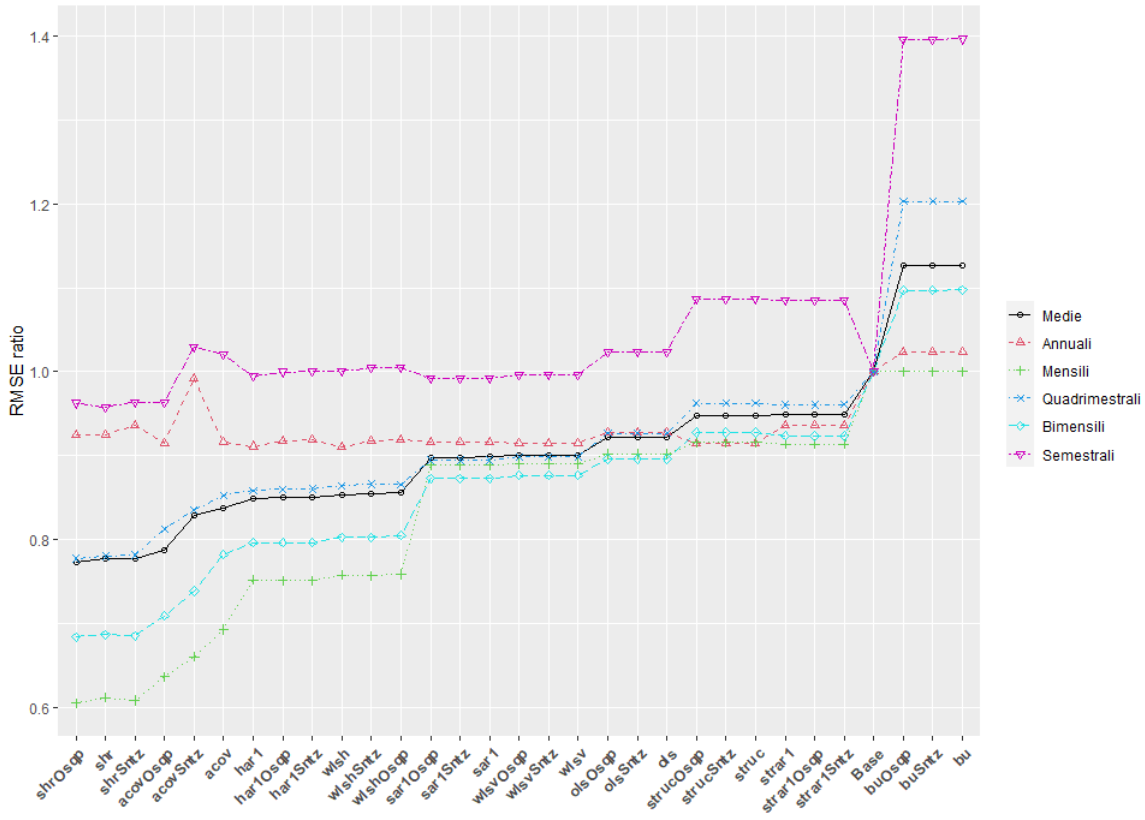


Figura 7.7: Rappresentazione grafica dell'RMSE ratio per le previsioni riconciliate temporalmente per le *upper time series* della gerarchia contemporanea

Capitolo 8

Conclusioni

In questo eleborato, sono stati esposti i principali metodi per la riconciliazione di previsioni di serie storiche gerarchiche raggruppate e di gerarchie temporali di serie storiche, evidenziando i punti di forza e le debolezze dei diversi metodi. Quindi ci si è concentrati sugli applicativi disponibili attualmente per implementare i precedenti metodi. L'attenzione si è soffermata in particolare sul pacchetto di R FoReco, per il quale sono stati sottolineati i motivi che lo fanno distinguere rispetto ad altre implementazioni dei metodi di riconciliazione di previsioni.

Sono state esposte, poi, le potenzialità del linguaggio python, i motivi per i quali risulterebbe conveniente la disponibilità in questo linguaggio delle tecniche esposte e gli accorgimenti scelti per sviluppare in python gli algoritmi.

Durante lo sviluppo di questo lavoro gli obiettivi che erano stati fissati sono stati raggiunti solo parzialmente: pur riuscendo a sviluppare con successo un pacchetto di funzioni che permette l'applicazione delle tecniche di riconciliazione di previsioni di serie storiche gerarchiche in Python, non si sono ottenuti i risultati sperati in quanto ad efficienza, visto le scarse performance che sono state riscontrate. In particolare, sono stati evidenziati i limiti che allo stato attuale possiede ancora Python rispetto ad R per il calcolo scientifico in alcuni specifici ambiti, come l'inversione di matrici sparse.

Infine, nel capitolo 7 è stato appurato come la riconciliazione trami-

te i metodi che sono stati implementati in Python permemette un incremento dell'accuratezza delle previsioni sia nel contesto delle gerarchie contemporanee che in quello delle gerarchie temporali.

Un naturale proseguimento di questo lavoro consisterebbe nel rendere disponibili in Python anche le tecniche per la riconciliazione di previsioni di gerarchie cross-temporali che FoReco offre in R, ma abbiamo ragione di credere che anche in questo caso le performance risulteranno comunque inferiori a quelle di R in quanto gli ordini di grandezza delle matrici risultano ancora maggiori rispetto al caso cross sezionale.

Link utile

La versione per Python del pacchetto FoReco sviluppata per questa tesi con la relativa documentazione può essere consultata e scaricata dal seguente link: `https://github.com/AlbertoMaronilli/FoReco`

Bibliografia

[1] AA. VV. (2015). *Parallel Programming with numpy and scipy*. SciPy Cookbook. `https://scipy-cookbook.readthedocs.io/items/ParallelProgramming.html`

[2] Athanasopoulos, G., Ahmed, R. A. e Hyndman, R. J. (2009). *Hierarchical forecasts for Australian domestic tourism*. International Journal of Forecasting 25, 1, pp. 146–166.

[3] Athanasopoulos, G., Hyndman, R. J., Kourentzes, N. e Petropoulos, F. (2017). *Forecasting with temporal hierarchies*. European Journal of Operational Research 262, 1, pp. 60–74.

[4] Bates, D. e Maechler, M. (2022). *Matrix: Sparse and Dense Matrix Classes and Methods*. `https://cran.r-project.org/package=Matrix`.

[5] Box, G. e Jenkins, G. (1970). *Time Series Analysis-Forecasting and Control*. San Francisco: Holden Day.

[6] Boyd, S., Parikh, N., Chu, E., Peleato, B e Eckstein J. (2011). *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Foundations and Trends® in Machine Learning: 3, 1, pp 1-122.

[7] Cheng, Y. Y., Chan, P. P. e Qiu, Z. W. (2012). *Random forest based ensemble system for short term load forecasting*. Proceedings - International Conference on Machine Learning and Cybernetics. Vol. 1, pp. 52–56.

[8] Cugliari, J., Van Erven, T. (2015). *gtop: Game-Theoretically OPtimal (GTOP) Reconciliation Method*. https://cran.r-project.org/web/packages/gtop/index.html

[9] Di Fonzo, T. e Girolimetto, D. (2020). *Riconciliazione contemporanea, temporale e cross-temporale di previsioni di serie storiche* [Tesi di laurea magistrale]. Università degli studi di padova

[10] Donadio L. (2021). *TimeHierarchy*. GitHub repository. https://github.com/lorenzodonadio/TimeHierarchy

[11] van Erven, T. e Cugliari, J. (2015). *Game-Theoretically Optimal reconciliation of contemporaneous hierarchical time series forecasts*. Modeling and Stochastic Learning for Forecasting in High Dimensions. Springer Science e Business Media, LLC, pp. 297–317.

[12] Facebook Inc. (2022). *Prophet: Automatic Forecasting Procedure*. GitHub repository. https://github.com/facebook/prophet

[13] Girolimetto, D. e Di Fonzo T. (2022). *FoReco: Point Forecast Reconciliation*. https://cran.r-project.org/web/packages/FoReco/index.html

[14] Gross, C. W. e Sohl, J. E. (1990). *Disaggregation methods to expedite product line forecasting*. Journal of Forecasting 9, 3, pp. 233–254.

[15] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. (2020) *Array programming with NumPy* Nature 585, pp. 357–362. https://doi.org/10.1038/s41586-020-2649-2

[16] Hastings L. (2015). *Python's Infamous GIL*. EuroPython 2015 https://ep2015.europython.eu/conference/talks/pythons-infamous-gil.html

[17] Hyndman, R. J. e Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. Monash University, Australia. https://otexts.com/fpp2/

[18] Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F. (2022). *forecast: Forecasting functions for time series and linear models*. `https://pkg.robjhyndman.com/forecast/`.

[19] Hyndman, R. J., Koehler, A. B., Snyder, R. D. e Grose, S. (2002). *A state space framework for automatic forecasting using exponential smoothing methods*. International Journal of Forecasting 18, 3, pp. 439–454

[20] Hyndman, R. J., Ahmed, R. A., Athanasopoulos, G. e Shang, H. L. (2011), *Optimal combination forecasts for hierarchical time series*. Computational Statistics and Data Analysis 55, 9, pp. 2579–2589.

[21] Hyndman, R. J., Lee, A., Wang, E. e Wickramasuriya, S. L. (2021). *hts: Hierarchical and Grouped Time Series*. `https://cran.r-project.org/package=hts`.

[22] Hyndman, R. J. e Kourentzes, N. (2018). *thief: Temporal HIErarchical Forecasting* `http://pkg.robjhyndman.com/thief/`, ultima consultazione: 17/07/2022

[23] Jeon, J., Panagiotelis, A. e Petropoulos, F. (2019). *Probabilistic forecast reconciliation with applications to wind power and electric load*. European Journal of Operational Research 279, 2, pp. 364–379.

[24] Ledoit, O. e Wolf, M. (2004). *Honey, I Shrunk the Sample Covariance Matrix*. J. Portfolio Management 30, pp. 110–119.

[25] Lütkepohl, H. (2005). *New introduction to multiple time series analysis*. New York: Springer.

[26] Mazzaferro, C., et al. (2021). *scikit-hts*. GitHub repository. `https://github.com/carlomazzaferro/scikit-hts`

[27] Nystrup, P., Lindström, E., Pinson, P. e Madsen, H. (2020). *Temporal hierarchies with autocorrelation for load forecasting*. European Journal of Operational Research 280, 3, pp. 876–888.

[28] Nelson B. (2021). *Scientific Computing with Python*. University of Chicago. https://caam37830.github.io/book/

[29] O'Hara-Wild, M., Hyndman, R. J. e Wang, E. (2021). *fable: Forecasting Models for Tidy Time Series*. https://cran.r-project.org/package=fable.

[30] Orcutt, G. H., Watts, H. W. e Edwards, J. B. (1968), *Data aggregation and information loss*. The American Economic Review 58, 4, pp. 773–787.

[31] Panagiotelis, A., Gamakumara, P., Athanasopoulos, G. e Hyndman, R. J. (2021). *Forecast Reconciliation: A geometric View with New Insightson Bias Correction*. International Journal of Forecasting in press.

[32] The pandas development team (2020). *pandas-dev/pandas: Pandas*. Zenodo https://doi.org/10.5281/zenodo.3509134

[33] Peters, T. (2004). *PEP 20 – The Zen of Python*. https://peps.python.org/pep-0020/

[34] Python Software Foundation (2022). *The Python Language Reference*. https://docs.python.org/3/reference/

[35] Rooney, C. (2021). textithtsprophet. GitHub repository. https://github.com/CollinRooney12/htsprophet

[36] Schäfer, J. e Strimmer, K. (2005). *A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics*. Statistical Applications in Genetics and Molecular Biology 4, 1, pp. 1–30.

[37] Spiliotis, E., Abolghasemi, M., Hyndman, R. J., Petropoulos, F. e Assimakopoulos, V. (2020). *Hierarchical forecast reconciliation with machine learning*. arXiv: 2006.02043

[38] Stellato, B., Banjac, G., Goulart, P., Bemporad, A. e Boyd, S. (2020). *OSQP: an operator splitting solver for quadratic programs*. Mathematical Programming Computation 12, 4, pp. 637–672

[39] Vanners, B. (2003). *The Making of Python: A Conversation with Guido van Rossum*. https://www.artima.com/articles/the-making-of-python

[40] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020).*SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, 17, pp. 261–272. https://doi.org/10.1038/s41592-019-0686-2

[41] Wickham H., Rstudio (2022). *tidyverse: Easily Install and Load the 'Tidyverse'*. https://cran.r-project.org/web/packages/tidyverse/index.html

[42] Wickramasuriya, S. L., Athanasopoulos, G. e Hyndman, R. J. (2019). *Optimal Forecast Reconciliation for Hierarchical and Grouped Time Series Through Trace Minimization*. Journal of the American Statistical Association 114, 526, pp. 804–819.

[43] Wickramasuriya, S. L., Turlach, B. A. e Hyndman, R. J. (2020). *Optimal non-negative forecast reconciliation*. Statistics and Computing 30, 5, pp. 1167–1182.

[44] Zhang, B et al. (2021). *pyhts*. GitHub repository. https://github.com/AngelPone/pyhts

Ringraziamenti

A conclusione di questo elaborato, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita professionale e personale.

Innanzitutto vorrei ringraziare il mio relatore Tommaso Di Fonzo, per i suoi importantissimi consigli, per la sua grande pazienza e soprattutto per le conoscenze che mi ha trasmesso durante tutto il percorso di stesura dell'elaborato e non solo. Ringrazio anche il dottorando Daniele Girolimetto per il supporto che mi ha fornito durante lo sviluppo della versione per Python del pacchetto FoReco.

Un sentito ringraziamento anche ai miei colleghi Ludovico Copetti, Jacopo Gallocchio, Francesco Gugole, Erica Marcolin, Giovanni Romanò e Monica Taschetti con cui ho condiviso le gioie e le fatiche in questi anni di laurea magistrale.

E un ringraziamento speciale alla mia famiglia che mi ha sempre sostenuto e incoraggiato soprattutto nei momenti più difficili.