Computational Statistics and Data Analysis 97 (2016) 16–32

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

journal homepage: www.elsevier.com/locate/csda

Fast computation of reconciled forecasts for hierarchical and grouped time series

Rob J. Hyndman[a,*], Alan J. Lee[b], Earo Wang[a]

[a] Department of Econometrics and Business Statistics, Monash University, VIC 3800, Australia
[b] Department of Statistics, University of Auckland, New Zealand

ARTICLE INFO

Article history:
Received 17 November 2014
Received in revised form 17 November 2015
Accepted 17 November 2015
Available online 30 November 2015

Keywords:
Combining forecasts
Grouped time series
Hierarchical time series
Reconciling forecasts
Weighted least squares

ABSTRACT

It is shown that the least squares approach to reconciling hierarchical time series forecasts can be extended to much more general collections of time series with aggregation constraints. The constraints arise due to the need for forecasts of collections of time series to add up in the same way as the observed time series. It is also shown that the computations involved can be handled efficiently by exploiting the structure of the associated design matrix, or by using sparse matrix routines. The proposed algorithms make forecast reconciliation feasible in business applications involving very large numbers of time series.

1. Introduction

Time series can often be naturally disaggregated in a hierarchical or grouped structure. For example, a manufacturing company can disaggregate total demand for their products by country of sale, retail outlet, product type, package size, and so on. As a result, there can be tens of thousands of individual time series to forecast at the most disaggregated level, plus additional series to forecast at higher levels of aggregation.

The most disaggregated series often have a high degree of volatility (and are therefore hard to forecast), while the most aggregated time series is usually smooth and less noisy (and is therefore easier to forecast). Consequently, forecasting only the most disaggregated series and summing the results tends to give poor results at the higher levels of aggregation. On the other hand, if all the series at all levels of aggregation are forecast independently, the forecasts will not add up consistently between levels of aggregation. Therefore, it is necessary to reconcile the forecasts to ensure that the forecasts of the disaggregated series add up to the forecasts of the aggregated series (Fliedner, 2001).

In some cases, the disaggregation can be expressed in a hierarchical structure. For example, time series of sales may be disaggregated geographically by country, state and region, where each level of disaggregation sits within one node at the higher level (a region is within a state which is within a country).

In other cases, the disaggregation may not be uniquely hierarchical. For example, time series of sales may involve a geographical grouping (by country) and a product grouping (by type of product sold). Each of these grouping variables leads to a unique hierarchy, but if we wish to use both grouping variables, the order in which the disaggregation occurs is not unique. We may disaggregate by geography first, then by product type, or we may disaggregate by product type, then by

* Corresponding author.
E-mail addresses: rob.hyndman@monash.edu (R.J. Hyndman), lee@stat.auckland.ac.nz (A.J. Lee), yiru.wang@monash.edu (E. Wang).

http://dx.doi.org/10.1016/j.csda.2015.11.007
0167-9473/

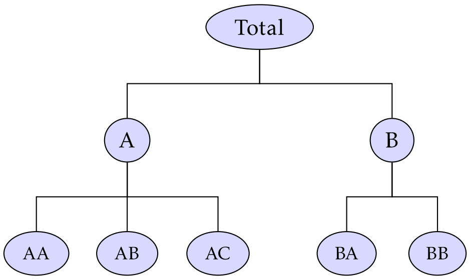

Fig. 1. A simple hierarchy of time series containing five bottom-level series.

geography. Both types of disaggregation may be useful, so choosing only one of these hierarchies is not appropriate. We call this a grouped time series. In general, there can be many different grouping variables, some of which may be hierarchical.

Hyndman et al. (2011, CSDA) proposed a method for optimally reconciling forecasts of all series in a hierarchy to ensure they add up. Our first contribution (Section 2) is to show that this result is easily extended to cover non-hierarchical groups of time series, and groups with a partial hierarchical structure.

The optimal reconciliation method involves fitting a linear regression model where the design matrix has one column for each of the series at the most disaggregated level. Consequently, for problems involving tens of thousands (or even millions) of series, the model is impossible to estimate using standard regression algorithms such as the QR decomposition.

The second contribution of this paper is to propose a solution to this problem, exploiting the unique structure of the linear model to efficiently estimate the coefficients, even when there are millions of time series at the most disaggregated level. Our solution involves two important cases: (1) where there can be any number of groups that are strictly hierarchical (Section 3); and (2) where there are two groups that may not be hierarchical (Section 4). All other cases can be handled using sparse matrix algorithms as discussed in Section 5.

For smaller hierarchies, where the computation is tractable without our new algorithms, good results have been obtained in forecasting tourism demand in Australia (Athanasopoulos et al., 2009) and inflation in Mexico (Capistrán et al., 2010).

Our algorithms make forecast reconciliation on very large collections of grouped and hierarchical time series feasible in practice. The algorithm has applications in any situation where large numbers of related time series need to be forecast, particularly in forecasting demand in product hierarchies, or geographical hierarchies. In Section 6 we demonstrate the different algorithms on a collection of quarterly time series giving the number of people holding different occupations in Australia.

2. Optimal reconciliation of hierarchical and grouped time series

2.1. Hierarchical time series

To introduce the notation and key ideas, we will use a very small and simple hierarchy, shown in Fig. 1, with five series at the most disaggregated level. The "Total" node is the most aggregate level of the data, with the tth observation denoted by y_t. It is disaggregated into series A and B, which are each further disaggregated into the five bottom-level series. The tth observation at node X is denoted by $y_{X,t}$.

For any time t, the observations of the bottom level series will aggregate to the observations of the series above. To represent this in matrix notation, we introduce the $n \times n_K$ "summing" matrix $\boldsymbol{S}$. For the hierarchy in Fig. 1 we can write

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

or in more compact notation $\boldsymbol{y}_t = \boldsymbol{S}\boldsymbol{b}_t$, where $\boldsymbol{y}_t$ is a vector of all the observations in the hierarchy at time t, $\boldsymbol{S}$ is the summing matrix as defined above, and $\boldsymbol{b}_t$ is a vector of all the observations in the bottom level of the hierarchy at time t.

We are interested in generating forecasts for each series in the hierarchy. Let $\hat{y}_{\mathrm{X},h}$ be an h-step-ahead forecast for the series at node X and let $\hat{y}_h$ be an h-step-ahead forecast generated for the "Total" series, computed using the data to time T. These forecasts are computed independently for each series, using only the history of the series and not taking account of any relationships with other series in the hierarchy. We refer to these as "base forecasts". The sum of the base forecasts at lower levels are unlikely to be equal to the base forecast of their parent nodes.

Table 1
Two-way table representation of the groups of time series shown in Fig. 2.

			Sum
	AX	BX	X
	AY	BY	Y
Sum	A	B	Total

We wish to adjust these base forecasts so that they are consistent with the structure of the hierarchy, with adjusted forecasts aggregating to be equal to the adjusted forecast of their parent node. We refer to the adjusted forecasts as "reconciled forecasts".

Hyndman et al. (2011) proposed the linear model

$$\hat{\boldsymbol{y}}_h = \boldsymbol{S}\boldsymbol{\beta}_h + \boldsymbol{\varepsilon}_h, \tag{1}$$

where $\hat{\boldsymbol{y}}_h$ is the vector of the h-step-ahead base forecasts for the whole hierarchy, $\boldsymbol{\beta}_h = \mathrm{E}(\boldsymbol{b}_{T+h} \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$ is the unknown conditional mean of the future values of the bottom level series, and $\boldsymbol{\varepsilon}_h$ represents the "aggregation error" (the difference between the base forecasts and their expected values if they were reconciled). We assume that $\boldsymbol{\varepsilon}_h$ has zero mean and covariance matrix $\boldsymbol{\Sigma}_h$. Then the optimally reconciled forecasts are given by the generalized least squares (GLS) solution

$$\tilde{\boldsymbol{y}}_h = \boldsymbol{S}\hat{\boldsymbol{\beta}}_h = \boldsymbol{S}(\boldsymbol{S}'\boldsymbol{\Sigma}_h^{\dagger}\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Sigma}_h^{\dagger}\hat{\boldsymbol{y}}_h, \tag{2}$$

where $\boldsymbol{\Sigma}_h^{\dagger}$ is a generalized inverse of $\boldsymbol{\Sigma}_h$. This approach combines the independent base forecasts and generates a set of reconciled forecasts that are as close as possible (in an L_2 sense) to the original forecasts but also aggregate consistently with the hierarchical structure.

Unfortunately, $\boldsymbol{\Sigma}_h$ is not known and very difficult (or impossible) to estimate for large hierarchies. Instead, we could use the weighted least squares (WLS) solution given by

$$\tilde{\boldsymbol{y}}_h = \boldsymbol{S}(\boldsymbol{S}'\boldsymbol{\Lambda}_h\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Lambda}_h\hat{\boldsymbol{y}}_h, \tag{3}$$

where $\boldsymbol{\Lambda}_h$ is a diagonal matrix with elements equal to the inverse of the variances of $\boldsymbol{\varepsilon}_h$. But even in this case, estimates of $\boldsymbol{\Lambda}_h$ are not readily available.

An alternative approach is to use ordinary least squares (OLS) instead of GLS or WLS. Then the set of reconciled forecasts is given by

$$\tilde{\boldsymbol{y}}_h = \boldsymbol{S}(\boldsymbol{S}'\boldsymbol{S})^{-1}\boldsymbol{S}'\hat{\boldsymbol{y}}_h. \tag{4}$$

Hyndman et al. (2011) noted that the GLS estimates (2) are identical to the OLS estimates (4) when the aggregation errors approximately satisfy the same aggregation structure as the original data, (i.e., $\boldsymbol{\varepsilon}_h \approx \boldsymbol{S}\boldsymbol{\varepsilon}_{b,h}$ where $\boldsymbol{\varepsilon}_{b,h}$ contains the aggregation errors in the bottom level). Even if the aggregation errors do not satisfy this assumption, the OLS solution will still be a consistent way of reconciling the base forecasts.

We may also regard the OLS and WLS solutions in approximation terms: the solution gives the adjusted base forecasts with the property that the whole set of adjusted reconciled forecasts $\tilde{\boldsymbol{y}}_h$ best approximates the unadjusted base forecasts in the least squares or weighted least squares sense. In this interpretation, the weights could be used to control how much the adjusted forecasts are allowed to differ from the unadjusted ones. Series with more accurate forecasts might well have a larger weight.

Consequently, in practice, we either use OLS, or we use WLS where the elements of Λ_h are set to the inverse of the variances of the base forecasts, $\mathrm{var}(\boldsymbol{y}_{T+1} - \hat{\boldsymbol{y}}_1)$. Note that we use the one-step forecast variances here, not the h-step forecast variances. This is because the one-step forecast variances are readily available as the residual variances for each of the base forecasting models. We are assuming that these are roughly proportional to the h-step forecast variances, which is true for almost all standard time series forecasting models (see, e.g., Hyndman et al., 2008).

However, even with these simpler solutions (3) or (4), the problem of inverting the potentially large matrices $\boldsymbol{S}'\boldsymbol{S}$ or $\boldsymbol{S}'\boldsymbol{\Lambda}_h\boldsymbol{S}$ remains.

A similar idea has been used for imposing aggregation constraints on time series produced by national statistical agencies (Quenneville and Fortier, 2012). A related problem is temporal reconciliation of time series and forecasts (e.g., to ensure that monthly values add up to annual values). Di Fonzo and Marini (2011) discuss how systems of time series subject to both temporal and contemporaneous constraints can be reconciled using a linear model approach equivalent to that described above. They propose a reconciliation algorithm that exploits the sparsity of the linear system, as we do in Section 5.

Grouped time series

It is not difficult to see that the same linear model (1) will be satisfied in situations with more general groups of time series. Consider the example shown in Fig. 2. These data can form two alternative hierarchies depending on which variable is first used to disaggregate the data. Another way to think of these data is in a two way table as shown in Table 1.

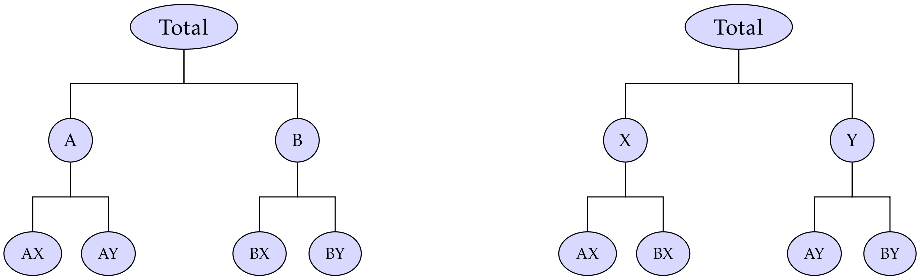

Fig. 2. A simple example of grouped time series with four bottom-level series. The series can be split into groups A and B first, then into groups X and Y (shown on the left), or they can be split first into groups X and Y, and then into groups A and B (shown on the right).

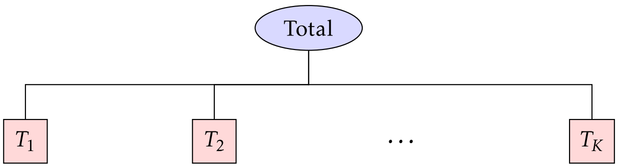

Fig. 3. A recursive hierarchical tree diagram.

Then if forecasts of all bottom level series, and of each grouping aggregate, are obtained, then the same simple linear model (1) can be used to represent them where

$$\boldsymbol{y}_t = \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{X,t} \\ y_{Y,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{pmatrix}, \qquad \boldsymbol{S} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{b}_t = \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{pmatrix}.$$

More complicated grouping structures can be represented similarly. Each grouping aggregate of interest is included in vector $\boldsymbol{y}_t$, and the corresponding row in $\boldsymbol{S}$ defines how the bottom-level series are grouped for that aggregation. The grouping structures may be hierarchical, non-hierarchical, or a mix of each. In Section 6 we describe examples of all three types. The same GLS, WLS, or OLS solutions can be used to reconcile the base forecasts provided the aggregation constraints can be expressed in the form (1).

We now proceed to consider fast algorithms for computing the WLS solution given by (3) when the time series are strictly hierarchical (Section 3), or there are two non-hierarchical grouping variables (Section 4), and all other cases (Section 5). In each case, the OLS solution (4) can be considered as a special case of the WLS results.

3. Fast computation to reconcile hierarchical time series

Every hierarchical time series can be represented recursively as a tree of trees. Consider the hierarchy depicted in Fig. 3, consisting of a root node (labelled Total) and several appended sub-trees, denoted by $T_1, T_2, \ldots, T_K$.

Each of the sub-trees are themselves hierarchies of time series. Suppose that the trees $T_1, T_2, \ldots, T_K$ contain $n_1, n_2, \ldots, n_K$ terminal nodes respectively.

Let $\boldsymbol{y}_{k,t}$ be the vector of time series associated with the tree T_k, and let $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_K$ be the summing matrices corresponding to the subtrees $T_1, T_2, \ldots, T_K$. Then the summing matrix corresponding to the whole tree T is

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{1}'_{n_1} & \boldsymbol{1}'_{n_2} & \cdots & \boldsymbol{1}'_{n_K} \\ \boldsymbol{S}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{S}_K \end{bmatrix} \tag{5}$$

where $\boldsymbol{1}_n$ is an n-vector of ones.

We want to produce the adjusted forecasts given by (3). Direct computation by standard least squares methods may be difficult if the matrix $\boldsymbol{S}$ is too large. Instead, we consider a recursive method for computing the adjusted forecasts which

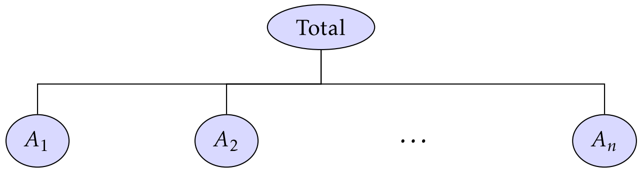

Fig. 4. A simple tree with two levels.

can handle quite large tree structures, particularly if the number of forecasts being aggregated at the lowest level is large compared to those being aggregated at higher levels. The method can also handle different forecast weights, and allows efficient computation of the matrix $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}$ by exploiting the structure of $\boldsymbol{S}$.

Write

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_0 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\Lambda}_K \end{bmatrix},$$

where λ_0 is the weight associated with the root node, and $\boldsymbol{\Lambda}_1, \ldots, \boldsymbol{\Lambda}_K$ are the diagonal weight matrices associated with the nodes in $T_1, \ldots, T_K$. Then

$$\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S} = \lambda_0 \boldsymbol{J}_n + \begin{bmatrix} \boldsymbol{S}_1'\boldsymbol{\Lambda}_1\boldsymbol{S}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{S}_2'\boldsymbol{\Lambda}_2\boldsymbol{S}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{S}_K'\boldsymbol{\Lambda}_K\boldsymbol{S}_K \end{bmatrix}, \tag{6}$$

where $\boldsymbol{J}_n$ is an $n \times n$ matrix of ones, and $n = \sum_k n_k$. Using the Sherman–Morrison–Woodbury updating formula (see, e.g., Seber and Lee, 2003, p.467) we can express the inverse of $\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S}$ as

$$(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1} = \begin{bmatrix} (\boldsymbol{S}_1'\boldsymbol{\Lambda}_1\boldsymbol{S}_1)^{-1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & (\boldsymbol{S}_2'\boldsymbol{\Lambda}_2\boldsymbol{S}_2)^{-1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & (\boldsymbol{S}_K'\boldsymbol{\Lambda}_K\boldsymbol{S}_K)^{-1} \end{bmatrix} - c\boldsymbol{S}_0, \tag{7}$$

where the $n \times n$ matrix $\boldsymbol{S}_0$ can be partitioned into K^2 blocks, and the j, k block (of dimension $n_j \times n_k$) is $(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1}\boldsymbol{J}_{n_j,n_k}$ $(\boldsymbol{S}_k'\boldsymbol{\Lambda}_k\boldsymbol{S}_k)^{-1}$, where $\boldsymbol{J}_{n_j,n_k}$ is a $n_j \times n_k$ matrix of ones. The constant c is given by

$$c^{-1} = \lambda_0^{-1} + \sum_j \mathbf{1}_{n_j}'(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1}\mathbf{1}_{n_j}.$$

3.1. Recursive calculation of the inverse

We can establish the form of $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}$ more precisely by an inductive argument: consider a simple tree consisting of a root node and n terminal nodes, as depicted in Fig. 4. In this case, the matrix $\boldsymbol{S}$ is of the form

$$\boldsymbol{S} = \begin{bmatrix} \mathbf{1}_n' \\ \boldsymbol{I}_n \end{bmatrix}, \tag{8}$$

where $\boldsymbol{I}_n$ is an $n \times n$ identity matrix. Then

$$\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S} = \operatorname{diag}(\boldsymbol{\lambda}) + \lambda_0 \boldsymbol{J}_n, \tag{9}$$

where the elements of $\boldsymbol{\lambda}$ are the weights of the terminal nodes, and

$$(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1} = \operatorname{diag}(\boldsymbol{d}) - c\boldsymbol{d}\boldsymbol{d}', \tag{10}$$

where $\boldsymbol{d}$ is the vector whose elements are the reciprocals of those of $\boldsymbol{\lambda}$, $d_0 = \lambda_0^{-1}$ and $c^{-1} = d_0 + \boldsymbol{d}'\mathbf{1}_n$.

Now consider the situation in Fig. 5, with a tree having three levels, where the K subtrees $T_1, T_2, \ldots, T_K$ of Fig. 3 have the simple form depicted in Fig. 4. Let $\boldsymbol{d}_j$ be the vector of reciprocal weights for the terminal nodes of the subtrees T_j, d_{j0} the reciprocal weight of the root node of T_j, d_0 the reciprocal weight of the root node of the whole tree and $c_j = d_{j0} + \boldsymbol{d}_j'\mathbf{1}_{n_j}$. Then

$$(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1}\boldsymbol{J}_{n_j,n_k}(\boldsymbol{S}_k'\boldsymbol{\Lambda}_k\boldsymbol{S}_k)^{-1} = (1 - c_j\boldsymbol{d}_j'\mathbf{1}_{n_j})(1 - c_k\boldsymbol{d}_k'\mathbf{1}_{n_k})\boldsymbol{d}_j\boldsymbol{d}_k',$$

and

$$\sum_j \mathbf{1}_{n_j}'(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1}\mathbf{1}_{n_j} = \sum_j \boldsymbol{d}_j'\mathbf{1}_{n_j} - c_j(\boldsymbol{d}_j'\mathbf{1}_{n_j})^2.$$

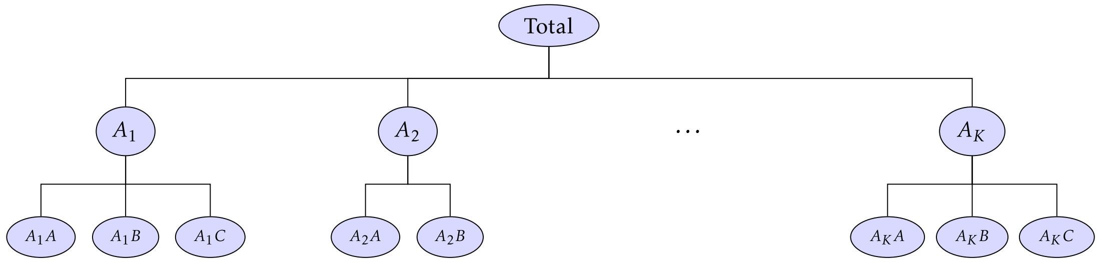

Fig. 5. A tree with three levels.

Thus, by (7), the (j, j) block of $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}$ is

$$(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1} - c(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1}\boldsymbol{J}_{n_j,n_j}(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1} = \mathrm{diag}(\boldsymbol{d}_j) - \left\{c_j + c(1 - c_j\boldsymbol{d}_j'\boldsymbol{1}_{n_j})^2\right\}\boldsymbol{d}_j\boldsymbol{d}_j', \tag{11}$$

where $c^{-1} = d_0 + \sum_j \boldsymbol{d}_k'\boldsymbol{1}_{n_j} - c_j(\boldsymbol{d}_j'\boldsymbol{1}_{n_j})^2$. For $j \neq k$, the (j, k) block is $-c(1 - c_j\boldsymbol{d}_j'\boldsymbol{1}_{n_j})(1 - c_k\boldsymbol{d}_k'\boldsymbol{1}_{n_k})\boldsymbol{d}_j\boldsymbol{d}_k'$. We can write this more compactly by defining

$$\tilde{\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{d}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{d}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{d}_K \end{bmatrix}. \tag{12}$$

Then we can write $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1} = \boldsymbol{D} - \tilde{\boldsymbol{D}}\boldsymbol{C}\tilde{\boldsymbol{D}}'$ where the symmetric matrix $\boldsymbol{C}$ has elements

$$c_{jk} = \begin{cases} c_j + c(1 - c_j\boldsymbol{d}_j'\boldsymbol{1}_{n_j})(1 - c_k\boldsymbol{d}_k'\boldsymbol{1}_{n_k}) & j = k, \\ c(1 - c_j\boldsymbol{d}_j'\boldsymbol{1}_{n_j})(1 - c_k\boldsymbol{d}_k'\boldsymbol{1}_{n_k}) & j \neq k; \end{cases}$$

and

$$\boldsymbol{D} = \begin{bmatrix} \mathrm{diag}(\boldsymbol{d}_1) & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \mathrm{diag}(\boldsymbol{d}_2) & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \mathrm{diag}(\boldsymbol{d}_K) \end{bmatrix}. \tag{13}$$

This pattern for the inverse also applies at higher levels in the hierarchy. Assume that for $j = 1, \ldots, K$, the matrix $(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1}$ has the form

$$(\boldsymbol{S}_j'\boldsymbol{\Lambda}_j\boldsymbol{S}_j)^{-1} = \boldsymbol{D}_j - \tilde{\boldsymbol{D}}_j\boldsymbol{C}_j\tilde{\boldsymbol{D}}_j', \tag{14}$$

where $\tilde{\boldsymbol{D}}_j$ and $\boldsymbol{D}_j$ have the form (12) and (13) for vectors $\boldsymbol{d}_{j1}, \ldots, \boldsymbol{d}_{jm_j}$. Substituting these in Eq. (7) we get

$$(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1} = \boldsymbol{D} - \tilde{\boldsymbol{D}}\boldsymbol{C}\tilde{\boldsymbol{D}}', \tag{15}$$

where $\boldsymbol{C}$ is symmetric with its j, k block equal to

$$\boldsymbol{C}_{jk} = \begin{cases} \boldsymbol{C}_j + c(\boldsymbol{1}_{m_j} - \boldsymbol{C}_j\tilde{\boldsymbol{D}}_j'\boldsymbol{1}_{n_j})(\boldsymbol{1}_{m_k} - \boldsymbol{C}_k\tilde{\boldsymbol{D}}_k'\boldsymbol{1}_{n_k})', & j = k, \\ c(\boldsymbol{1}_{m_j} - \boldsymbol{C}_j\tilde{\boldsymbol{D}}_j'\boldsymbol{1}_{n_j})(\boldsymbol{1}_{m_k} - \boldsymbol{C}_k\tilde{\boldsymbol{D}}_k'\boldsymbol{1}_{n_k})', & j \neq k, \end{cases} \tag{16}$$

$$c^{-1} = d_0 + \mathrm{trace}\,\boldsymbol{D} - \sum_j \boldsymbol{1}_{n_j}'\tilde{\boldsymbol{D}}_j\boldsymbol{C}_j\tilde{\boldsymbol{D}}_j'\boldsymbol{1}_{n_j}, \tag{17}$$

and

$$\tilde{\boldsymbol{D}} = \begin{bmatrix} \tilde{\boldsymbol{D}}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{D}}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \tilde{\boldsymbol{D}}_K \end{bmatrix}, \qquad \boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{D}_K \end{bmatrix}. \tag{18}$$

Using these recursions, we can calculate the matrix $\boldsymbol{C}$ for each subtree in the tree. If the tree has L levels, labelled $1, 2, \ldots, L$, we start with the set of "terminal" subtrees whose root nodes are the nodes at level $L-1$. We calculate the $\boldsymbol{C}$-matrix (which consists of a single element) for each subtree, using Eq. (10), so that the matrix corresponding to a node at level $L-1$ has the single element $d_0 + \boldsymbol{d}'\boldsymbol{1}_{n_j}$. Using these $\boldsymbol{C}$-matrices, we can use (16) and (17) to calculate the $\boldsymbol{C}$-matrices for all subtrees whose root nodes are the nodes at the next level $L-2$. Proceeding in this way, we eventually arrive at the single $\boldsymbol{C}$-matrix which represents the inverse for the whole tree. The dimension of the final $\boldsymbol{C}$ is equal to the number of nodes at the second-to-bottom level of the tree. Thus, if the number of terminal nodes at level L is considerably greater than the number of nodes at level $L-1$, the dimensions of this matrix will be considerably less than that of $\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S}$.

Note that if $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N$ are the vectors of inverse weights of the N terminal trees, the elements of these vectors are the diagonal elements of the matrix $\boldsymbol{D}$ in (13), and the matrix $\tilde{\boldsymbol{D}}$ is

$$\tilde{\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{d}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{d}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{d}_N \end{bmatrix}.$$

3.2. Calculation of the adjusted forecasts

The elements of the vector $\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}$, where $\hat{\boldsymbol{y}}$ is the vector of unadjusted forecasts, can also be calculated recursively. If $\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_K$ are the forecasts corresponding to the subtrees $T_1, T_2, \ldots, T_K$ and $\hat{y}_0$ is the forecast of the aggregate, then

$$\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{1}_{n_1} & \boldsymbol{S}_1' & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{1}_{n_2} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{1}_{n_k} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{S}_K' \end{bmatrix} \begin{bmatrix} \lambda_0\hat{y}_0 \\ \boldsymbol{\Lambda}_1\hat{\boldsymbol{y}}_1 \\ \boldsymbol{\Lambda}_2\hat{\boldsymbol{y}}_2 \\ \vdots \\ \boldsymbol{\Lambda}_K\hat{\boldsymbol{y}}_K \end{bmatrix} = \hat{y}_0\lambda_0\boldsymbol{1}_n + \begin{bmatrix} \boldsymbol{S}_1'\boldsymbol{\Lambda}_1\hat{\boldsymbol{y}}_1 \\ \boldsymbol{S}_2'\boldsymbol{\Lambda}_2\hat{\boldsymbol{y}}_2 \\ \vdots \\ \boldsymbol{S}_K'\boldsymbol{\Lambda}_K\hat{\boldsymbol{y}}_K \end{bmatrix}. \tag{19}$$

The recursion is started off at level $L-1$ using the equation

$$\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}} = \begin{bmatrix} \lambda_0\hat{y}_0 + \lambda_1\hat{\boldsymbol{y}}_1 \\ \vdots \\ \lambda_0\hat{y}_0 + \lambda_n\hat{\boldsymbol{y}}_n \end{bmatrix}, \tag{20}$$

where $\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_n$ are the forecasts corresponding to the terminal nodes and $\hat{y}_0$ is the forecast of the aggregate.

Partition $\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}$ into subvectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_N$ corresponding to the N terminal subtrees and let $t_j = \boldsymbol{d}_j'\boldsymbol{s}_j$. Put $\boldsymbol{t} = (t_1, \ldots, t_N)'$. Then the vector of "bottom level" adjusted forecasts is

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{d}_1 \cdot \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{d}_N \cdot \boldsymbol{s}_N \end{bmatrix} - \begin{bmatrix} (\boldsymbol{C}\boldsymbol{t})_1\boldsymbol{d}_1 \\ \vdots \\ (\boldsymbol{C}\boldsymbol{t})_N\boldsymbol{d}_N \end{bmatrix}, \tag{21}$$

where $\cdot$ denotes an elementwise product.

Finally, the whole vector of adjusted forecasts can be calculated from the recursion

$$\boldsymbol{S}\hat{\boldsymbol{\beta}} = \begin{bmatrix} \boldsymbol{1}_{n_1}' & \boldsymbol{1}_{n_2}' & \cdots & \boldsymbol{1}_{n_{K-1}}' & \boldsymbol{1}_{n_K}' \\ \boldsymbol{S}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{S}_K \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}}_1 \\ \hat{\boldsymbol{\beta}}_2 \\ \vdots \\ \hat{\boldsymbol{\beta}}_K \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{\beta}}_1'\boldsymbol{1}_{n_1} + \cdots + \hat{\boldsymbol{\beta}}_K'\boldsymbol{1}_{n_K} \\ \boldsymbol{S}_1\hat{\boldsymbol{\beta}}_1 \\ \vdots \\ \boldsymbol{S}_K\hat{\boldsymbol{\beta}}_K \end{bmatrix}, \tag{22}$$

where $\hat{\boldsymbol{\beta}}_1, \ldots, \hat{\boldsymbol{\beta}}_K$ are the vectors of "bottom level" adjusted forecasts for the subtrees $T_1, \ldots, T_K$.

When the weights are all unity (as for OLS), the formulas above simplify considerably. In this case at each stage the matrix $(\boldsymbol{S}_j'\boldsymbol{S}_j)^{-1}$ is of the form $(\boldsymbol{S}_j'\boldsymbol{S}_j)^{-1} = \boldsymbol{I}_{n_j} - \boldsymbol{R}$ where the matrix $\boldsymbol{R}$ consists of blocks of equal elements, whose common value is given by the corresponding element of the $\boldsymbol{C}$-matrix.

3.3. The algorithm

We may summarize the algorithm described above as follows:

Step 1: Calculate the (1×1) $\boldsymbol{C}$-matrices corresponding to level $L-1$ of the tree, using Eq. (7), and for each forecast horizon h, the list of vectors $\boldsymbol{S}_k'\boldsymbol{\Lambda}_k\hat{\boldsymbol{y}}_{kh}$, using Eq. (20).

Step 2: For each level from $L-2$ to level 1, and each node in that level, calculate the updated $\boldsymbol{C}$-matrices and the updated vectors $\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}_h$, using Eqs. (16), (17) and (19).

Step 3: Using the final $\boldsymbol{C}$-matrix and the final vector $\boldsymbol{s} = \boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}_h$ corresponding to the whole tree, calculate the adjusted forecasts using Eqs. (21) and (22).

This approach has been implemented in the hts package for R (Hyndman et al., 2015). For a 6-level hierarchy involving a total of 100,000 series at the bottom level, and 101,125 series in total, the calculations took 73 s on a PC running 64-bit Ubuntu with an 8-core Xeon processor at 3.70 GHz and 8 Gb RAM. No parallel processing was used and all the code was in native R.

4. A fast algorithm for grouped time series

In this section, we consider the case where the time series have a tabular rather than a hierarchical structure. We have a set of time series $\{y_{i,j,t}\}$, arranged in an $m \times n$ array, as in Table 1:

$$\begin{bmatrix} y_{1,1,t} & \cdots & y_{1,n,t} \\ \vdots & \dots & \vdots \\ y_{m,1,t} & \cdots & y_{m,n,t} \end{bmatrix}. \tag{23}$$

For example, we might have separate time series for each gender and for each region of a country. As well as the series above, we will be interested in the aggregated series $y_{i,+,t}$, $y_{+,j,t}$ and $y_{+,+,t}$ of the row, column and grand totals.

Suppose we have h-step-ahead forecasts $\hat{y}_{ij}$ of the series $y_{i,j,t}$, and also separate forecasts of the series $y_{i,+,t}$, $y_{+,j,t}$, and $y_{+,+,t}$, which we denote by $\hat{y}_{i0}$, $\hat{y}_{0j}$ and $\hat{y}_{00}$ respectively.

Let $\boldsymbol{S}$ be the $(m+1)(n+1) \times mn$ summing matrix for this situation, satisfying

$$\boldsymbol{S} \begin{bmatrix} y_{1,1,t} \\ y_{1,2,t} \\ \vdots \\ y_{m,n,t} \end{bmatrix} = \begin{bmatrix} y_{+,+,t} \\ y_{+,1,t} \\ \vdots \\ y_{+,n,t} \\ y_{1,+,t} \\ \vdots \\ y_{m,+,t} \\ y_{1,1,t} \\ y_{1,2,t} \\ \vdots \\ y_{m,n,t} \end{bmatrix}. \tag{24}$$

(Here and subsequently, when stringing the elements of a matrix out into a vector, we do this in the order first row, second row and so on.) Let $\hat{\boldsymbol{y}}$ be the vector of $(m+1)(n+1)$ forecasts, arranged in the same order as the right side of (24). Ignoring the weights $\boldsymbol{\Lambda}$ for now, the adjusted forecasts of the unaggregated series $y_{ij,t}$ in (23) are given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{S}'\boldsymbol{S})^{-1}\boldsymbol{S}'\hat{\boldsymbol{y}}, \tag{25}$$

and the aggregated adjusted forecasts are obtained by aggregating up the adjusted unaggregated forecasts, or, in matrix terms by $\boldsymbol{S}\hat{\boldsymbol{\beta}}$.

4.1. Solving the normal equations

Using the ordering in (24), the matrix $\boldsymbol{S}$ can be written as

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{1}_m' \otimes \boldsymbol{1}_n' \\ \boldsymbol{1}_m' \otimes \boldsymbol{I}_n \\ \boldsymbol{I}_m \otimes \boldsymbol{1}_n' \\ \boldsymbol{I}_m \otimes \boldsymbol{I}_n \end{bmatrix},$$

where $\boldsymbol{I}_n$ is an $n \times n$ identity matrix and $\boldsymbol{1}_n$ an n-vector of ones. Using the properties of Kronecker products, we obtain

$$\begin{aligned} \boldsymbol{S}'\boldsymbol{S} &= [\boldsymbol{1}_m \otimes \boldsymbol{1}_n \mid \boldsymbol{1}_m \otimes \boldsymbol{I}_n \mid \boldsymbol{I}_m \otimes \boldsymbol{1}_n \mid \boldsymbol{I}_m \otimes \boldsymbol{I}_n] \times \begin{bmatrix} \boldsymbol{1}_m' \otimes \boldsymbol{1}_n' \\ \boldsymbol{1}_m' \otimes \boldsymbol{I}_n \\ \boldsymbol{I}_m \otimes \boldsymbol{1}_n' \\ \boldsymbol{I}_m \otimes \boldsymbol{I}_n \end{bmatrix} \\ &= (\boldsymbol{1}_m \otimes \boldsymbol{1}_n)(\boldsymbol{1}_m' \otimes \boldsymbol{1}_n') + (\boldsymbol{1}_m \otimes \boldsymbol{I}_n)(\boldsymbol{1}_m' \otimes \boldsymbol{I}_n) + (\boldsymbol{I}_m \otimes \boldsymbol{1}_n)(\boldsymbol{I}_m \otimes \boldsymbol{1}_n') + (\boldsymbol{I}_m \otimes \boldsymbol{I}_n)(\boldsymbol{I}_m \otimes \boldsymbol{I}_n) \\ &= \boldsymbol{J}_m \otimes \boldsymbol{J}_n + \boldsymbol{J}_m \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{J}_n + \boldsymbol{I}_m \otimes \boldsymbol{I}_n \\ &= (\boldsymbol{I}_m + \boldsymbol{J}_m) \otimes (\boldsymbol{I}_n + \boldsymbol{J}_n), \end{aligned}$$

and so

$$\begin{aligned}(\boldsymbol{S}'\boldsymbol{S})^{-1} &= [(\boldsymbol{I}_m + \boldsymbol{J}_m) \otimes (\boldsymbol{I}_n + \boldsymbol{J}_n)]^{-1} \\ &= (\boldsymbol{I}_m + \boldsymbol{J}_m)^{-1} \otimes (\boldsymbol{I}_n + \boldsymbol{J}_n)^{-1} \\ &= (\boldsymbol{I}_m - \boldsymbol{J}_m/(m+1)) \otimes (\boldsymbol{I}_n - \boldsymbol{J}_n/(n+1)).\end{aligned}$$

Finally, the solution of the normal equations is

$$\begin{aligned}(\boldsymbol{S}'\boldsymbol{S})^{-1}\boldsymbol{S}'\hat{\boldsymbol{y}} = \frac{1}{(m+1)(n+1)} \Big\{ & \boldsymbol{J}_{mn}\hat{y}_{00} + \big[(n+1)(\boldsymbol{1}_m \otimes \boldsymbol{I}_n) - \boldsymbol{J}_{mn,n}\big]\hat{\boldsymbol{y}}_C + \big[(m+1)(\boldsymbol{I}_m \otimes \boldsymbol{1}_n) - \boldsymbol{J}_{mn,m}\big]\hat{\boldsymbol{y}}_R \\ & + \big[(m+1)(n+1)\boldsymbol{I}_m - (m+1)(\boldsymbol{I}_m \otimes \boldsymbol{J}_n) - (n+1)(\boldsymbol{J}_m \otimes \boldsymbol{I}_n) - \boldsymbol{J}_{mn}\big]\hat{\boldsymbol{y}}_B \Big\}\end{aligned}$$

where $\hat{y}_{00}$ is the forecast of the grand total, and $\hat{\boldsymbol{y}}_C$, $\hat{\boldsymbol{y}}_R$ and $\hat{\boldsymbol{y}}_B$ are the vectors of column, row and unaggregated forecasts respectively, and $\boldsymbol{J}_{mn,n}$ is a $mn \times n$ matrix of ones. Writing this vector as an $m \times n$ matrix, the adjusted forecast for row i, column j is

$$\begin{aligned}&\big\{\hat{y}_{00} - \hat{y}_{0+} - \hat{y}_{+0} + \hat{y}_{++} + (m+1)\hat{y}_{i0} + (n+1)\hat{y}_{0j} + (m+1)(n+1)\hat{y}_{ij} - (m+1)\hat{y}_{i+} \\ &\quad - -(n+1)\hat{y}_{+j}\big\} \big/ (m+1)(n+1),\end{aligned}$$

where $+$ denotes summation of the unadjusted forecasts over $1, 2, \ldots, m$ or $1, 2, \ldots, n$.

4.2. Incorporating weights

Suppose now for each forecast we have a weight, and we calculate the adjusted unaggregated forecasts by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}, \tag{26}$$

instead of (25), where $\boldsymbol{\Lambda}$ is the diagonal matrix of weights. Direct computation of (26) may be difficult if m and/or n are large, since it involves dealing with a matrix of dimension $mn \times mn$. However, we can exploit the special structure of $\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S}$ to develop a more efficient computational scheme that does not involve forming such large matrices.

Let λ_{ij}, $i = 0, 1, \ldots, m$; $j = 0, 1, \ldots, n$ denote the weights corresponding to the forecast $\hat{y}_{ij}$, so that for example λ_{00} is the weight associated with the grand total forecast $\hat{y}_{00}$.

Also, denote by $\boldsymbol{\Lambda}_R$, $\boldsymbol{\Lambda}_C$ and $\boldsymbol{\Lambda}_U$ the diagonal matrices whose diagonal elements are the row weights $\lambda_{10}, \ldots, \lambda_{m0}$, the column weights $\lambda_{01}, \ldots, \lambda_{0n}$ and the "unaggregated" weights $\lambda_{11}, \lambda_{12}, \ldots, \lambda_{mn}$.

We can write $\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S}$ as

$$\begin{aligned}\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S} &= \lambda_{00}(\boldsymbol{1}_m \otimes \boldsymbol{1}_n)(\boldsymbol{1}'_m \otimes \boldsymbol{1}'_n) + (\boldsymbol{I}_m \otimes \boldsymbol{1}_n)\boldsymbol{\Lambda}_R(\boldsymbol{I}_m \otimes \boldsymbol{1}'_n) + (\boldsymbol{1}_m \otimes \boldsymbol{I}_n)\boldsymbol{\Lambda}_C(\boldsymbol{1}'_m \otimes \boldsymbol{I}_n) + \boldsymbol{\Lambda}_U \\ &= \lambda_{00}\boldsymbol{J}_{mn} + (\boldsymbol{\Lambda}_R \otimes \boldsymbol{J}_n) + (\boldsymbol{J}_m \otimes \boldsymbol{\Lambda}_C) + \boldsymbol{\Lambda}_U.\end{aligned}$$

The first three matrices in the above sum have ranks 1, m and n and the last has a simple, easily computed inverse as it is diagonal. This suggests we might be able to compute $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}$ efficiently by successive applications of the Sherman–Morrison–Woodbury (SMW) updating formula (see e.g. Seber and Lee, 2003, p. 467), and this turns out to be the case. We now turn to the details.

First, note that we can write

$$(\boldsymbol{J}_m \otimes \boldsymbol{\Lambda}_C) = (\boldsymbol{1}_m \otimes \boldsymbol{\Lambda}_C^{1/2})(\boldsymbol{1}'_m \otimes \boldsymbol{\Lambda}_C^{1/2}) = \boldsymbol{B}'_C\boldsymbol{B}_C,$$

say. Thus if $\boldsymbol{A}_1 = ((\boldsymbol{J}_m \otimes \boldsymbol{\Lambda}_C) + \boldsymbol{\Lambda}_U)^{-1}$, by the SMW formula we have

$$\boldsymbol{A}_1 = \boldsymbol{\Lambda}_U^{-1} - \boldsymbol{\Lambda}_U^{-1}\boldsymbol{B}'_C(\boldsymbol{I}_n + \boldsymbol{B}_C\boldsymbol{\Lambda}_U^{-1}\boldsymbol{B}'_C)^{-1}\boldsymbol{B}_C\boldsymbol{\Lambda}_U^{-1}.$$

Now write

$$\boldsymbol{\Lambda}_U = \begin{bmatrix} \boldsymbol{\Lambda}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{0} \\ & & & \\ \boldsymbol{0} & \boldsymbol{0} & \vdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{\Lambda}_m \end{bmatrix},$$

so that the diagonal elements of $\boldsymbol{\Lambda}_i$ are λ_{ik}, $k = 1, 2, \ldots, n$. Then $\boldsymbol{B}_C\boldsymbol{\Lambda}_U^{-1}\boldsymbol{B}'_C = \boldsymbol{\Lambda}_C^{1/2}\sum_i \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\Lambda}_C^{1/2}$ which is diagonal, so that $(\boldsymbol{I}_n + \boldsymbol{B}_C\boldsymbol{\Lambda}_U^{-1}\boldsymbol{B}'_C)^{-1}$ is also a diagonal matrix with jth diagonal element $1/(1 + \lambda_{0j}\sum_i \lambda_{ij}^{-1})$. Finally,

$$\boldsymbol{B}'_C(\boldsymbol{I}_n + \boldsymbol{B}_C\boldsymbol{\Lambda}_U^{-1}\boldsymbol{B}'_C)^{-1}\boldsymbol{B}_C = \boldsymbol{J}_m \otimes \boldsymbol{D},$$

where $\boldsymbol{D}$ is a diagonal matrix with elements $d_j = \lambda_{0j}/(1 + \lambda_{0j}\sum_i \lambda_{ij}^{-1})$. Thus, we have

$$\boldsymbol{A}_1 = \boldsymbol{\Lambda}_U^{-1} - \boldsymbol{\Lambda}_U^{-1}(\boldsymbol{J}_m \otimes \boldsymbol{D})\boldsymbol{\Lambda}_U^{-1}. \tag{27}$$

Now consider

$$\boldsymbol{A}_2 = (\boldsymbol{A}_1^{-1} + (\boldsymbol{\Lambda}_R \otimes \boldsymbol{J}_n))^{-1} = (\boldsymbol{A}_1^{-1} + \boldsymbol{B}_R'\boldsymbol{B}_R)^{-1},$$

where $\boldsymbol{B}_R = (\boldsymbol{\Lambda}_R^{1/2} \otimes \boldsymbol{1}_n')$. Again by SMW, we have

$$\boldsymbol{A}_2 = \boldsymbol{A}_1 - \boldsymbol{A}_1\boldsymbol{B}_R'(\boldsymbol{I}_m + \boldsymbol{B}_R\boldsymbol{A}_1\boldsymbol{B}_R')^{-1}\boldsymbol{B}_R\boldsymbol{A}_1. \tag{28}$$

The $m \times m$ matrix $\boldsymbol{M} = \boldsymbol{I}_m + \boldsymbol{B}_R\boldsymbol{A}_1\boldsymbol{B}_R'$ has i, j element

$$\begin{cases} 1 + \lambda_{i0}\sum_k \lambda_{ik}^{-1} - \lambda_{i0}\sum_k \lambda_{ik}^{-2}d_k, & i = j, \\ -\lambda_{i0}^{1/2}\lambda_{j0}^{1/2}\sum_k \lambda_{ik}^{-1}\lambda_{jk}^{-1}d_k, & i \neq j. \end{cases} \tag{29}$$

We can also get an explicit formula for the elements of the matrix $\boldsymbol{E} = \boldsymbol{A}_1\boldsymbol{B}_R'$. We can write

$$\boldsymbol{E} = \begin{bmatrix} e_{11} & \cdots & e_{1m} \\ \vdots & \dots & \vdots \\ e_{m1} & \cdots & e_{mm} \end{bmatrix},$$

where the n-vectors e_{ij} have kth element

$$(e_{ij})_k = \begin{cases} \lambda_{i0}^{1/2}\lambda_{ik}^{-1} - \lambda_{i0}^{1/2}\lambda_{ik}^{-2}d_k, & i = j, \\ -\lambda_{j0}^{1/2}\lambda_{ik}^{-1}\lambda_{jk}^{-1}d_k, & i \neq j. \end{cases} \tag{30}$$

Thus,

$$\boldsymbol{A}_2 = \boldsymbol{A}_1 - \boldsymbol{E}\boldsymbol{M}^{-1}\boldsymbol{E}'. \tag{31}$$

Finally, set $\boldsymbol{A}_3 = (\boldsymbol{A}_2^{-1} + \lambda_{00}\boldsymbol{J}_{mn})^{-1} = (\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}$. This is just a rank one update, so

$$\boldsymbol{A}_3 = \boldsymbol{A}_2 - \frac{\boldsymbol{A}_2\boldsymbol{1}_{mn}\boldsymbol{1}_{mn}'\boldsymbol{A}_2}{1/\lambda_{00} + \boldsymbol{1}_{mn}'\boldsymbol{A}_2\boldsymbol{1}_{mn}}. \tag{32}$$

4.3. Computation of the adjusted forecasts

To compute the adjusted unaggregated forecasts efficiently when m and/or n are large, we must compute $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}$ without explicitly forming $\boldsymbol{S}$ or $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})$. The formulae in the previous section suggest the following procedure:

1. Suppose we arrange $\boldsymbol{Z} = \boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}$ as a $m \times n$ matrix. Then the i, j element is

$$z_{ij} = \lambda_{00}\hat{y}_{00} + \lambda_{i0}\hat{y}_{i0} + \lambda_{0j}\hat{y}_{0j} + \lambda_{ij}\hat{y}_{ij}. \tag{33}$$

2. To compute $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}} = \boldsymbol{A}_3\boldsymbol{Z}$, (regarding $\boldsymbol{Z}$ as a vector) we use the formulae of the last section.
   2.1. From (32) we have

$$\boldsymbol{A}_3\boldsymbol{Z} = \boldsymbol{A}_2\boldsymbol{Z} - \frac{(\boldsymbol{1}_{mn}'\boldsymbol{A}_2\boldsymbol{Z})\boldsymbol{A}_2\boldsymbol{1}_{mn}}{1/\lambda_{00} + \boldsymbol{1}_{mn}'\boldsymbol{A}_2\boldsymbol{1}_{mn}}. \tag{34}$$

   which involves the computation of $\boldsymbol{A}_2\boldsymbol{Z}$ and $\boldsymbol{A}_2\boldsymbol{1}_{mn}$.
   2.2 These are computed using

$$\boldsymbol{A}_2\boldsymbol{Z} = \boldsymbol{A}_1\boldsymbol{Z} - \boldsymbol{E}\boldsymbol{M}^{-1}\boldsymbol{E}'\boldsymbol{Z}, \tag{35}$$

   and

$$\boldsymbol{A}_2\boldsymbol{1}_{mn} = \boldsymbol{A}_1\boldsymbol{1}_{mn} - \boldsymbol{E}\boldsymbol{M}^{-1}\boldsymbol{E}'\boldsymbol{1}_{mn}, \tag{36}$$

   which involves computation of $\boldsymbol{A}_1\boldsymbol{Z}, \boldsymbol{A}_1\boldsymbol{1}_{mn}, \boldsymbol{E}'\boldsymbol{Z}, \boldsymbol{E}'\boldsymbol{1}_{mn}$ and $\boldsymbol{E}\boldsymbol{v}$ where the m-vector $\boldsymbol{v}$ is either $\boldsymbol{M}^{-1}\boldsymbol{E}'\boldsymbol{Z}$ or $\boldsymbol{M}^{-1}\boldsymbol{E}'\boldsymbol{1}_{mn}$.
   2.3 Computation of the m-vector $\boldsymbol{E}'\boldsymbol{Z}$ can be done efficiently using the equations

$$(\boldsymbol{E}'\boldsymbol{Z})_i = \lambda_{i0}^{1/2}\sum_j \lambda_{ij}^{-1}z_{ij} - \lambda_{i0}^{1/2}\sum_j\sum_k \lambda_{ij}^{-1}\lambda_{kj}^{-1}d_j z_{kj}, \tag{37}$$

   with a similar equation for $\boldsymbol{E}'\boldsymbol{1}_{mn}$. If we arrange the mn-vector $\boldsymbol{E}\boldsymbol{v}$ as a matrix, its i, j element is

$$(\boldsymbol{E}\boldsymbol{v})_{ij} = \lambda_{i0}^{1/2}\lambda_{ij}^{-1}v_i - \lambda_{ij}^{-1}d_j\sum_k \lambda_{k0}^{1/2}\lambda_{kj}^{-1}v_k. \tag{38}$$

   2.4 From (27), and arranging $\boldsymbol{A}_1\boldsymbol{Z}$ as a matrix, we have

$$(\boldsymbol{A}_1\boldsymbol{Z})_{ij} = \lambda_{ij}^{-1}z_{ij} - \lambda_{ij}^{-1}d_j z_{ij}\sum_k \lambda_{kj}^{-1}, \tag{39}$$

   with a similar equation for $\boldsymbol{A}_1\boldsymbol{1}_{mn}$.

To summarize, the algorithm for the computation of $(\boldsymbol{S}'\boldsymbol{\Lambda}\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}$ proceeds as follows:

1. Calculate $\boldsymbol{Z} = \boldsymbol{S}'\boldsymbol{\Lambda}\hat{\boldsymbol{y}}$ using (33).
2. Calculate $\boldsymbol{A}_1\boldsymbol{Z}$ and $\boldsymbol{A}_1\mathbf{1}_{mn}$ using (39).
3. Calculate $\boldsymbol{E}'\boldsymbol{Z}$ and $\boldsymbol{E}'\mathbf{1}_{mn}$ using (37).
4. Calculate $\boldsymbol{M}^{-1}\boldsymbol{E}'\boldsymbol{Z}$ and $\boldsymbol{M}^{-1}\boldsymbol{E}'\mathbf{1}_{mn}$ using (29) and solving the linear system. Calculate $\boldsymbol{E}\boldsymbol{M}^{-1}\boldsymbol{E}'\boldsymbol{Z}$ and $\boldsymbol{E}\boldsymbol{M}^{-1}\boldsymbol{E}'\mathbf{1}_{mn}$ using (38).
5. Calculate $\boldsymbol{A}_2\boldsymbol{Z}$ and $\boldsymbol{A}_2\mathbf{1}_{mn}$ using (35) and (36).
6. Calculate $\boldsymbol{A}_3\boldsymbol{Z}$ using (34).

The major computational task is the calculation of $\boldsymbol{M}^{-1}\boldsymbol{E}\boldsymbol{Z}$ but as $\boldsymbol{M}$ is only an $m \times m$ matrix this should present no difficulty. Without loss of generality we can assume that $m \le n$ (otherwise interchange rows and columns).

The matrices involved in this approach are generally smaller than those for hierarchical time series, and so the computation times are less than those reported in the previous section.

5. Sparse matrix solutions

An alternative to the methods described above is provided by exploiting the sparsity of the matrices $\boldsymbol{S}$ and $\boldsymbol{S}'\boldsymbol{S}$ and performing matrix computations using a sparse matrix package. Sparse matrix algebra is useful whenever matrices contain a large proportion of zeros. Then common matrix operations can be greatly speeded up, by avoiding many multiplications by zeros for example Bunch and Rose (1976).

There are several sparse matrix routines for solving linear systems of equations including the LSQR algorithm of Paige and Saunders (1982a,b) and the Sparskit algorithms of Saad (1994). We have used the latter in our calculations, as implemented in the `SparseM` package for R (Koenker and Ng, 2015, 2003) and incorporated into the `hts` package.

The Sparskit routines are written in FORTRAN. In contrast, the `hts` implementation of the recursive method is in native R code with no C or FORTRAN components and no attempt to optimize the code for speed. This makes it much faster to write and maintain the code, but substantially slower to run. Thus we would expect the sparse matrix approach to be faster in the current implementation.

The sparse matrix approach also has the advantage of being applicable to any type of aggregation structure, and so is much more general than the approaches described in the previous two sections. On the other hand, because it is general, there is no opportunity to exploit specific patterns in the structure of the design matrix.

In this section we contrast the performance of the recursive hierarchical approach with the sparse matrix approach.

We compare the methods for hierarchical time series in three basic scenarios:

Scenario 1 where the bottom level has many child nodes per parent node, but the upper levels have many fewer child nodes per parent node;
Scenario 2 where the number of child nodes per parent node increases as we go down the levels;
Scenario 3 where the number of child nodes per parent node remains constant as we go down the levels.

We generated several hierarchical structures by specifying the number of child nodes per parent node at each level. Thus, in the case of three levels, the structure 2,3,4 denotes a structure with two nodes at level 1, six nodes at level 2, and 24 nodes at level 3. For each node, we adjusted the forecasts using the recursive method and the sparse method, using the `solve` function in the package `SparseM`. The results are shown in Table 2.

For Scenario 1, the recursive method is competitive, given the implementation differences. For moderate size problems under scenario 2, it is still competitive for three-level hierarchies, but not for bigger problems. It is competitive under scenario 3 only for small problems. Table 2 only shows results for the unweighted case but the situation for weights is similar.

We repeated the experiment for grouped time series, using various values of m (the number of rows), and n (the number of columns). In every case, for the unweighted adjustment, direct evaluation of the formula

$$\begin{aligned}&\big\{\hat{y}_{00} - \hat{y}_{0+} - \hat{y}_{+0} + \hat{y}_{++} + (m+1)\hat{y}_{0j} + (n+1)\hat{y}_{i0}\\&\quad + (m+1)(n+1)\hat{y}_{ij} - (m+1)\hat{y}_{i+} - (n+1)\hat{y}_{+j} + \hat{y}_{++}\big\} \big/ (m+1)(n+1),\end{aligned} \tag{40}$$

is very much faster than the sparse approach, as one would expect. In the weighted case, the comparisons are shown in Table 3.

In the grouped case, the method based on the Sherman–Morrison–Woodbury formula is much more competitive, again given the implementation differences, provided $\max(m, n)$ is not too large.

6. Application to forecasting the Australian labour market

The Australian and New Zealand Standard Classification of Occupations (ANZSCO) provides a detailed hierarchical classification of occupations (ABS, 2013). Occupations are organized into progressively larger groups based on the

Table 2
Timings in seconds to reconcile forecasts. Total number of nodes in the hierarchy are also shown.

Hierarchy	Recursive method	Sparse method	Recursive/sparse	Nodes
Scenario 1				
5, 5, 200	0.01	0.02	0.50	5 031
10, 10, 500	0.16	0.22	0.73	50 111
5, 5, 5, 200	0.14	0.05	2.80	25 156
10, 10, 10, 500	3.16	0.87	3.63	501 111
5, 5, 5, 5, 100	0.61	0.12	5.08	63 281
5, 5, 5, 5, 200	0.92	0.23	4.00	125 781
Scenario 2				
5, 20, 100	0.08	0.03	2.67	10 106
10, 40, 200	0.76	0.25	3.04	80 411
5, 15, 30, 100	6.00	0.40	15.00	227 331
10, 30, 50, 200	140.90	6.63	21.25	3 015 311
2, 5, 10, 20, 50	4.32	0.17	25.41	102 113
5, 10, 15, 20, 60	119.08	1.66	71.73	915 806
Scenario 3				
10, 10, 10	0.04	0.01	4.00	1 111
20, 20, 20	0.27	0.02	13.50	8 421
5, 5, 5, 5	0.05	0.02	2.50	781
10, 10, 10, 10	0.55	0.05	11.00	11 111
15, 15, 15, 15	5.88	0.08	73.50	54 241
5, 5, 5, 5, 5	0.22	0.03	7.33	3 906
10, 10, 10, 10, 10	42.82	0.85	50.38	111 111

Table 3
Timings in seconds to reconcile forecasts. Total number of nodes in the hierarchy are also shown.

m	n	Present method	Sparse method	Recursive/sparse	Nodes
50	50	0.02	0.03	0.67	2 500
75	75	0.05	0.05	1.00	5 625
100	100	0.18	0.11	1.64	10 000
150	150	1.04	0.24	4.33	22 500
10	100	0.01	0.01	1.00	1 000
10	500	0.03	0.06	0.50	5 000
50	200	0.18	0.11	1.64	10 000
50	500	3.48	0.26	13.38	25 000

similarities of skill specialities and levels. ANZSCO (version 1.2) lists 1023 occupations, grouped into 359 unit groups. These, in turn, form 97 minor groups, which make up 43 sub-major groups, which combine to give 8 major groups of occupations. For example, a statistician has ANZSCO code 224113, meaning it sits within the hierarchy as follows.

2 Professionals
  22 Business, Human Resource and Marketing Professionals
    224 Information and Organization Professionals
      2241 Actuaries, Mathematicians and Statisticians
        224113 Statistician

Each of the first four digits represents the group at that level, the last two digits denote the occupation group at the bottom level of the hierarchy.

Detailed data on the occupation groups are only recorded for censuses, held every five years. However, the Australian Labour Force Survey provides data down to the level of unit groups for each quarter and for each of the states of Australia, based on a multi-stage area sample of approximately 26,000 private dwellings (ABS, 2014). Quarterly data on the numbers of people employed in each unit group and each state were obtained from the Australian Bureau of Statistics for the period 1986:Q3–2013:Q2.

So the hierarchy consists of 359 unit groups, 97 minor groups, 43 sub-major groups, 8 major groups, and the total, giving 508 time series, each of length 108 observations. These are then further disaggregated into seven regions (the 6 Australian states and the Northern Territory). Not all occupations are present in every state; at the most disaggregated level we have 2864 time series. Compared to typical examples arising in manufacturing and sales, this is still a relatively small collection of time series, but is sufficient to demonstrate our algorithms.

Purely hierarchical time series

First, we consider a strictly hierarchical collection of time series by aggregating the data across states, leaving only the hierchical grouping structure defined by the ANZSCO codes.

Table 4
Out-of-sample forecasting performance: Australian labour data disaggregated only by ANZSCO occupation codes to level 4. A rolling forecast origin is used, beginning with a training set of 24 observations and forecasting up to 8 steps ahead.

RMSE	Forecast horizon (h)								Average
	1	2	3	4	5	6	7	8	
Top level									
BU	74.71	102.02	121.70	131.17	147.08	157.12	169.60	178.93	135.29
OLS	52.20	77.77	101.50	119.03	138.27	150.75	160.04	166.38	120.74
WLS	61.77	86.32	107.26	119.33	137.01	146.88	156.71	162.38	122.21
ANZSCO level 1									
BU	21.59	27.33	30.81	32.94	35.45	37.10	39.00	40.51	33.09
OLS	21.89	28.55	32.74	35.58	38.82	41.24	43.34	45.49	35.96
WLS	20.58	26.19	29.71	31.84	34.36	35.89	37.53	38.86	31.87
ANZSCO level 2									
BU	8.78	10.72	11.79	12.42	13.13	13.61	14.14	14.65	12.40
OLS	9.02	11.19	12.34	13.04	13.92	14.56	15.17	15.77	13.13
WLS	8.58	10.48	11.54	12.15	12.88	13.36	13.87	14.36	12.15
ANZSCO level 3									
BU	5.44	6.57	7.17	7.53	7.94	8.27	8.60	8.89	7.55
OLS	5.55	6.78	7.42	7.81	8.29	8.68	9.04	9.37	7.87
WLS	5.35	6.46	7.06	7.42	7.84	8.17	8.48	8.76	7.44
ANZSCO level 4									
BU	2.35	2.79	3.02	3.15	3.29	3.42	3.54	3.65	3.15
OLS	2.40	2.86	3.10	3.24	3.41	3.55	3.68	3.80	3.25
WLS	2.34	2.77	2.99	3.12	3.27	3.40	3.52	3.63	3.13

Fig. 6 shows some of the time series obtained after aggregating across states. The top panel shows the total number of people in the labour force aggregated across all occupations and all states. The panel marked "Level 1" shows the total number of people in each of the 8 major groups. The remaining panels show only one time series per major group; specifically, the largest sub-group at each level from each of the major groups.

We fitted ARIMA models using the automatic algorithm of Hyndman and Khandakar (2008), as implemented in Hyndman (2015), to all series in the hierarchy. Thus, a separate ARIMA model was independently estimated for each of the 508 time series, and forecasts were obtained. These were then combined using the weighted least squares reconciliation approach defined in (3), where the weights were specified as the inverse of the one-step residual variances for each of the series. Fig. 7 shows some examples of the original ARIMA (base) forecasts, along with the adjusted forecasts after reconciliation. The examples shown are those with the largest changes in the one-step forecast at each level.

To compare the accuracy of our approach with alternative strategies for hierarchical forecasting, we computed forecasts based on a rolling forecast origin for the Australian labour market data. Beginning with a training set of 24 observations, we forecast eight steps ahead, and computed forecast errors using the first eight observations of the withheld data. Then the training set was increased by one observation and new forecast errors were obtained. The training set continued to increase in size, one observation at a time, until it comprised all but the last observation. In this way, we computed many forecast errors for horizons 1–8, and averaged their squares to obtain out-of-sample mean squared errors for each horizon.

We repeated this exercise for bottom-up forecasts, top-down forecasts with the proportions of disaggregation specified by the "forecast proportions" approach of Athanasopoulos et al. (2009), OLS reconciled forecasts and WLS reconciled forecasts. The resulting root MSE values are given in Table 4. The results show that on average the WLS reconciliation approach outperforms the other approaches at all levels except the very top level where the OLS method does slightly better.

Time series with two groups

To demonstrate our algorithm involving two groups, we consider the series obtained by disaggregating on state and occupations at ANZSCO Level 4. Thus, the summing matrix consists of rows defining the aggregations by states, and the aggregations by occupation. As these do not form a hierarchy (there is no natural ordering to the two grouping variables), we apply the algorithm described in Section 4. The same rolling forecast as was done for the purely hierarchical example was used here as well. The results are shown in Table 5.

Mixed hierarchy with groups

To demonstrate a more complicated example, we consider the entire collection of 2864 time series, with grouping by occupation code and by state. The summing matrix includes aggregations that would be obtained from the hierarchical occupation codes, from the states, and from the cross-product of the state grouping variable with the occupational hierarchy. For this more general problem, we have used the sparse matrix routines described in Section 5 to do the calculations.

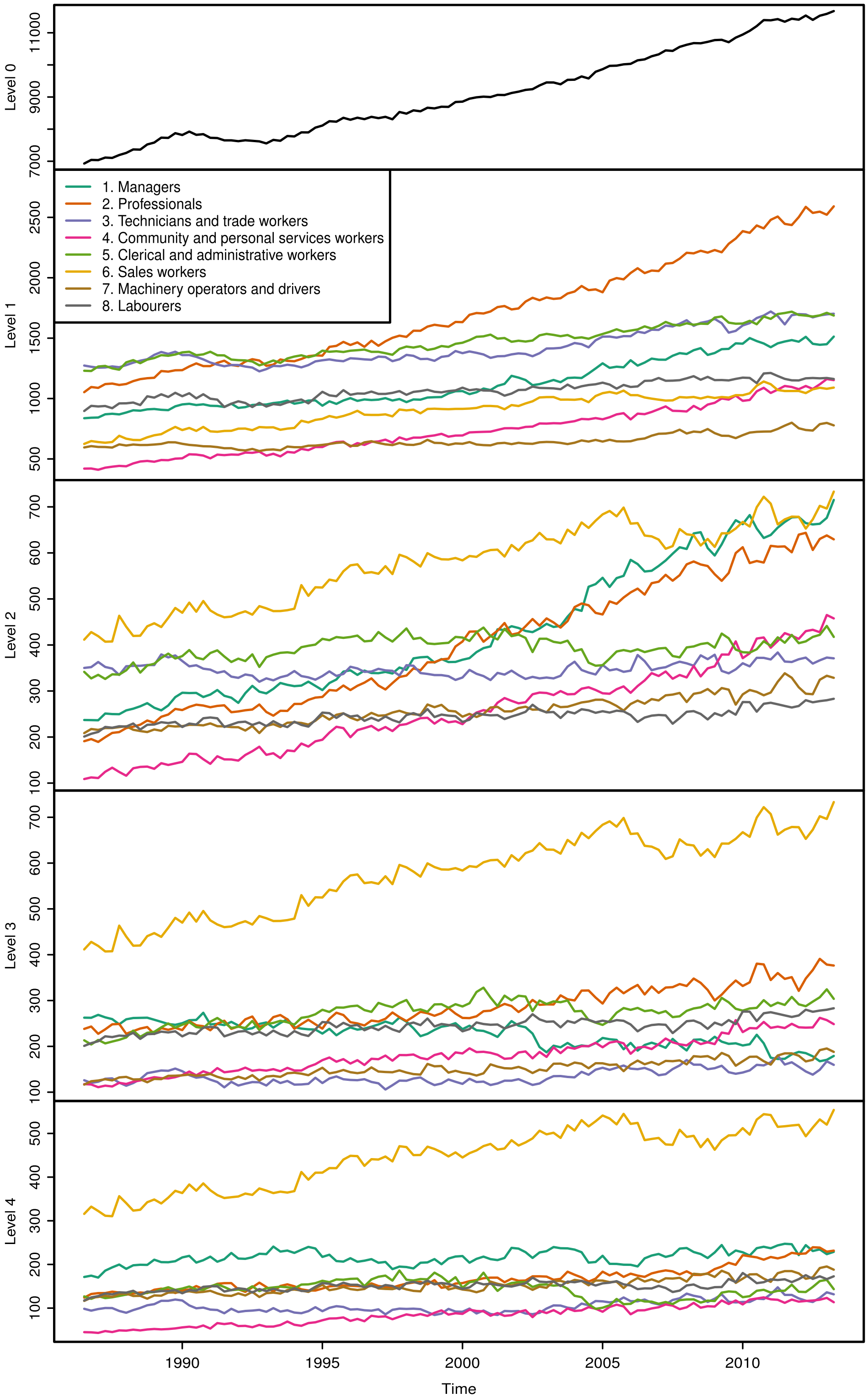

Fig. 6. Some of the 508 time series in the Australian labour force hierarchy after aggregating across states. Top: total number of people aggregated across all occupations; Level 1 shows the total number of people in each of the 8 major occupation groups; the remaining panels show the largest sub-group at each level from each of the major groups.

Again, we used the same rolling forecast origin as in the previous examples. The results (shown in Table 6) can be compared with the simpler structures in the previous two tables. The larger, more disaggregated, structure has led to

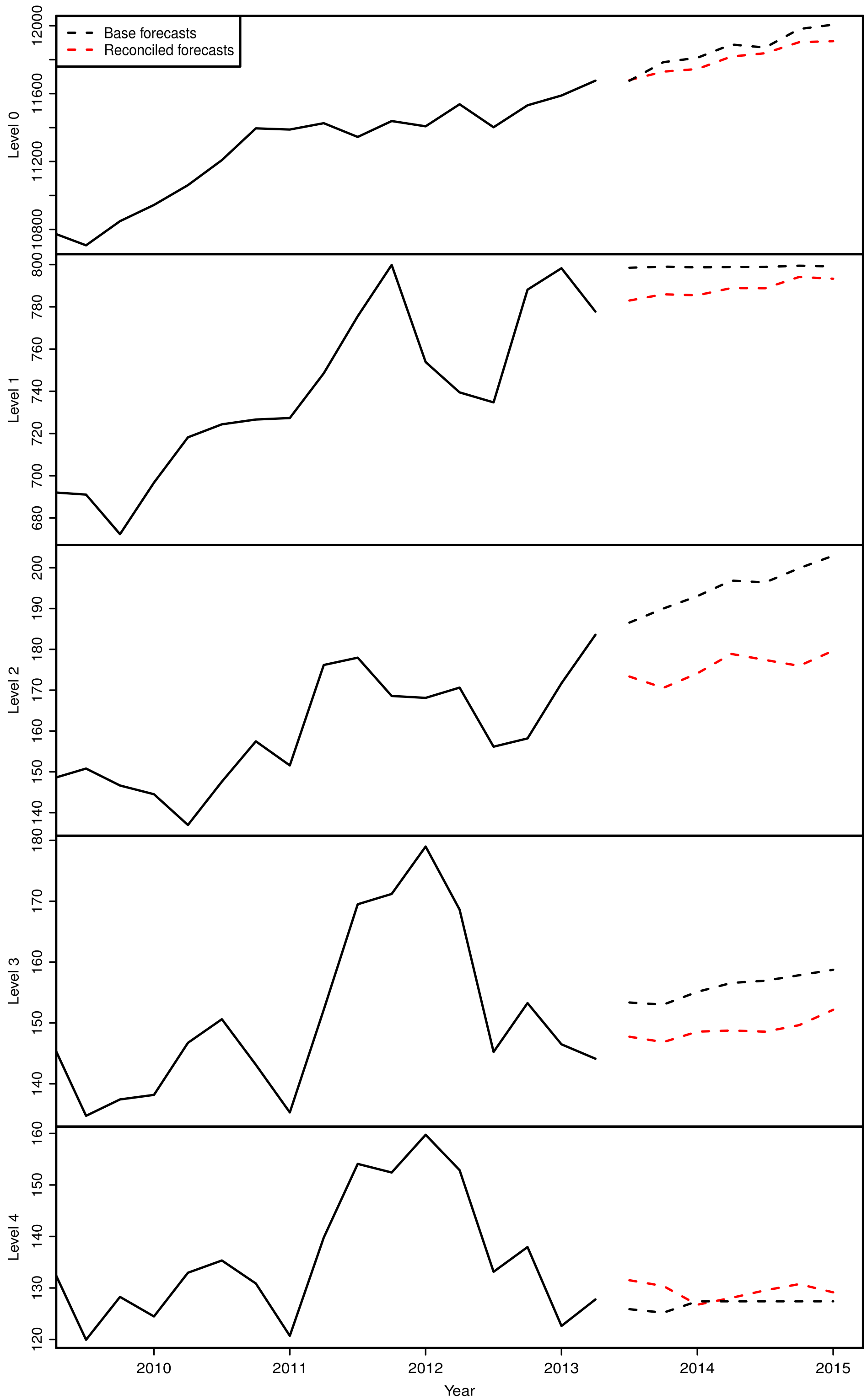

Fig. 7. Some examples of the base forecasts and the reconciled forecasts. The series plotted are those with the largest changes in the one-step forecast at each level.

more accurate forecasts in general as we are able to "borrow strength" from the time series patterns seen in the various aggregations.

Table 5
Out-of-sample forecasting performance: Australian labour data disaggregated by state and ANZSCO occupation code level 4. A rolling forecast origin is used, beginning with a training set of 24 observations and forecasting up to 8 steps ahead.

RMSE	Forecast horizon (h)								Average
	1	2	3	4	5	6	7	8	
Top level									
BU	98.35	137.54	161.95	176.20	191.76	203.68	217.87	230.51	177.23
OLS	53.02	79.13	103.68	122.29	142.80	156.54	167.10	174.84	124.92
WLS	63.85	88.89	111.04	124.25	142.84	154.19	166.09	174.08	128.16
States									
BU	15.77	21.38	24.64	26.62	28.78	30.54	32.52	34.18	26.80
OLS	11.48	15.47	19.40	21.81	25.18	27.54	29.82	31.40	22.76
WLS	11.61	15.36	18.44	20.39	22.96	24.90	26.71	28.06	21.05
ANZSCO level 4									
BU	2.37	2.81	3.03	3.16	3.30	3.42	3.53	3.64	3.16
OLS	2.33	2.77	2.99	3.13	3.27	3.40	3.51	3.63	3.13
WLS	2.31	2.74	2.95	3.08	3.22	3.34	3.46	3.56	3.08
States × ANZSCO level 4									
BU	0.63	0.73	0.76	0.78	0.80	0.82	0.84	0.86	0.78
OLS	0.64	0.74	0.78	0.80	0.82	0.84	0.86	0.88	0.80
WLS	0.63	0.72	0.76	0.78	0.80	0.82	0.84	0.85	0.77

Table 6
Out-of-sample forecasting performance: Australian labour data disaggregated by state and by ANZSCO occupation codes down to level 4. A rolling forecast origin is used, beginning with a training set of 24 observations and forecasting up to 8 steps ahead.

RMSE	Forecast horizon (h)								Average
	1	2	3	4	5	6	7	8	
Top level									
BU	98.35	137.54	161.95	176.20	191.76	203.68	217.87	230.51	177.23
OLS	52.35	77.75	101.65	119.09	138.53	151.09	160.55	166.92	120.99
WLS	68.22	93.49	114.67	125.62	142.32	151.06	161.68	167.87	128.12
States									
BU	15.77	21.38	24.64	26.62	28.78	30.54	32.52	34.18	26.80
OLS	11.15	14.89	18.53	20.78	23.87	26.04	28.07	29.44	21.60
WLS	11.98	15.74	18.63	20.37	22.73	24.37	26.02	27.12	20.87
ANZSCO level 1									
BU	23.88	30.74	34.46	36.83	39.17	40.98	42.96	44.89	36.74
OLS	21.82	28.38	32.51	35.34	38.55	40.96	43.04	45.15	35.72
WLS	21.00	26.42	29.72	31.71	34.08	35.57	37.28	38.66	31.80
ANZSCO level 2									
BU	9.15	11.28	12.36	13.02	13.67	14.16	14.70	15.25	12.95
OLS	8.96	11.09	12.21	12.90	13.77	14.39	14.98	15.56	12.98
WLS	8.57	10.41	11.42	12.01	12.68	13.13	13.61	14.07	11.99
ANZSCO level 3									
BU	5.55	6.74	7.34	7.73	8.11	8.43	8.74	9.04	7.71
OLS	5.51	6.72	7.34	7.73	8.20	8.59	8.93	9.25	7.78
WLS	5.30	6.38	6.95	7.30	7.69	8.00	8.29	8.56	7.31
ANZSCO level 4									
BU	2.37	2.81	3.03	3.16	3.30	3.42	3.53	3.64	3.16
OLS	2.38	2.84	3.07	3.21	3.37	3.51	3.63	3.76	3.22
WLS	2.30	2.72	2.93	3.06	3.20	3.32	3.43	3.53	3.06
States × ANZSCO level 1									
BU	5.51	6.71	7.21	7.50	7.80	8.08	8.36	8.60	7.47
OLS	5.51	6.76	7.34	7.67	8.10	8.45	8.78	9.07	7.71
WLS	5.15	6.19	6.63	6.86	7.15	7.37	7.61	7.78	6.84
States × ANZSCO level 2									
BU	2.29	2.70	2.86	2.95	3.04	3.12	3.20	3.27	2.93
OLS	2.33	2.76	2.95	3.05	3.17	3.27	3.36	3.45	3.04
WLS	2.22	2.60	2.76	2.84	2.93	3.00	3.08	3.14	2.82
States × ANZSCO level 3									
BU	1.45	1.69	1.79	1.84	1.90	1.94	1.99	2.03	1.83
OLS	1.47	1.74	1.85	1.90	1.97	2.03	2.09	2.13	1.90
WLS	1.42	1.65	1.75	1.79	1.85	1.90	1.94	1.98	1.78
States × ANZSCO level 4									
BU	0.63	0.73	0.76	0.78	0.80	0.82	0.84	0.86	0.78
OLS	0.65	0.75	0.80	0.82	0.84	0.87	0.89	0.91	0.82
WLS	0.62	0.72	0.75	0.77	0.79	0.81	0.83	0.85	0.77

7. Discussion and conclusions

We have shown how the reconciliation model of Hyndman et al. (2011) can be naturally applied in much more general situations, wherever forecasts need to be reconciled under aggregation constraints.

The computational algorithms discussed in this paper provide a way of handling enormous collections of time series. In two special but important cases (when the time series are arranged in a hierarchy, and when the time series have a two-way grouping structure), we have shown how to exploit the structure of the aggregation constraints to reduce the computational complexity of the problem. In practice, more than two grouping variables are often available, and hierarchical data often have grouping variables available as well; then the sparse matrix routines described in Section 5 can be used instead.

Our algorithms are implemented in the **hts** package for R (Hyndman et al., 2015). Very general hierarchical and grouping structures are possible. When the time series are strictly hierarchical, the algorithm described in Section 3 may be used. When the time series are grouped in a two-way structure, the algorithm described in Section 4 may be used. However, given the results in Section 5, the default approach is to use sparse matrix algebra for all cases. The package provides facilities for creation of the $\boldsymbol{S}$ matrix and estimation of $\boldsymbol{\Lambda}$ from one-step forecast error variances, without requiring user input. The base forecasts are generated automatically using one of the algorithms described in Hyndman and Khandakar (2008).

An important research need is the estimation of the covariance matrix $\boldsymbol{\Sigma}_h$ so that the more general GLS solution (2) can be used. This is difficult because of the sheer size of the matrix, and standard estimators give very poor results. Even if $\boldsymbol{\Sigma}_h$ could be accurately estimated, computation of the reconciled forecasts would remain an issue as our fast algorithms do not apply in that case, and the matrices requiring inversion will not, in general, be sparse. We leave this problem for a later paper.

Acknowledgements

Thanks to Dr. Chandra Shah for assistance with the Australian labour market data, and for translating the older ASCO codes into ANZSCO codes, to give a consistent classification of occupations across the period of the available data.

References

ABS, 2013. ANZSCO—Australian and New Zealand Standard Classification of Occupations. Cat. 1220.0, Belconnen, ACT, Australia. www.abs.gov.au/AUSSTATS/abs@.nsf/mf/1220.0.
ABS, 2014. Labor Force, Australia. Cat. 6202.0, Belconnen, ACT, Australia. www.abs.gov.au/AUSSTATS/abs@.nsf/mf/6202.0.
Athanasopoulos, G., Ahmed, R.A., Hyndman, R.J., 2009. Hierarchical forecasts for Australian domestic tourism. Int. J. Forecast. 25 (1), 146–166.
Bunch, J.R., Rose, D.J., 1976. Sparse Matrix Computations. Academic Press, New York.
Capistrán, C., Constandse, C., Ramos-Francia, M., 2010. Multi-horizon inflation forecasts using disaggregated data. Ecol. Modell. 27 (3), 666–677.
Di Fonzo, T., Marini, M., 2011. Simultaneous and two-step reconciliation of systems of time series: methodological and practical issues. J. Roy. Statist. Soc. Ser. C 60 (2), 143–164.
Fliedner, G., 2001. Hierarchical forecasting: issues and use guidelines. Ind. Manag. Data Syst. 101 (1), 5–12.
Hyndman, R.J., 2015. forecast: Forecasting functions for time series and linear models. R package version 6.2. cran.r-project.org/package=forecast.
Hyndman, R.J., Ahmed, R.A., Athanasopoulos, G., Shang, H.L., 2011. Optimal combination forecasts for hierarchical time series. Comput. Statist. Data Anal. 55 (9), 2579–2589.
Hyndman, R.J., Khandakar, Y., 2008. Automatic time series forecasting: the forecast package for R. J. Stat. Softw. 26 (3), 1–22.
Hyndman, R.J., Koehler, A.B., Ord, J.K., Snyder, R.D., 2008. Forecasting with Exponential Smoothing: The State Space Approach. Springer-Verlag, Berlin.
Hyndman, R.J., Lee, A.J., Wang, E., 2015. hts: Hierarchical and Grouped Time Series. R package version 4.5. cran.r-project.org/package=hts.
Koenker, R., Ng, P., 2003. SparseM: A sparse matrix package for R. J. Stat. Softw. 8 (6), 1–9.
Koenker, R., Ng, P., 2015. SparseM: Sparse Linear Algebra. R package version 1.7. http://CRAN.R-project.org/package=SparseM.
Paige, C.C., Saunders, M.A., 1982a. Algorithm 583: LSQR: Sparse linear equations and least squares problems. ACM Trans. Math. Software (TOMS) 8 (2), 195–209.
Paige, C.C., Saunders, M.A., 1982b. LSQR: An algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Software (TOMS) 8 (1), 43–71.
Quenneville, B., Fortier, S., 2012. Restoring accounting constraints in time series: methods and software for a statistical agency. In: Bell, W.R., Holan, S.H., McElroy, T.S. (Eds.), Economic Time Series: Modeling and Seasonality. Taylor & Francis, pp. 231–253.
Saad, Y., 1994. SPARSKIT: a basic tool kit for sparse matrix computations, Technical Report. Computer Science Department, University of Minnesota.
Seber, G.A.F., Lee, A.J., 2003. Linear Regression Analysis. Wiley-Interscience.