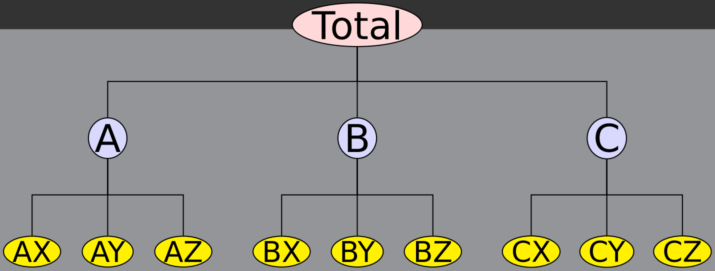
Rob J Hyndman

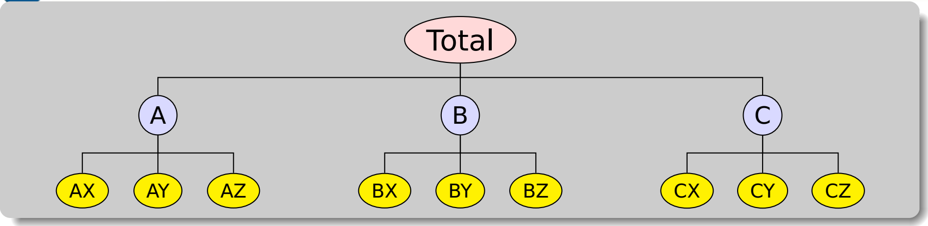
Fast computation of reconciled forecasts in hierarchical and grouped time series



Outline

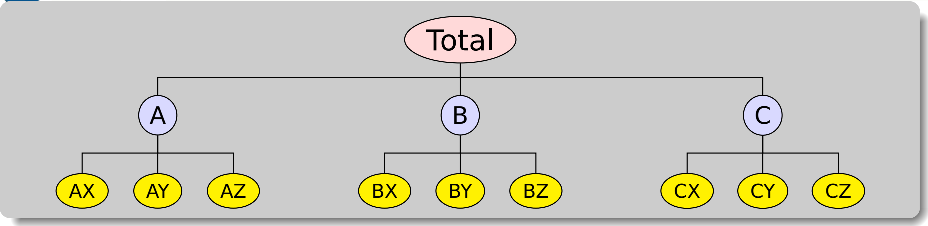
1. **Optimally reconciled forecasts**
2. Fast computation
3. hts package for R
4. References

Hierarchical data



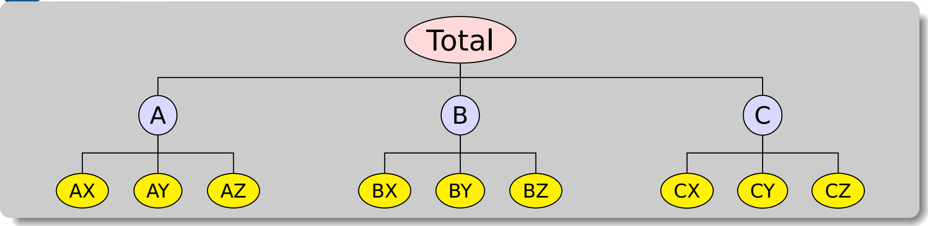
$$\mathbf{Y}_t = \begin{pmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{C,t} \\ Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{S}} \underbrace{\begin{pmatrix} Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix}}_{\mathbf{B}_t}$$

Hierarchical data



$$\boldsymbol{Y}_t = \begin{pmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{C,t} \\ Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{S}} \underbrace{\begin{pmatrix} Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix}}_{\boldsymbol{B}_t}$$

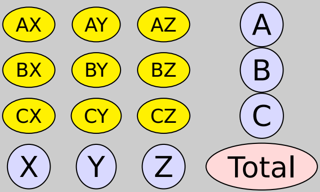
Hierarchical data



$$\boldsymbol{Y}_t = \begin{pmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{C,t} \\ Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{S}} \underbrace{\begin{pmatrix} Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix}}_{\boldsymbol{B}_t}$$

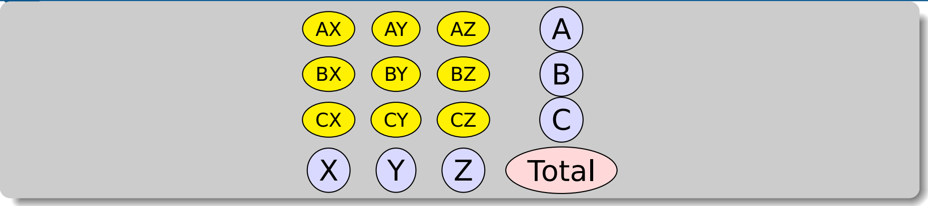
$$\boxed{\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t}$$

Grouped data



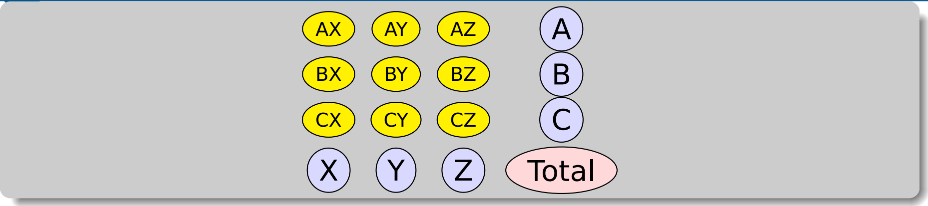
$$
\boldsymbol{Y}_t = \begin{pmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{C,t} \\ Y_{X,t} \\ Y_{Y,t} \\ Y_{Z,t} \\ Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix}}_{\boldsymbol{B}_t}
$$

Grouped data



$$\boldsymbol{Y}_t = \begin{pmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{C,t} \\ Y_{X,t} \\ Y_{Y,t} \\ Y_{Z,t} \\ Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix}}_{\boldsymbol{B}_t}$$

Grouped data



$$\boldsymbol{Y}_t = \begin{pmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{C,t} \\ Y_{X,t} \\ Y_{Y,t} \\ Y_{Z,t} \\ Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix}}_{\boldsymbol{B}_t}$$

$$\boxed{\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t}$$

Forecasts

Key idea: forecast reconciliation

- Ignore structural constraints and forecast every series of interest independently.
- Adjust forecasts to impose constraints.

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$.

$\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t$. So $\hat{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\beta_n(h) + \varepsilon_h$.

Forecasts

Key idea: forecast reconciliation

- Ignore structural constraints and forecast every series of interest independently.
- Adjust forecasts to impose constraints.

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$.

$\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t$. So $\hat{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\beta_n(h) + \varepsilon_h$.

Forecasts

Key idea: forecast reconciliation

- Ignore structural constraints and forecast every series of interest independently.
- Adjust forecasts to impose constraints.

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$.

$\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t$. So $\hat{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\beta_n(h) + \varepsilon_h$.

- $\beta_n(h) = \mathrm{E}[\boldsymbol{B}_{n+h} \mid \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$.
- ε_h has zero mean.
- Estimate $\beta_n(h)$ using weighted least squares.

Forecasts

Key idea: forecast reconciliation

- Ignore structural constraints and forecast every series of interest independently.
- Adjust forecasts to impose constraints.

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$.

$\boxed{\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t}$. So $\boxed{\hat{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\beta_n(h) + \varepsilon_h}$.

- $\beta_n(h) = \mathrm{E}[\boldsymbol{B}_{n+h} \mid \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$.
- ε_h has zero mean.
- Estimate $\beta_n(h)$ using weighted least squares.

Forecasts

Key idea: forecast reconciliation

- Ignore structural constraints and forecast every series of interest independently.
- Adjust forecasts to impose constraints.

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$.

$\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t$. So $\hat{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\beta_n(h) + \varepsilon_h$.

- $\beta_n(h) = \mathrm{E}[\boldsymbol{B}_{n+h} \mid \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$.
- ε_h has zero mean.
- Estimate $\beta_n(h)$ using weighted least squares.

Forecasts

Key idea: forecast reconciliation

- Ignore structural constraints and forecast every series of interest independently.
- Adjust forecasts to impose constraints.

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$.

$\boxed{\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t}$. So $\boxed{\hat{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\beta_n(h) + \varepsilon_h}$.

- $\beta_n(h) = \mathrm{E}[\boldsymbol{B}_{n+h} \mid \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$.
- ε_h has zero mean.
- Estimate $\beta_n(h)$ using weighted least squares.

Forecasts

Key idea: forecast reconciliation

- Ignore structural constraints and forecast every series of interest independently.
- Adjust forecasts to impose constraints.

Let $\hat{\boldsymbol{Y}}_n(h)$ be vector of initial h-step forecasts, made at time n, stacked in same order as $\boldsymbol{Y}_t$.

$\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t$. So $\hat{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\beta_n(h) + \varepsilon_h$.

- $\beta_n(h) = \mathrm{E}[\boldsymbol{B}_{n+h} \mid \boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n]$.
- ε_h has zero mean.
- Estimate $\beta_n(h)$ using weighted least squares.

Optimal reconciled forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\hat{\beta}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}\boldsymbol{S}'\Lambda\hat{\boldsymbol{Y}}_n(h)$$

- Λ weights (usually inverse of one-step forecast variances, but any choice possible).
- Easy to estimate, and places weight where we have best forecasts.
- Ignores covariances.
- Still difficult to compute for large numbers of time series.
- Need to do calculation without explicitly forming S or $(S'\Lambda S)^{-1}$ or $S'\Lambda$.

Optimal reconciled forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\hat{\beta}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}\boldsymbol{S}'\Lambda\hat{\boldsymbol{Y}}_n(h)$$

Initial forecasts

- Λ weights (usually inverse of one-step forecast variances, but any choice possible).
- Easy to estimate, and places weight where we have best forecasts.
- Ignores covariances.
- Still difficult to compute for large numbers of time series.
- Need to do calculation without explicitly forming $\boldsymbol{S}$ or $(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}$ or $\boldsymbol{S}'\Lambda$.

Optimal reconciled forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\hat{\beta}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}\boldsymbol{S}'\Lambda\hat{\boldsymbol{Y}}_n(h)$$

Revised forecasts

Initial forecasts

- Λ weights (usually inverse of one-step forecast variances, but any choice possible).
- Easy to estimate, and places weight where we have best forecasts.
- Ignores covariances.
- Still difficult to compute for large numbers of time series.
- Need to do calculation without explicitly forming $\boldsymbol{S}$ or $(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}$ or $\boldsymbol{S}'\Lambda$.

Optimal reconciled forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\hat{\beta}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}\boldsymbol{S}'\Lambda\hat{\boldsymbol{Y}}_n(h)$$

Revised forecasts

Initial forecasts

- Λ weights (usually inverse of one-step forecast variances, but any choice possible).
- Easy to estimate, and places weight where we have best forecasts.
- Ignores covariances.
- Still difficult to compute for large numbers of time series.
- Need to do calculation without explicitly forming $\boldsymbol{S}$ or $(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}$ or $\boldsymbol{S}'\Lambda$.

Optimal reconciled forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\hat{\beta}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}\boldsymbol{S}'\Lambda\hat{\boldsymbol{Y}}_n(h)$$

Revised forecasts

Initial forecasts

- Λ weights (usually inverse of one-step forecast variances, but any choice possible).
- Easy to estimate, and places weight where we have best forecasts.
- Ignores covariances.
- Still difficult to compute for large numbers of time series.
- Need to do calculation without explicitly forming $\boldsymbol{S}$ or $(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}$ or $\boldsymbol{S}'\Lambda$.

Optimal reconciled forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\hat{\beta}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}\boldsymbol{S}'\Lambda\hat{\boldsymbol{Y}}_n(h)$$

Revised forecasts

Initial forecasts

- Λ weights (usually inverse of one-step forecast variances, but any choice possible).
- Easy to estimate, and places weight where we have best forecasts.
- Ignores covariances.
- Still difficult to compute for large numbers of time series.
- Need to do calculation without explicitly forming $\boldsymbol{S}$ or $(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}$ or $\boldsymbol{S}'\Lambda$.

Optimal reconciled forecasts

$$\tilde{\boldsymbol{Y}}_n(h) = \boldsymbol{S}\hat{\beta}_n(h) = \boldsymbol{S}(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}\boldsymbol{S}'\Lambda\hat{\boldsymbol{Y}}_n(h)$$

Revised forecasts

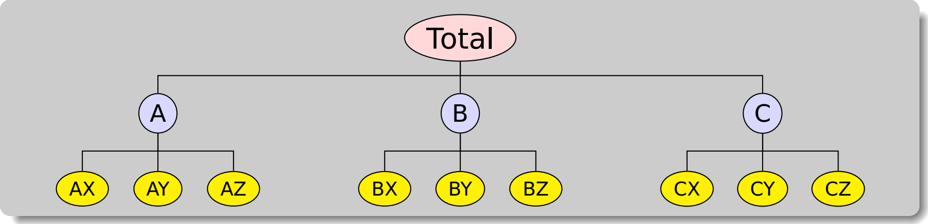
Initial forecasts

- Λ weights (usually inverse of one-step forecast variances, but any choice possible).
- Easy to estimate, and places weight where we have best forecasts.
- Ignores covariances.
- Still difficult to compute for large numbers of time series.
- Need to do calculation without explicitly forming $\boldsymbol{S}$ or $(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1}$ or $\boldsymbol{S}'\Lambda$.

Outline

1. Optimally reconciled forecasts
2. **Fast computation**
3. hts package for R
4. References

Fast computation: hierarchical data

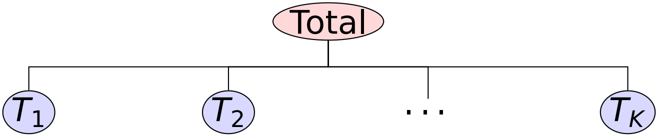


$$
\boldsymbol{Y}_t = \begin{pmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{C,t} \\ Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{S}} \underbrace{\begin{pmatrix} Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix}}_{\boldsymbol{B}_t}
$$

$$
\boxed{\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t}
$$

Fast computation: hierarchies

Think of the hierarchy as a tree of trees:



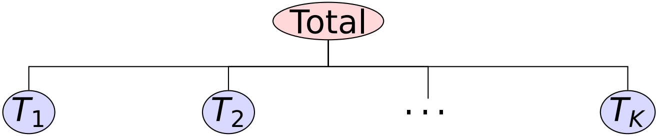
Then the summing matrix contains k smaller summing matrices:

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

where $\mathbf{1}_n$ is an n-vector of ones and tree T_i has n_i terminal nodes.

Fast computation: hierarchies

Think of the hierarchy as a tree of trees:



Then the summing matrix contains k smaller summing matrices:

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

where $\mathbf{1}_n$ is an n-vector of ones and tree T_i has n_i terminal nodes.

Fast computation: hierarchies

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

- λ_0 is the top left element of $\boldsymbol{\Lambda}$;
- $\boldsymbol{\Lambda}_k$ is a block of $\boldsymbol{\Lambda}$, corresponding to tree T_k;
- $\boldsymbol{J}_n$ is a matrix of ones;
- $n = \sum_k n_k$.

Now apply the Sherman-Morrison formula . . .

Fast computation: hierarchies

$$\boldsymbol{S}'\Lambda\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S}_1'\Lambda_1\boldsymbol{S}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_2'\Lambda_2\boldsymbol{S}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{S}_K'\Lambda_K\boldsymbol{S}_K \end{bmatrix} + \lambda_0 \boldsymbol{J}_n$$

- λ_0 is the top left element of Λ;
- Λ_k is a block of Λ, corresponding to tree T_k;
- $\boldsymbol{J}_n$ is a matrix of ones;
- $n = \sum_k n_k$.

Now apply the Sherman-Morrison formula . . .

Fast computation: hierarchies

$$(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1} = \begin{bmatrix} (\boldsymbol{S}_1'\Lambda_1\boldsymbol{S}_1)^{-1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & (\boldsymbol{S}_2'\Lambda_2\boldsymbol{S}_2)^{-1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & (\boldsymbol{S}_K'\Lambda_K\boldsymbol{S}_K)^{-1} \end{bmatrix} - c\boldsymbol{S}_0$$

- $\boldsymbol{S}_0$ can be partitioned into K^2 blocks, with the (k,ℓ) block (of dimension $n_k \times n_\ell$) being

$$(\boldsymbol{S}_k'\Lambda_k\boldsymbol{S}_k)^{-1}\boldsymbol{J}_{n_k,n_\ell}(\boldsymbol{S}_\ell'\Lambda_\ell\boldsymbol{S}_\ell)^{-1}$$

- $\boldsymbol{J}_{n_k,n_\ell}$ is a $n_k \times n_\ell$ matrix of ones.
- $c^{-1} = \lambda_0^{-1} + \sum_k \mathbf{1}_{n_k}'(\boldsymbol{S}_k'\Lambda_k\boldsymbol{S}_k)^{-1}\mathbf{1}_{n_k}.$
- Each $\boldsymbol{S}_k'\Lambda_k\boldsymbol{S}_k$ can be inverted similarly.
- $\boldsymbol{S}'\Lambda\boldsymbol{Y}$ can also be computed recursively.

Fast computation: hierarchies

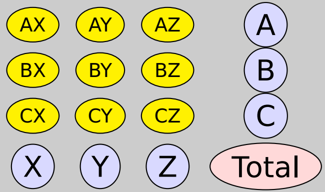
$$(\boldsymbol{S}'\Lambda\boldsymbol{S})^{-1} = \begin{bmatrix} (\boldsymbol{S}_1'\Lambda_1\boldsymbol{S}_1)^{-1} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & (\boldsymbol{S}_2'\Lambda_2\boldsymbol{S}_2)^{-1} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & (\boldsymbol{S}_K'\Lambda_K\boldsymbol{S}_K)^{-1} \end{bmatrix} - c\boldsymbol{S}_0$$

- $\boldsymbol{S}_0$ can be partitioned into K^2 blocks, with the (k,ℓ) block (of dimension $n_k \times n_\ell$) being

The recursive calculations can be done in such a way that we never store any of the large matrices involved.

- $\boldsymbol{J}_{n_k,n_\ell}$
- c^{-1}
- Each $\boldsymbol{S}_k'\Lambda_k\boldsymbol{S}_k$ can be inverted similarly.
- $\boldsymbol{S}'\Lambda\boldsymbol{Y}$ can also be computed recursively.

Fast computation: grouped data



$$\boldsymbol{Y}_t = \begin{pmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{C,t} \\ Y_{X,t} \\ Y_{Y,t} \\ Y_{Z,t} \\ Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} Y_{AX,t} \\ Y_{AY,t} \\ Y_{AZ,t} \\ Y_{BX,t} \\ Y_{BY,t} \\ Y_{BZ,t} \\ Y_{CX,t} \\ Y_{CY,t} \\ Y_{CZ,t} \end{pmatrix}}_{\boldsymbol{B}_t}$$

$$\boldsymbol{Y}_t = \boldsymbol{S}\boldsymbol{B}_t$$

Fast computation: grouped data

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

m = number of rows
n = number of columns

$$\boldsymbol{S}'\Lambda\boldsymbol{S} = \lambda_{00}\boldsymbol{J}_{mn} + (\Lambda_R \otimes \boldsymbol{J}_n) + (\boldsymbol{J}_m \otimes \Lambda_C) + \Lambda_U$$

- Λ_R, Λ_C and Λ_U are diagonal matrices corresponding to rows, columns and unaggregated series;
- λ_{00} corresponds to aggregate.

Fast computation: grouped data

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

$m =$ number of rows
$n =$ number of columns

$$\boldsymbol{S}'\Lambda\boldsymbol{S} = \lambda_{00}\boldsymbol{J}_{mn} + (\Lambda_R \otimes \boldsymbol{J}_n) + (\boldsymbol{J}_m \otimes \Lambda_C) + \Lambda_U$$

- Λ_R, Λ_C and Λ_U are diagonal matrices corresponding to rows, columns and unaggregated series;
- λ_{00} corresponds to aggregate.

Fast computation: grouped data

$$(\boldsymbol{S}\Lambda\boldsymbol{S})^{-1} = \boldsymbol{A} - \frac{\boldsymbol{A}\mathbf{1}_{mn}\mathbf{1}_{mn}'\boldsymbol{A}}{1/\lambda_{00} + \mathbf{1}_{mn}'\boldsymbol{A}\mathbf{1}_{mn}}$$

$\boldsymbol{A} = \Lambda_U^{-1} - \Lambda_U^{-1}(\boldsymbol{J}_m \otimes \boldsymbol{D})\Lambda_U^{-1} - \boldsymbol{E}\boldsymbol{M}^{-1}\boldsymbol{E}'$.

$\boldsymbol{D}$ is diagonal with elements $d_j = \lambda_{0j}/(1 + \lambda_{0j}\sum_i \lambda_{ij}^{-1})$.

$\boldsymbol{E}$ has $m \times m$ blocks where $\boldsymbol{e}_{ij}$ has kth element

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

$\boldsymbol{M}$ is $m \times m$ with (i,j) element

$$(\boldsymbol{M})_{ij} = \begin{cases} 1 + \lambda_{i0}\sum_k \lambda_{ik}^{-1} - \lambda_{i0}\sum_k \lambda_{ik}^{-2}d_k, & i = j, \\ -\lambda_{i0}^{1/2}\lambda_{j0}^{1/2}\sum_k \lambda_{ik}^{-1}\lambda_{jk}^{-1}d_k, & i \neq j. \end{cases}$$

$$(\boldsymbol{S}\Lambda\boldsymbol{S})^{-1} = \boldsymbol{A} - \frac{\boldsymbol{A}\boldsymbol{1}_{mn}\boldsymbol{1}_{mn}'\boldsymbol{A}}{1/\lambda_{00} + \boldsymbol{1}_{mn}'\boldsymbol{A}\boldsymbol{1}_{mn}}$$

$\boldsymbol{A} = \Lambda_U^{-1} - \Lambda_U^{-1}(\boldsymbol{J}_m \otimes \boldsymbol{D})\Lambda_U^{-1} - \boldsymbol{E}\boldsymbol{M}^{-1}\boldsymbol{E}'$.

$\boldsymbol{D}$ is diagonal with elements $d_j = \lambda_{0j}/(1 + \lambda_{0j}\sum_i \lambda_{ij}^{-1})$.

$\boldsymbol{E}$ has $m \times m$ blocks where $\boldsymbol{e}_{ij}$ has kth element

Again, the calculations can be done in such a way that we never store any of the large matrices involved.

$\boldsymbol{M}$ is $m \times m$ with (i,j) element

$$(\boldsymbol{M})_{ij} = \begin{cases} 1 + \lambda_{i0}\sum_k \lambda_{ik}^{-1} - \lambda_{i0}\sum_k \lambda_{ik}^{-2}d_k, & i = j, \\ -\lambda_{i0}^{1/2}\lambda_{j0}^{1/2}\sum_k \lambda_{ik}^{-1}\lambda_{jk}^{-1}d_k, & i \neq j. \end{cases}$$

Fast computation

When the time series are not strictly hierarchical and have more than two grouping variables:

- Use sparse matrix storage and arithmetic.
- Use iterative approximation for inverting large sparse matrices.

Paige & Saunders (1982)
ACM Trans. Math. Software

Fast computation

When the time series are not strictly hierarchical and have more than two grouping variables:

- Use sparse matrix storage and arithmetic.
- Use iterative approximation for inverting large sparse matrices.

Paige & Saunders (1982)
ACM Trans. Math. Software

Fast computation

When the time series are not strictly hierarchical and have more than two grouping variables:

- Use sparse matrix storage and arithmetic.
- Use iterative approximation for inverting large sparse matrices.

Paige & Saunders (1982)
ACM Trans. Math. Software

Outline

1 Optimally reconciled forecasts

2 Fast computation

3 hts package for R

4 References

hts package for R

hts: Hierarchical and grouped time series

Methods for analysing and forecasting hierarchical and grouped time series

Version:	4.3
Depends:	forecast ($\geq$ 5.0)
Imports:	SparseM, parallel, utils
Published:	2014-06-10
Author:	Rob J Hyndman, Earo Wang and Alan Lee
Maintainer:	Rob J Hyndman $<$Rob.Hyndman at monash.edu$>$
BugReports:	`https://github.com/robjhyndman/hts/issues`
License:	GPL ($\geq$ 2)

Example using R

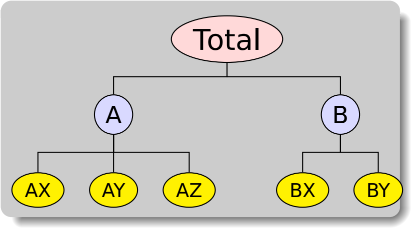
```
library(hts)

# bts is a matrix containing the bottom level time series
# nodes describes the hierarchical structure
y <- hts(bts, nodes=list(2, c(3,2)))
```

Example using R

```
library(hts)

# bts is a matrix containing the bottom level time series
# nodes describes the hierarchical structure
y <- hts(bts, nodes=list(2, c(3,2)))
```



Example using R

```
library(hts)

# bts is a matrix containing the bottom level time series
# nodes describes the hierarchical structure
y <- hts(bts, nodes=list(2, c(3,2)))

# Forecast 10-step-ahead using WLS combination method
# ETS used for each series by default
fc <- forecast(y, h=10)
```

forecast.gts function

Usage

```
forecast(object, h,
  method = c("comb", "bu", "mo", "tdgsf", "tdgsa", "tdfp"),
  fmethod = c("ets", "rw", "arima"),
  weights = c("sd", "none", "nseries"),
  positive = FALSE,
  parallel = FALSE, num.cores = 2, ...)
```

Arguments

`object`	Hierarchical time series object of class `gts`.
`h`	Forecast horizon
`method`	Method for distributing forecasts within the hierarchy.
`fmethod`	Forecasting method to use
`positive`	If TRUE, forecasts are forced to be strictly positive
`weights`	Weights used for "optimal combination" method. When weights = "sd", it takes account of the standard deviation of forecasts.
`parallel`	If TRUE, allow parallel processing
`num.cores`	If parallel = TRUE, specify how many cores are going to be used

Outline

1 Optimally reconciled forecasts

2 Fast computation

3 hts package for R

4 References

References

RJ Hyndman, RA Ahmed, G Athanasopoulos, and HL Shang (Sept. 2011). "Optimal combination forecasts for hierarchical time series". *Computational statistics & data analysis* **55**(9), 2579–2589.

RJ Hyndman, AJ Lee, and E Wang (2014). *Fast computation of reconciled forecasts for hierarchical and grouped time series*. Working paper 17/14. Department of Econometrics & Business Statistics, Monash University

RJ Hyndman, AJ Lee, and E Wang (2014). *hts: Hierarchical and grouped time series*. `cran.r-project.org/package=hts`.

RJ Hyndman and G Athanasopoulos (2014). *Forecasting: principles and practice*. OTexts. `OTexts.org/fpp/`

References

RJ Hyndman, RA Ahmed, G Athanasopoulos, and HL Shang (Sept. 2011). "Optimal combination forecasts for hierarchical time series". *Computational statistics & data analysis* **55**(9), 2579–2589.

RJ Hyndman, AJ Lee, and E Wang (2014). *Fast computation of reconciled forecasts for hierarchical and grouped time series*. Working paper 17/14. Department of Econometrics & Business Statistics, Monash University

- ➥ Papers and R code: **robjhyndman.com**
- ➥ Email: **Rob.Hyndman@monash.edu**