

Forecast reconciliation

1. Hierarchical time series & forecast reconciliation

Rob J Hyndman



MONASH University

Photo by Edvard Alexander Rølvaag on Unsplash

robjhyndman.com/fr2023

Outline

- 1 Hierarchical time series data
- 2 Hierarchical forecasting using single-level approaches
- 3 Linear forecast reconciliation
- 4 Example: Australian tourism
- 5 Fast computational tricks

Outline

- 1 Hierarchical time series data
- 2 Hierarchical forecasting using single-level approaches
- 3 Linear forecast reconciliation
- 4 Example: Australian tourism
- 5 Fast computational tricks

Labour market participation

Australia & New Zealand Standard Classification of Occupations

- 8 major groups
 - ▶ 43 sub-major groups
 - ★ 97 minor groups
 - 359 unit groups
 - 1023 occupations

Labour market participation

Australia & New Zealand Standard Classification of Occupations

- 8 major groups
 - ▶ 43 sub-major groups
 - ★ 97 minor groups
 - 359 unit groups
 - 1023 occupations

Example: statistician

2 Professionals

22 Business, Human Resource and Marketing Professionals

224 Information and Organisation Professionals

2241 Actuaries, Mathematicians and Statisticians

224113 Statistician

PBS sales



PBS sales

ATC drug classification

- A Alimentary tract and metabolism
- B Blood and blood forming organs
- C Cardiovascular system
- D Dermatologicals
- G Genito-urinary system and sex hormones
- H Systemic hormonal preparations, excluding sex hormones and insulins
- J Anti-infectives for systemic use
- L Antineoplastic and immunomodulating agents
- M Musculo-skeletal system
- N Nervous system
- P Antiparasitic products, insecticides and repellents
- R Respiratory system
- S Sensory organs
- V Various

PBS sales

ATC drug classification

14 classes

A

Alimentary tract and metabolism

84 classes

A10

Drugs used in diabetes

A10B

Blood glucose lowering drugs

A10BA

Biguanides

A10BA02

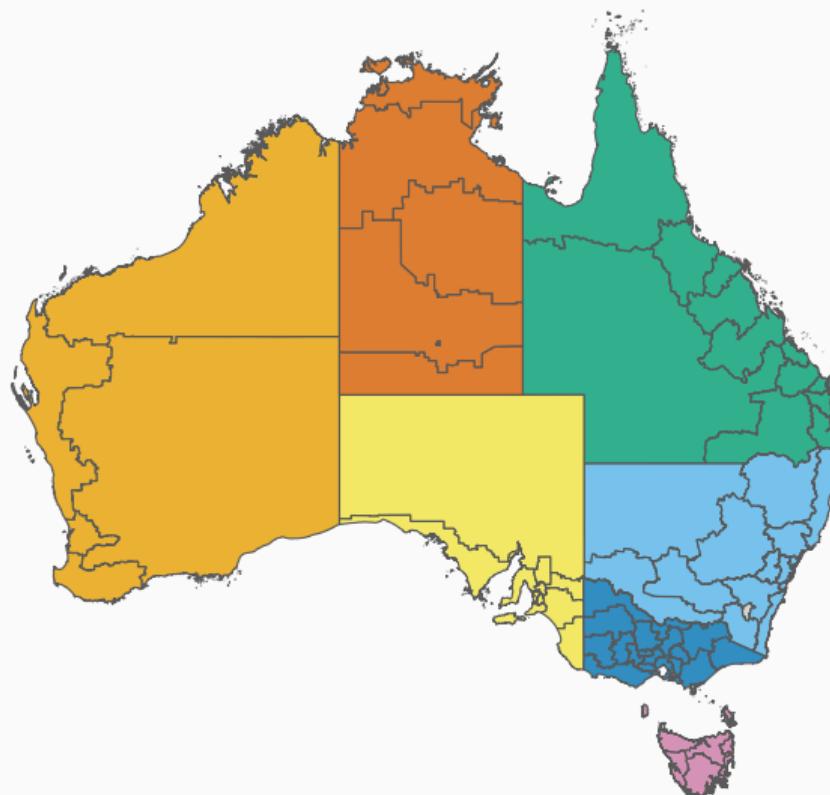
Metformin

Spectacle sales

- Monthly UK sales data from 2000 – 2014
- Provided by a large spectacle manufacturer
- Split by brand (26), gender (3), price range (6), materials (4), and stores (600)
- About 1 million bottom-level series



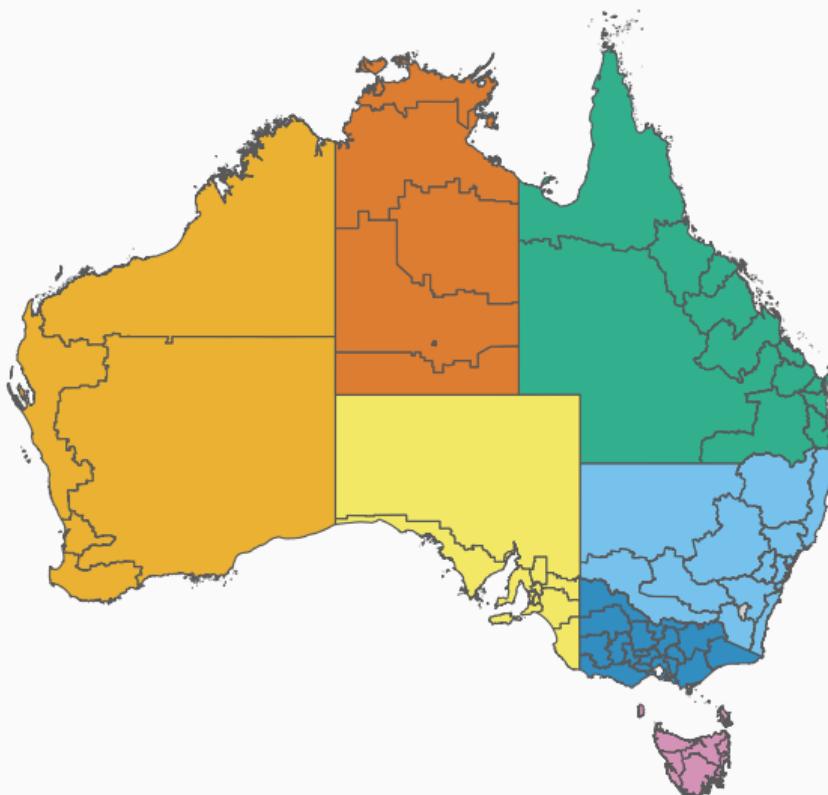
Australian tourism regions



State

- Australian Capital Territory
- New South Wales
- Northern Territory
- Queensland
- South Australia
- Tasmania
- Victoria
- Western Australia

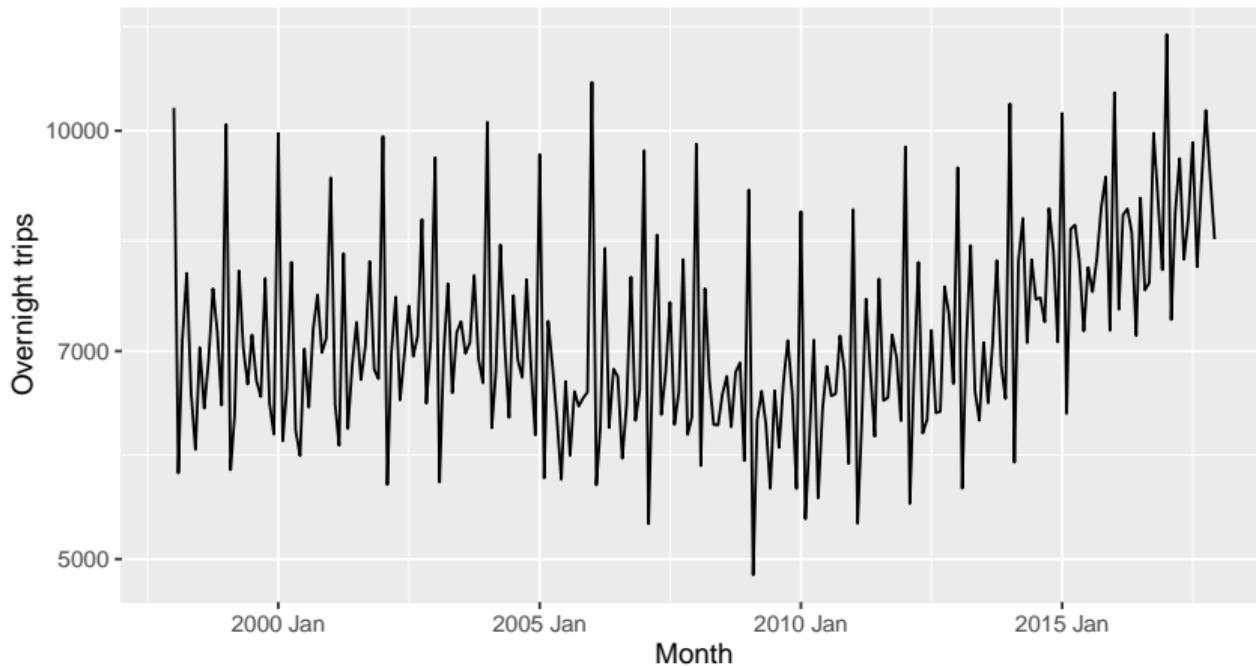
Australian tourism regions



- Monthly data on visitor night from 1998 – 2017
- From *National Visitor Survey*, annual interviews of 120,000 Australians aged 15+.
- Geographical hierarchy split by
 - ▶ 7 states
 - ▶ 27 zones
 - ▶ 75 regions

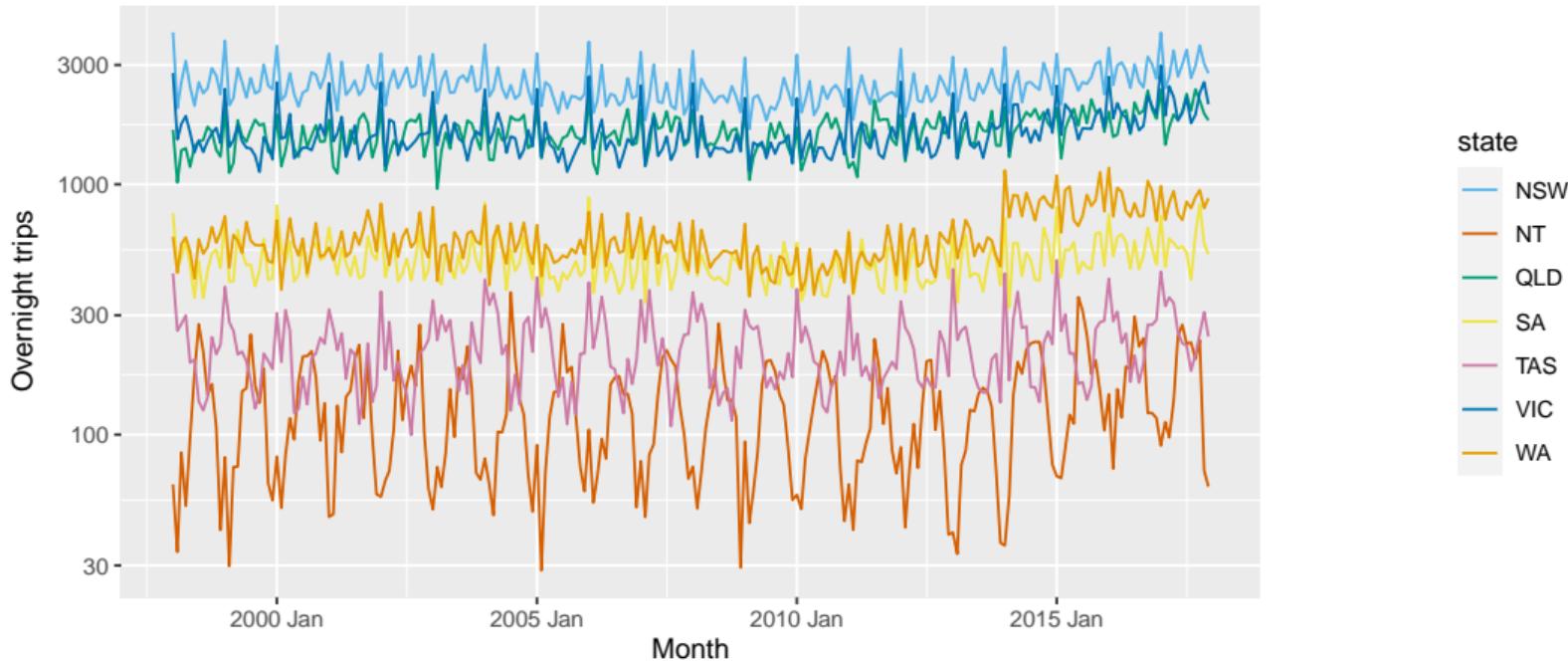
Australian tourism data

Total domestic travel: Australia



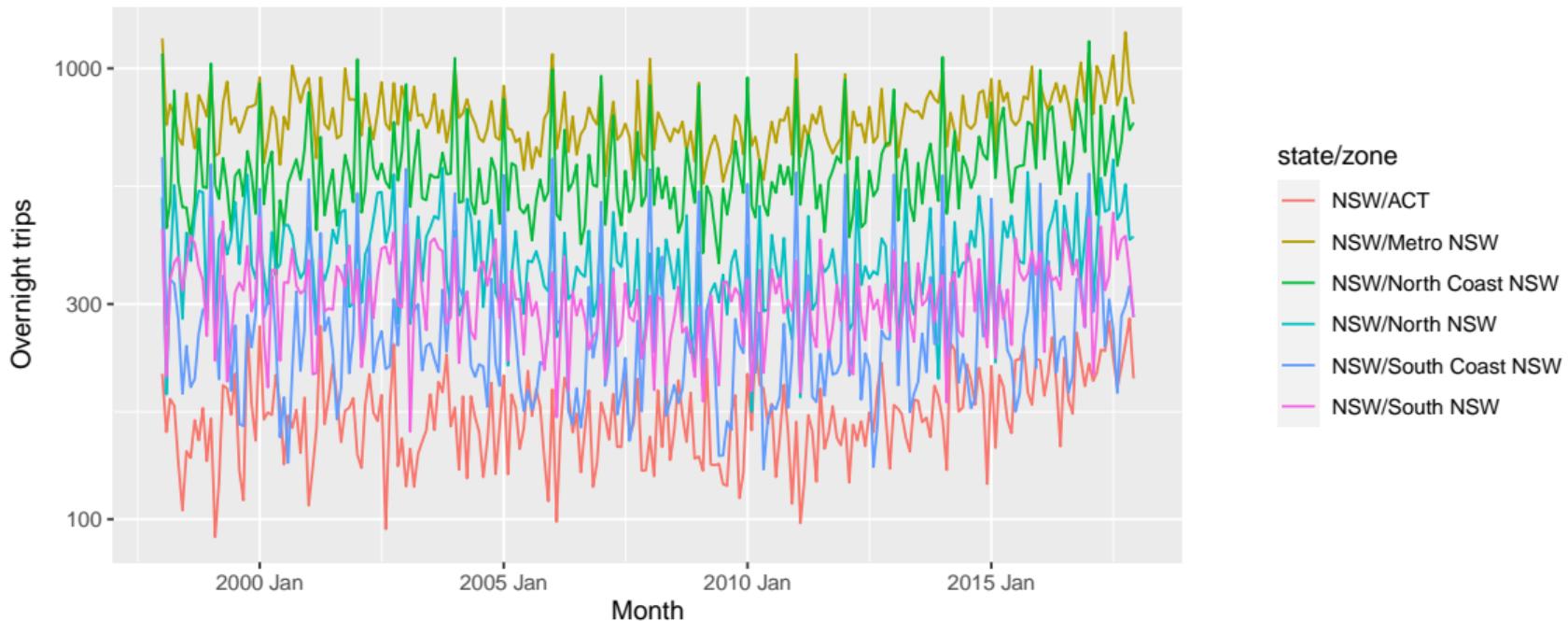
Australian tourism data

Total domestic travel: by state



Australian tourism data

Total domestic travel: NSW by zone



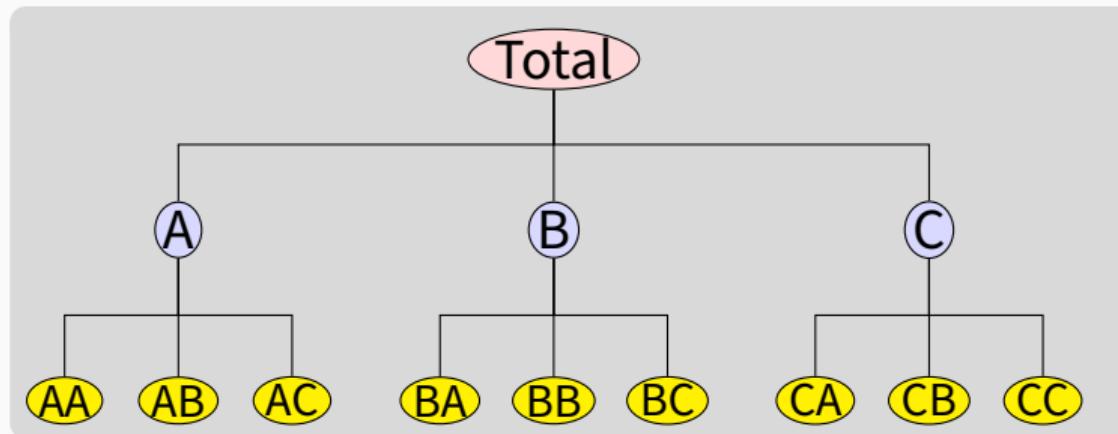
Australian tourism data

Total domestic travel: South NSW by region



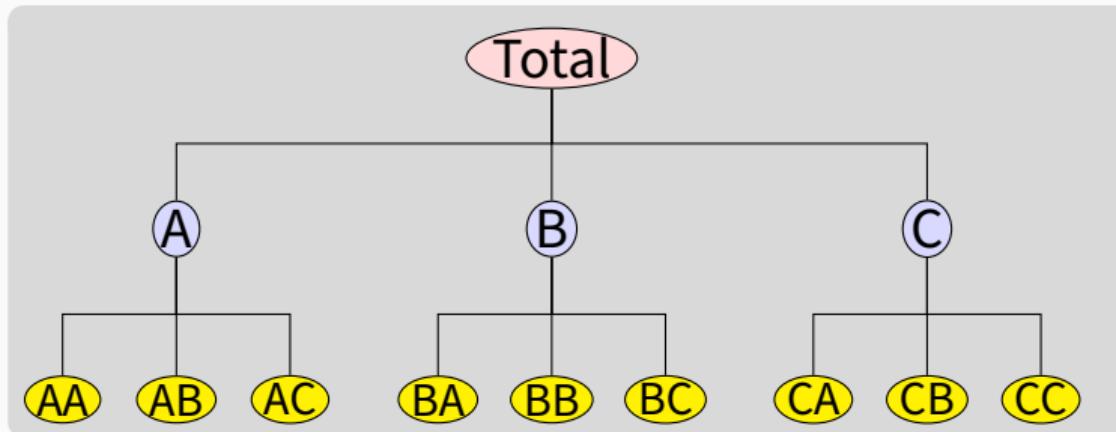
Hierarchical time series

A **hierarchical time series** is a collection of several time series that are linked together in a hierarchical structure.



Hierarchical time series

A **hierarchical time series** is a collection of several time series that are linked together in a hierarchical structure.

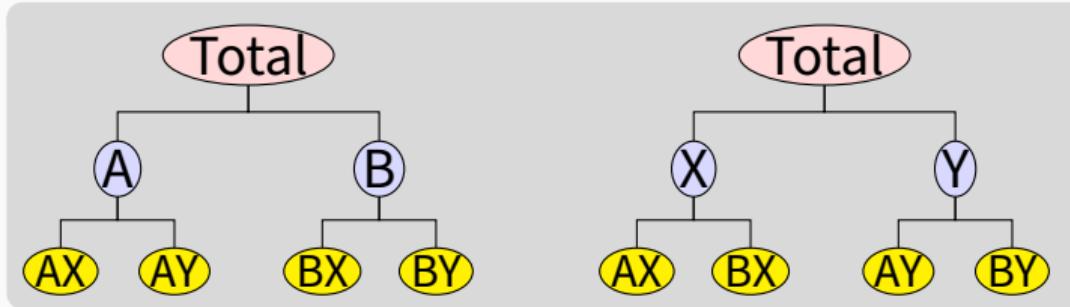


Examples

- Tourism by state and region
- Retail sales by product groups, sub groups, and SKUs

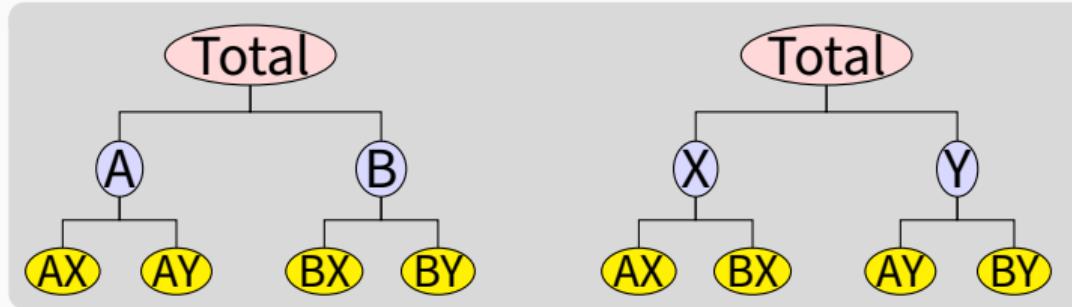
Grouped time series

A **grouped time series** is a collection of time series that can be grouped together in a number of non-hierarchical ways.



Grouped time series

A **grouped time series** is a collection of time series that can be grouped together in a number of non-hierarchical ways.



Examples

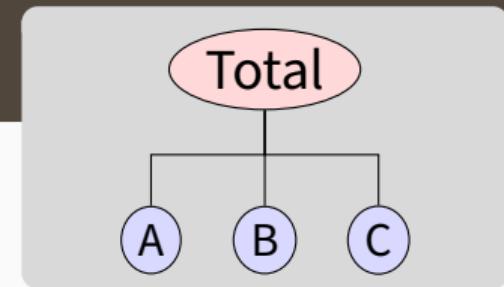
- Tourism by state and purpose of travel
- Retail sales by product groups/sub groups, and by countries/regions

Hierarchical and grouped time series

Every collection of time series with linear constraints can be written as

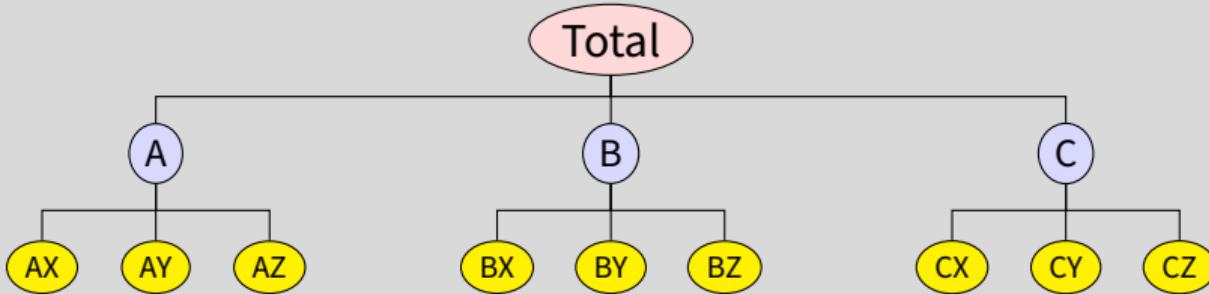
$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t$$

- \mathbf{y}_t = vector of all series at time t
- y_t = aggregate of all series at time t .
- $y_{X,t}$ = value of series X at time t .
- \mathbf{b}_t = vector of most disaggregated series at time t
- \mathbf{S} = “summing matrix” containing the linear constraints.

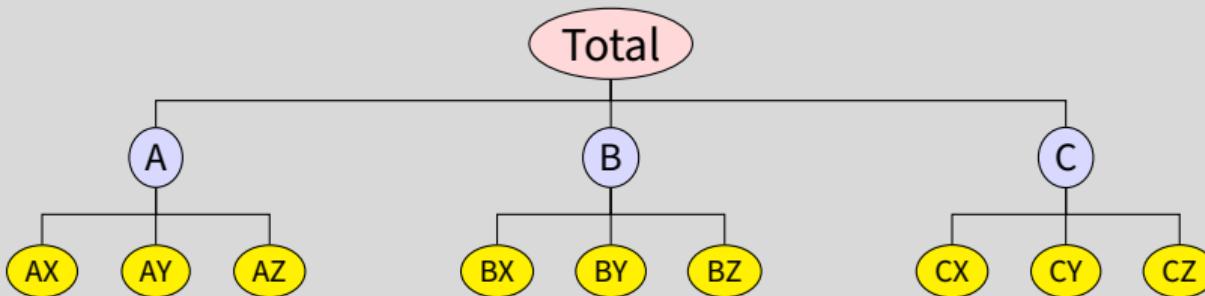


$$\begin{aligned} \mathbf{y}_t &= \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{C,t} \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_S \underbrace{\begin{pmatrix} y_{A,t} \\ y_{B,t} \\ y_{C,t} \end{pmatrix}}_{\mathbf{b}_t} \end{aligned}$$

Hierarchical time series

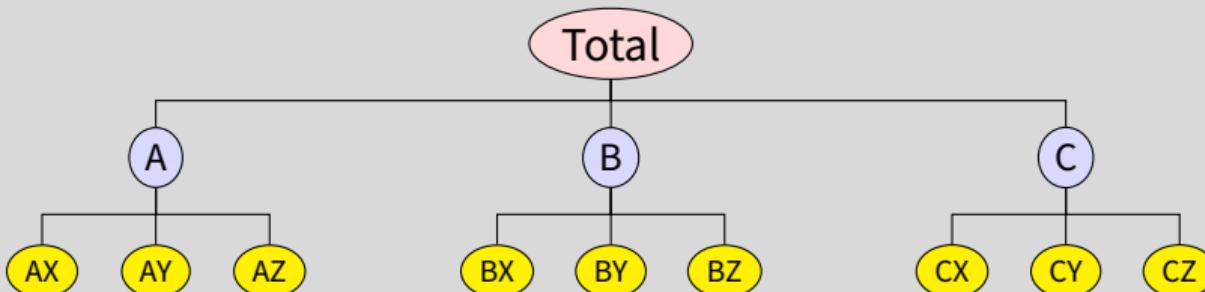


Hierarchical time series



$$\mathbf{y}_t = \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{C,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix}$$

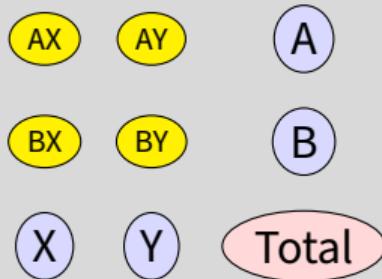
Hierarchical time series



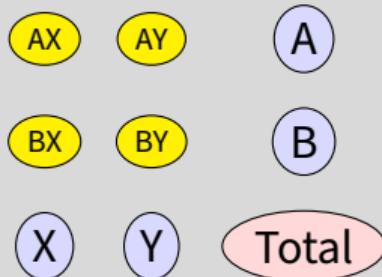
$$\mathbf{y}_t = \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{C,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix}$$

$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t$$

Grouped data

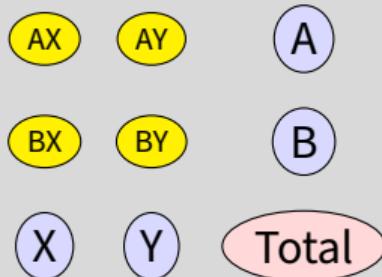


Grouped data



$$\mathbf{y}_t = \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{X,t} \\ y_{Y,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{pmatrix}$$

Grouped data



$$\mathbf{y}_t = \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{X,t} \\ y_{Y,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{BX,t} \\ y_{BY,t} \end{pmatrix}$$

$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t$$

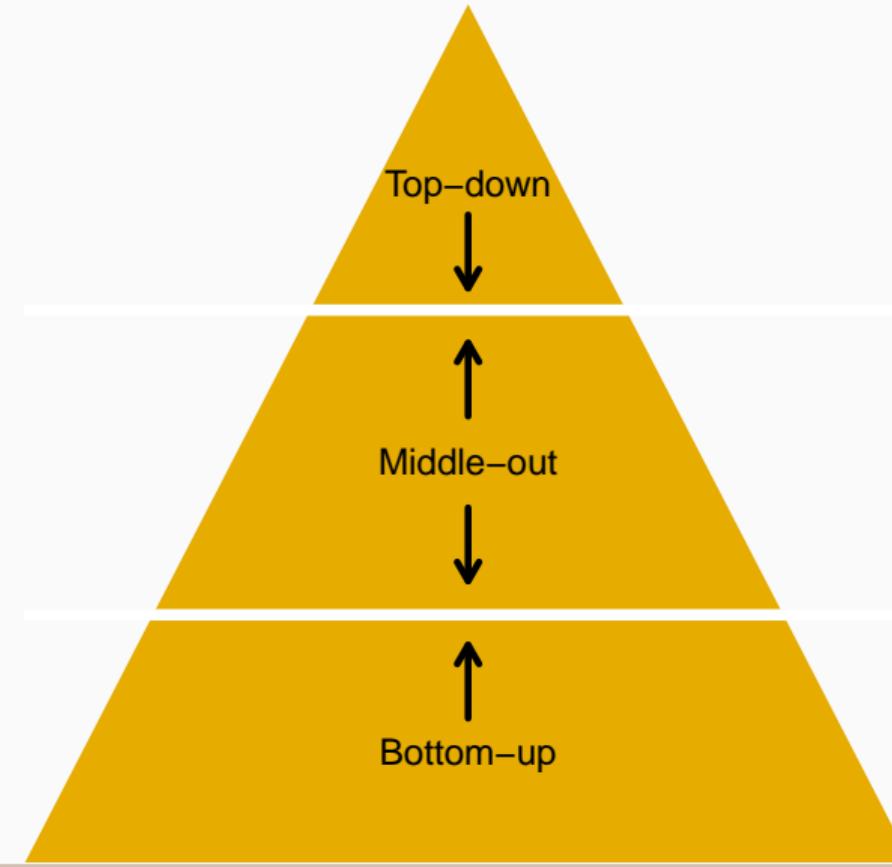
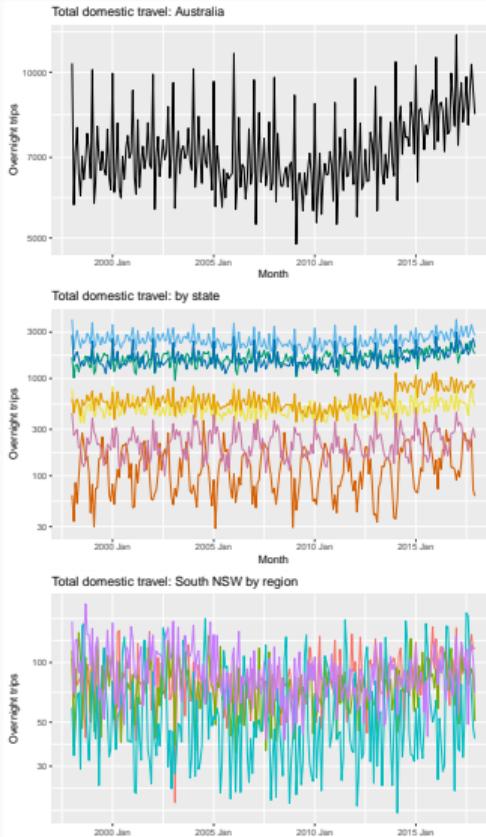
The hierarchical forecasting problem

- We want forecasts at all levels of aggregation.
- If we model and forecast each series independently, the forecasts will almost certainly not add up.
- We need to impose constraints on the forecasts to ensure they are “coherent”.
- We need to do this in a way that is computationally efficient.

Outline

- 1 Hierarchical time series data
- 2 Hierarchical forecasting using single-level approaches
- 3 Linear forecast reconciliation
- 4 Example: Australian tourism
- 5 Fast computational tricks

Hierarchical forecasting 20 years ago



Top-down forecasting

Advantages

- Works well in presence of low counts.
- Single forecasting model easy to build
- Provides reliable forecasts for aggregate levels.

Disadvantages

- Loss of information, especially individual series dynamics.
- Distribution of forecasts to lower levels can be difficult
- No prediction intervals

Bottom-up forecasting

Advantages

- No loss of information.
- Better captures dynamics of individual series.

Disadvantages

- Large number of series to be forecast.
- Constructing forecasting models is harder because of noisy data at bottom level.
- No prediction intervals

Forecasting notation

Let $\hat{\mathbf{y}}_{T+h|T}$ be vector of initial h -step forecasts, made at time T , stacked in same order as \mathbf{y}_t . (In general, they will not “add up”.)

Forecasting notation

Let $\hat{\mathbf{y}}_{T+h|T}$ be vector of initial h -step forecasts, made at time T , stacked in same order as \mathbf{y}_t . (In general, they will not “add up”.)

Coherent linear forecasts are of the form:

$$\tilde{\mathbf{y}}_{T+h|T} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{T+h|T}$$

for some matrix \mathbf{G} .

Forecasting notation

Let $\hat{\mathbf{y}}_{T+h|T}$ be vector of initial h -step forecasts, made at time T , stacked in same order as \mathbf{y}_t . (In general, they will not “add up”.)

Coherent linear forecasts are of the form:

$$\tilde{\mathbf{y}}_{T+h|T} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{T+h|T}$$

for some matrix \mathbf{G} .

- \mathbf{G} extracts and combines base forecasts $\hat{\mathbf{y}}_{T+h|T}$ to get bottom-level forecasts.
- \mathbf{S} adds them up

Bottom-up forecasting

$$\tilde{\mathbf{y}}_{T+h|T} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{T+h|T}$$

Bottom-up forecasts are obtained using

$$\mathbf{G} = [\mathbf{0} \mid \mathbf{I}],$$

where $\mathbf{0}$ is null matrix and \mathbf{I} is identity matrix.

- \mathbf{G} matrix extracts only bottom-level forecasts from $\hat{\mathbf{y}}_{T+h|T}$
- \mathbf{S} adds them up to give the bottom-up forecasts.

Top-down forecasting

$$\tilde{\mathbf{y}}_{T+h|T} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{T+h|T}$$

Top-down forecasts are obtained using

$$\mathbf{G} = [\mathbf{p} \mid \mathbf{0}]$$

where $\mathbf{p} = [p_1, p_2, \dots, p_{n_b}]'$ and $\sum_{k=1}^{n_b} p_k = 1$.

- \mathbf{G} distributes forecasts of aggregate to lowest level series.
- Different methods of top-down forecasting lead to different proportionality vectors \mathbf{p} .

Properties of single-level methods

$$\tilde{\mathbf{y}}_{t+h|t} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{t+h|t}$$

Mean

$$\begin{aligned} E[\tilde{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] &= \mathbf{S}\mathbf{G}E[\hat{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] \\ &= \mathbf{S}E[\mathbf{b}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] \end{aligned}$$

provided $\mathbf{SGS} = \mathbf{S}$ and $E[\hat{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] = \mathbf{SE}[\mathbf{b}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T]$. i.e., forecasts $\tilde{\mathbf{y}}_{T+h|T}$ are unbiased iff base forecasts $\hat{\mathbf{y}}_{T+h|T}$ are unbiased and $\mathbf{SGS} = \mathbf{S}$.

Properties of single-level methods

$$\tilde{\mathbf{y}}_{t+h|t} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{t+h|t}$$

Mean

$$\begin{aligned} E[\tilde{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] &= \mathbf{S}\mathbf{G}E[\hat{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] \\ &= \mathbf{S}E[\mathbf{b}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] \end{aligned}$$

provided $\mathbf{SGS} = \mathbf{S}$ and $E[\hat{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] = \mathbf{SE}[\mathbf{b}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T]$. i.e., forecasts $\tilde{\mathbf{y}}_{T+h|T}$ are unbiased iff base forecasts $\hat{\mathbf{y}}_{T+h|T}$ are unbiased and $\mathbf{SGS} = \mathbf{S}$.

- $\mathbf{SGS} = \mathbf{S}$ for bottom-up method
- $\mathbf{SGS} \neq \mathbf{S}$ for any top-down or middle-out method.

Properties of single-level methods

$$\tilde{\mathbf{y}}_{t+h|t} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{t+h|t}$$

Variance

$$\begin{aligned}\mathbf{V}_h &= \text{Var}[\mathbf{y}_{T+h} - \tilde{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] \\ &= \mathbf{S}\mathbf{G}\mathbf{W}_h\mathbf{G}'\mathbf{S}'\end{aligned}$$

where $\mathbf{W}_h = \text{Var}[\mathbf{y}_{T+h} - \hat{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T]$

Properties of single-level methods

$$\tilde{\mathbf{y}}_{t+h|t} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{t+h|t}$$

Variance

$$\begin{aligned}\mathbf{V}_h &= \text{Var}[\mathbf{y}_{T+h} - \tilde{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] \\ &= \mathbf{S}\mathbf{G}\mathbf{W}_h\mathbf{G}'\mathbf{S}'\end{aligned}$$

where $\mathbf{W}_h = \text{Var}[\mathbf{y}_{T+h} - \hat{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T]$

- \mathbf{W}_h is hard to estimate for $h > 1$.
- This suggests we should choose \mathbf{G} to minimise \mathbf{V}_h .

Outline

- 1 Hierarchical time series data
- 2 Hierarchical forecasting using single-level approaches
- 3 Linear forecast reconciliation
- 4 Example: Australian tourism
- 5 Fast computational tricks

Early history of forecast reconciliation

History

2001: Idea to use all available series to forecast Australia's labour market by occupation.

2004: PhD student Roman Ahmed begins, co-supervised with George Athanasopoulos.

2006: Presentation at ISF, Santander.

2007: Pre-print of "Optimal combination forecasts for hierarchical time series".

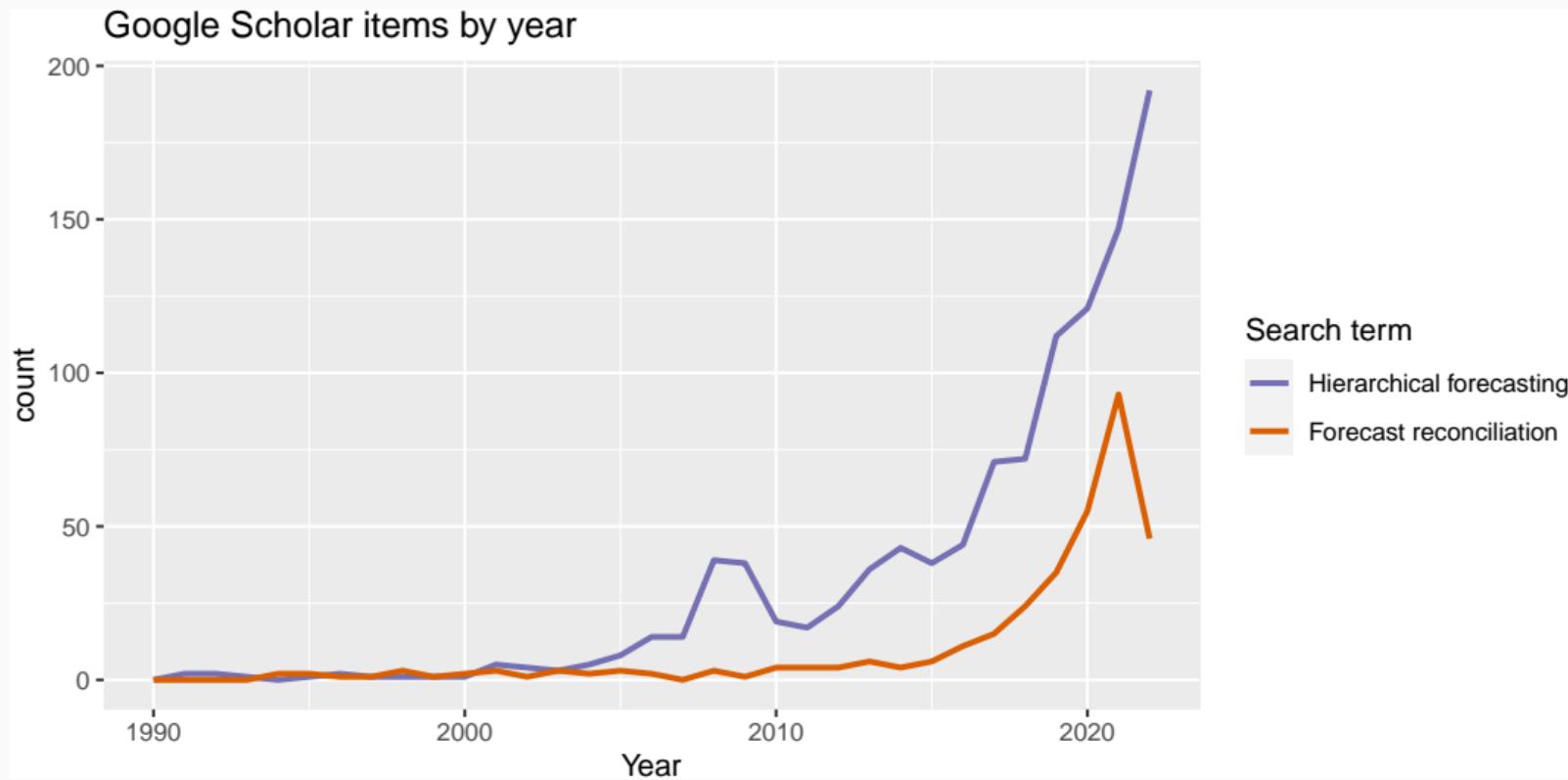
2009: Application to Australian tourism published in IJF.

2010: First version of hts package on CRAN.

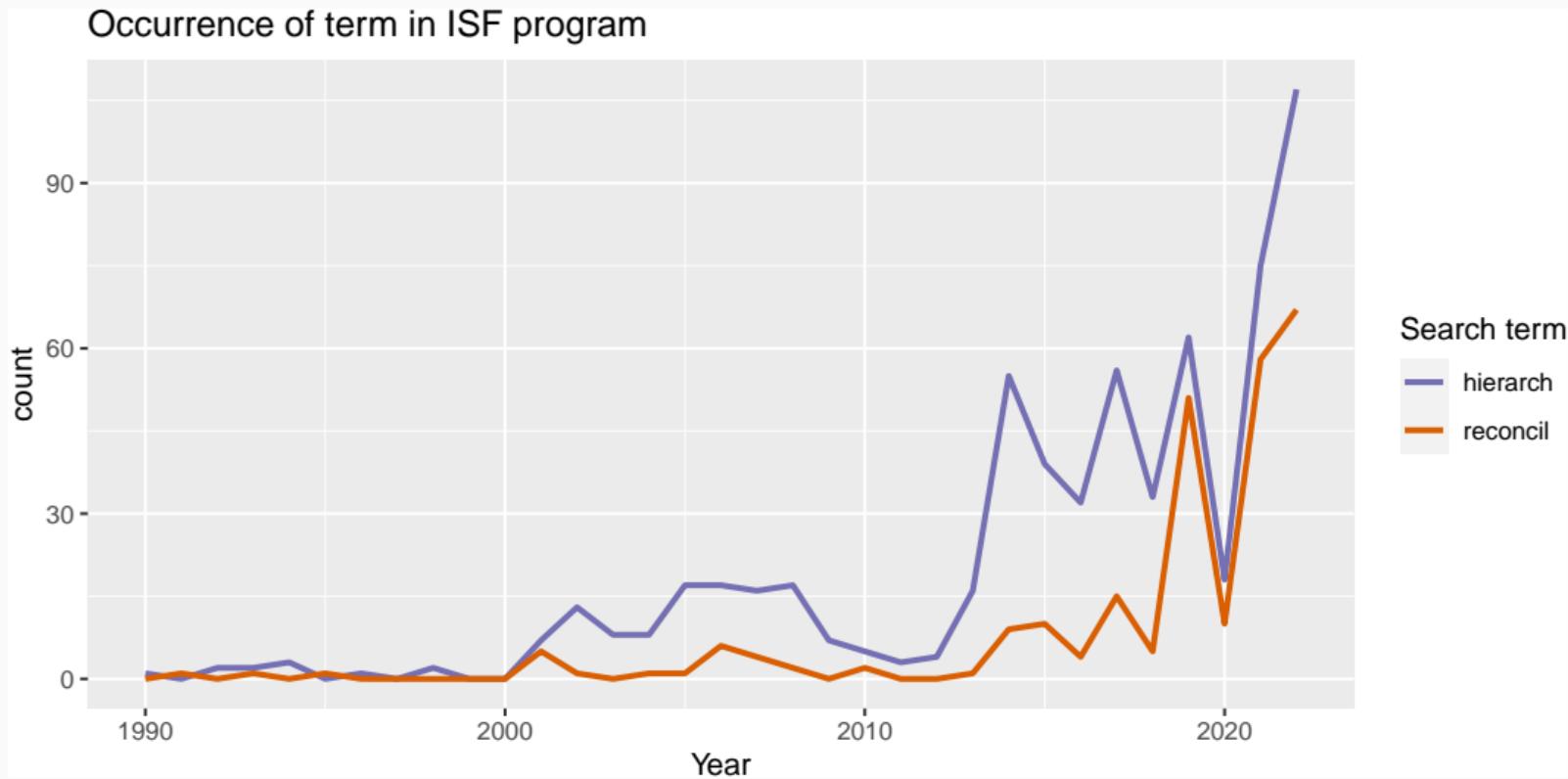
2011: "Optimal combination forecasts for hierarchical time series" appears in CSDA.

2019: "Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization" appears in JASA.

Forecast reconciliation research



Forecast reconciliation research



Linear forecast reconciliation

$$\tilde{\mathbf{y}}_{T+h|T} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{T+h|T}$$

- \mathbf{G} combines base forecasts $\hat{\mathbf{y}}_{T+h|T}$ to get bottom-level forecasts.
- \mathbf{S} creates linear combinations.
- Bottom-up and top-down can be seen as special cases.
- \mathbf{SG} is a projection matrix iff $\mathbf{SGS}' = \mathbf{S}$.
- Reconciled forecasts are unbiased iff base forecasts are unbiased and \mathbf{SG} is a projection.
- $\mathbf{V}_h = \text{Var}[\mathbf{y}_{T+h} - \tilde{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T] = \mathbf{SGW}_h\mathbf{G}'\mathbf{S}'$
- How to choose \mathbf{G} to create optimal forecasts?

Minimum trace reconciliation

Minimum trace (MinT) reconciliation

If $\mathbf{S}\mathbf{G}$ is a projection, then the trace of \mathbf{V}_h is minimized when

$$\mathbf{G} = (\mathbf{S}'\mathbf{W}_h^{-1}\mathbf{S})^{-1}\mathbf{S}'\mathbf{W}_h^{-1}$$

$$\tilde{\mathbf{y}}_{T+h|T} = \mathbf{S}(\mathbf{S}'\mathbf{W}_h^{-1}\mathbf{S})^{-1}\mathbf{S}'\mathbf{W}_h^{-1}\hat{\mathbf{y}}_{T+h|T}$$

Reconciled forecasts

Base forecasts

- Trace of \mathbf{V}_h is sum of forecast variances.
- MinT solution is L_2 optimal amongst linear unbiased forecasts.
- How to estimate $\mathbf{W}_h = \text{Var}[\mathbf{y}_{T+h} - \hat{\mathbf{y}}_{T+h|T} \mid \mathbf{y}_1, \dots, \mathbf{y}_T]$?
- Is $\mathbf{W}_h \approx k_h \mathbf{W}_1$ a reasonable approximation?

Linear projections

$$\tilde{\mathbf{y}}_{T+h|T} = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_{T+h|T}$$

Reconciliation method

\mathbf{G}

OLS $(\mathbf{S}'\mathbf{S})^{-1}\mathbf{S}'$

WLS(var) $(\mathbf{S}'\Lambda_v\mathbf{S})^{-1}\mathbf{S}'\Lambda_v$

WLS(struct) $(\mathbf{S}'\Lambda_s\mathbf{S})^{-1}\mathbf{S}'\Lambda_s$

MinT(sample) $(\mathbf{S}'\hat{\mathbf{W}}_{\text{sam}}^{-1}\mathbf{S})^{-1}\mathbf{S}'\hat{\mathbf{W}}_{\text{sam}}^{-1}$

MinT(shrink) $(\mathbf{S}'\hat{\mathbf{W}}_{\text{shr}}^{-1}\mathbf{S})^{-1}\mathbf{S}'\hat{\mathbf{W}}_{\text{shr}}^{-1}$

These approximate
MinT by assuming
 $\mathbf{W}_h = k_h \mathbf{W}_1$.

- $\Lambda_v = \text{diag}(\mathbf{W}_1)^{-1}$ ■ $\Lambda_s = \text{diag}(\mathbf{S}\mathbf{1})^{-1}$
- $\hat{\mathbf{W}}_{\text{sam}}$ is sample estimate of the residual covariance matrix
- $\hat{\mathbf{W}}_{\text{shr}}$ is shrinkage estimator $\tau \text{diag}(\hat{\mathbf{W}}_{\text{sam}}) + (1 - \tau)\hat{\mathbf{W}}_{\text{sam}}$
where τ selected optimally.
- Still need a good estimate of \mathbf{W}_h for forecast variance.

OLS is not as bad as you might think

- If all base forecasts come from the same model, then forecast errors are coherent.
- So $\hat{\mathbf{y}}_{T+h|T} = \mathbf{S}\hat{\mathbf{b}}_{T+h|T} + \mathbf{S}\hat{\boldsymbol{\varepsilon}}_{T+h}$ and

$$\mathbf{W}_h = \text{Var}(\mathbf{y}_{T+h} - \hat{\mathbf{y}}_{T+h|T}) = \mathbf{S}\Sigma_h\mathbf{S}'$$

where $\Sigma_h = \text{Var}(\hat{\boldsymbol{\varepsilon}}_{T+h})$. So

$$\begin{aligned}\mathbf{G} &= [\mathbf{S}'\mathbf{W}_h^{-1}\mathbf{S}]^{-1}\mathbf{S}'\mathbf{W}_h^{-1} \\ &= [\mathbf{S}'(\mathbf{S}\Sigma_h\mathbf{S}')^{-1}\mathbf{S}]^{-1}\mathbf{S}'(\mathbf{S}\Sigma_h\mathbf{S}')^{-1} \\ &= (\mathbf{S}'\mathbf{S})^{-1}\mathbf{S}'\end{aligned}$$

Outline

- 1 Hierarchical time series data
- 2 Hierarchical forecasting using single-level approaches
- 3 Linear forecast reconciliation
- 4 Example: Australian tourism
- 5 Fast computational tricks

Example: Australian tourism

tourism

```
# A tsibble: 18,000 x 5 [1M]
# Key:      state, zone, region [75]
  month state zone      region visitors
  <mth> <chr> <chr>      <chr>     <dbl>
1 1998 Jan NSW Metro NSW Sydney     926.
2 1998 Feb NSW Metro NSW Sydney     647.
3 1998 Mar NSW Metro NSW Sydney     716.
4 1998 Apr NSW Metro NSW Sydney     621.
5 1998 May NSW Metro NSW Sydney     598.
6 1998 Jun NSW Metro NSW Sydney     601.
7 1998 Jul NSW Metro NSW Sydney     720.
8 1998 Aug NSW Metro NSW Sydney     645.
9 1998 Sep NSW Metro NSW Sydney     633.
10 1998 Oct NSW Metro NSW Sydney    771.
# i 17,990 more rows
```

Example: Australian tourism

```
tourism_agg <- tourism |>  
  aggregate_key(state / zone / region, visitors = sum(visitors))
```

```
# A tsibble: 26,400 x 5 [1M]  
# Key:      state, zone, region [110]  
  month state       zone       region     visitors  
  <mth> <chr*>    <chr*>    <chr*>     <dbl>  
1 1998 Jan <aggregated> <aggregated> <aggregated> 10376.  
2 1998 Feb <aggregated> <aggregated> <aggregated> 5746.  
3 1998 Mar <aggregated> <aggregated> <aggregated> 7129.  
4 1998 Apr <aggregated> <aggregated> <aggregated> 7939.  
5 1998 May <aggregated> <aggregated> <aggregated> 6552.  
6 1998 Jun <aggregated> <aggregated> <aggregated> 5969.  
7 1998 Jul <aggregated> <aggregated> <aggregated> 7041.  
8 1998 Aug <aggregated> <aggregated> <aggregated> 6382.  
9 1998 Sep <aggregated> <aggregated> <aggregated> 6907.  
10 1998 Oct <aggregated> <aggregated> <aggregated> 7744.
```

Example: Australian tourism

```
fit <- tourism_agg |>  
  filter(year(month) <= 2015) |>  
  model(ets = ETS(visitors))
```

```
# A mable: 110 x 4  
# Key: state, zone, region [110]  
  
  state    zone           region          ets  
  <chr*> <chr*>        <chr*>        <model>  
1 NSW      ACT            Canberra       <ETS(M,N,A)>  
2 NSW      ACT            <aggregated> <ETS(M,N,A)>  
3 NSW      Metro NSW      Central Coast <ETS(M,N,M)>  
4 NSW      Metro NSW      Sydney         <ETS(M,N,A)>  
5 NSW      Metro NSW      <aggregated> <ETS(M,N,A)>  
6 NSW      North Coast NSW Hunter       <ETS(M,N,M)>  
7 NSW      North Coast NSW North Coast NSW <ETS(M,N,M)>  
8 NSW      North Coast NSW <aggregated> <ETS(M,N,M)>  
9 NSW      North NSW       Blue Mountains <ETS(M,N,A)>
```

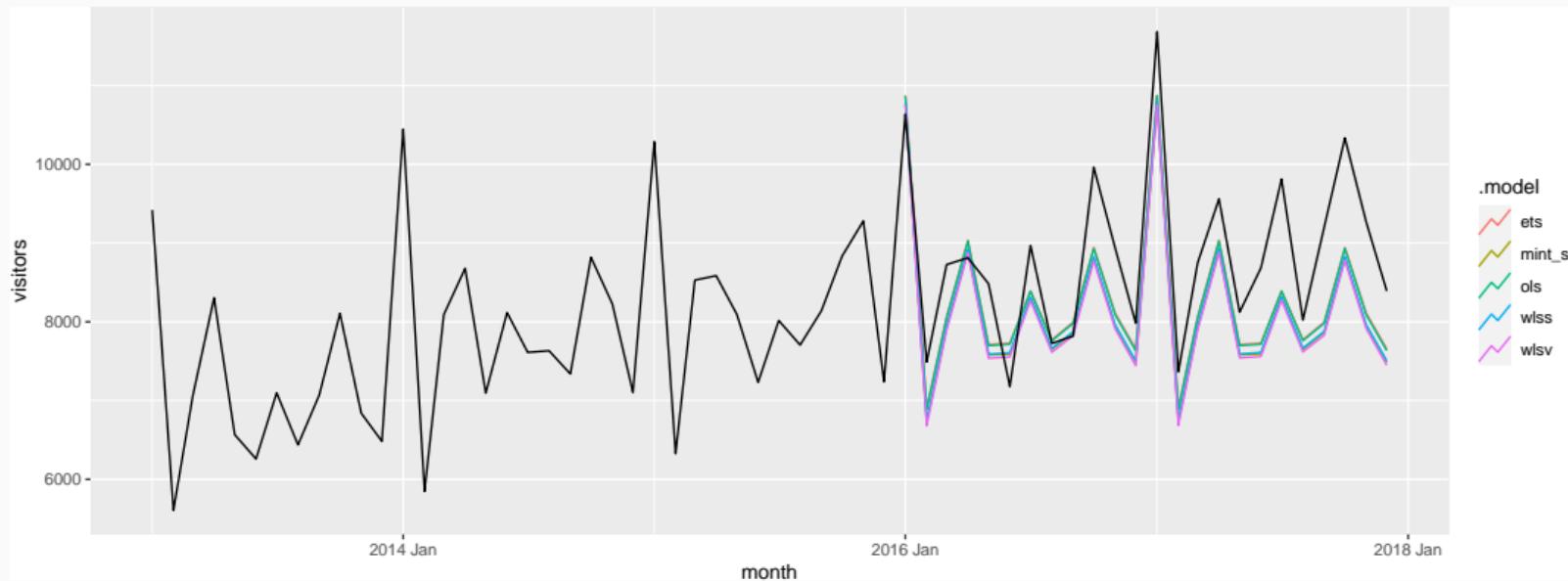
Example: Australian tourism

```
fc <- fit |>
  reconcile(
    ols = min_trace(ets, method = "ols"),
    wlsv = min_trace(ets, method = "wls_var"),
    wlss = min_trace(ets, method = "wls_struct"),
    # mint_c = min_trace(ets, method="mint_cov"),
    mint_s = min_trace(ets, method = "mint_shrink"),
  ) |>
  forecast(h = "2 years")
```

```
# A fable: 13,200 x 7 [1M]
# Key:      state, zone, region, .model [550]
  state   zone   region     .model     month     visitors .mean
  <chr*> <chr*> <chr*>     <chr>     <mth>       <dist> <dbl>
1 NSW     ACT     Canberra  ets      2016 Jan N(202, 1437) 202.
2 NSW     ACT     Canberra  ets      2016 Feb N(160, 912) 160.
3 NSW     ACT     Canberra  ets      2016 Mar N(204, 1489) 204.
```

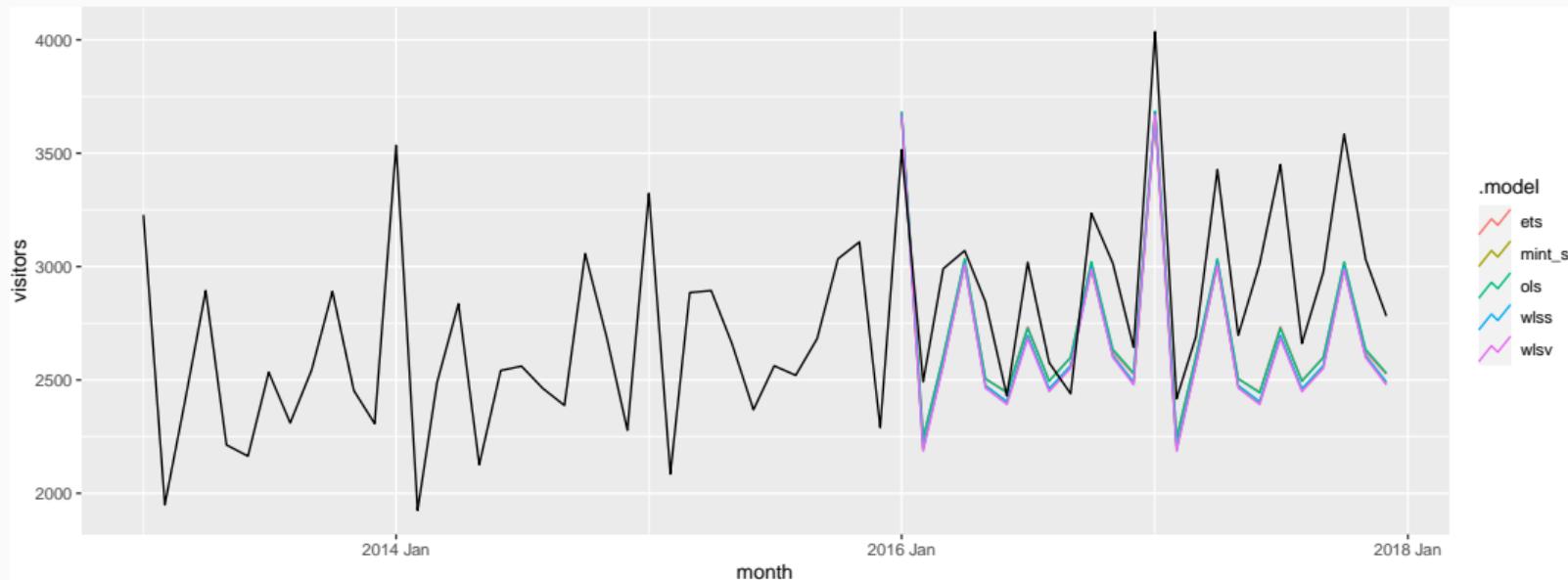
Example: Australian tourism

```
fc |>  
  filter(is_aggregated(state)) |>  
  autoplot(filter(tourism_agg, year(month) > 2012), level = NULL)
```



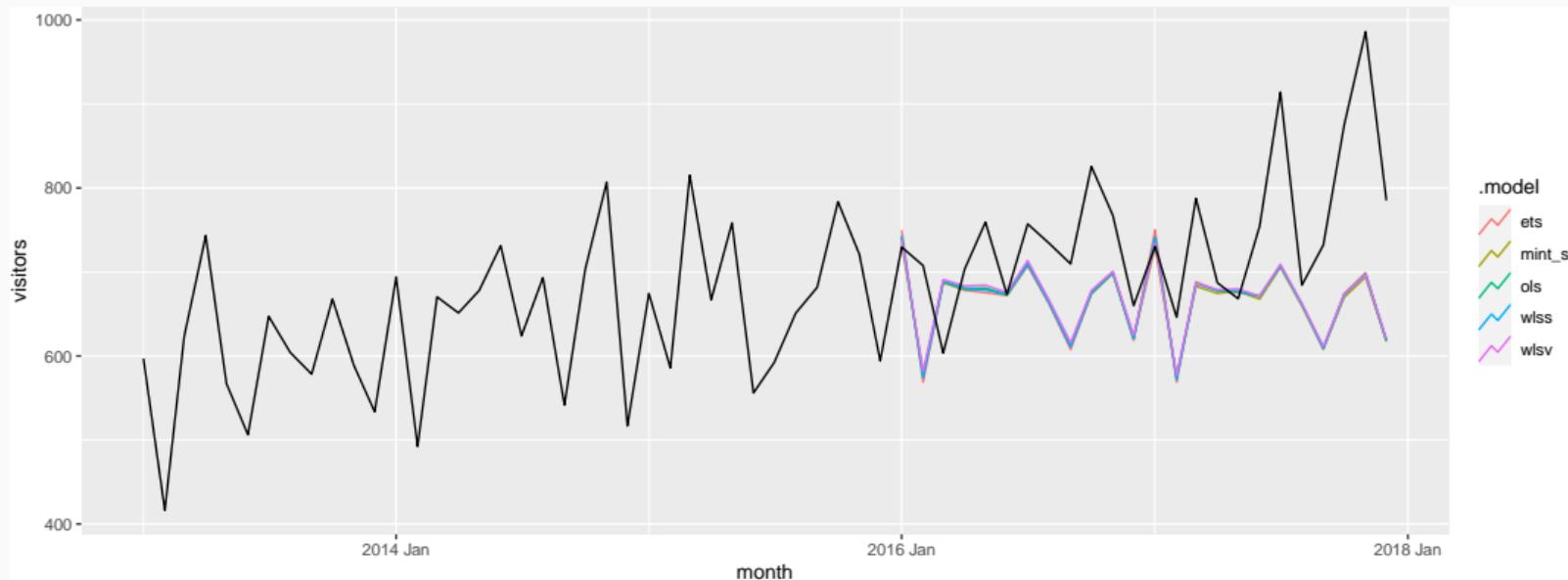
Example: Australian tourism

```
fc |>  
  filter(state == "NSW" & is_aggregated(zone)) |>  
  autoplot(filter(tourism_agg, year(month) > 2012), level = NULL)
```



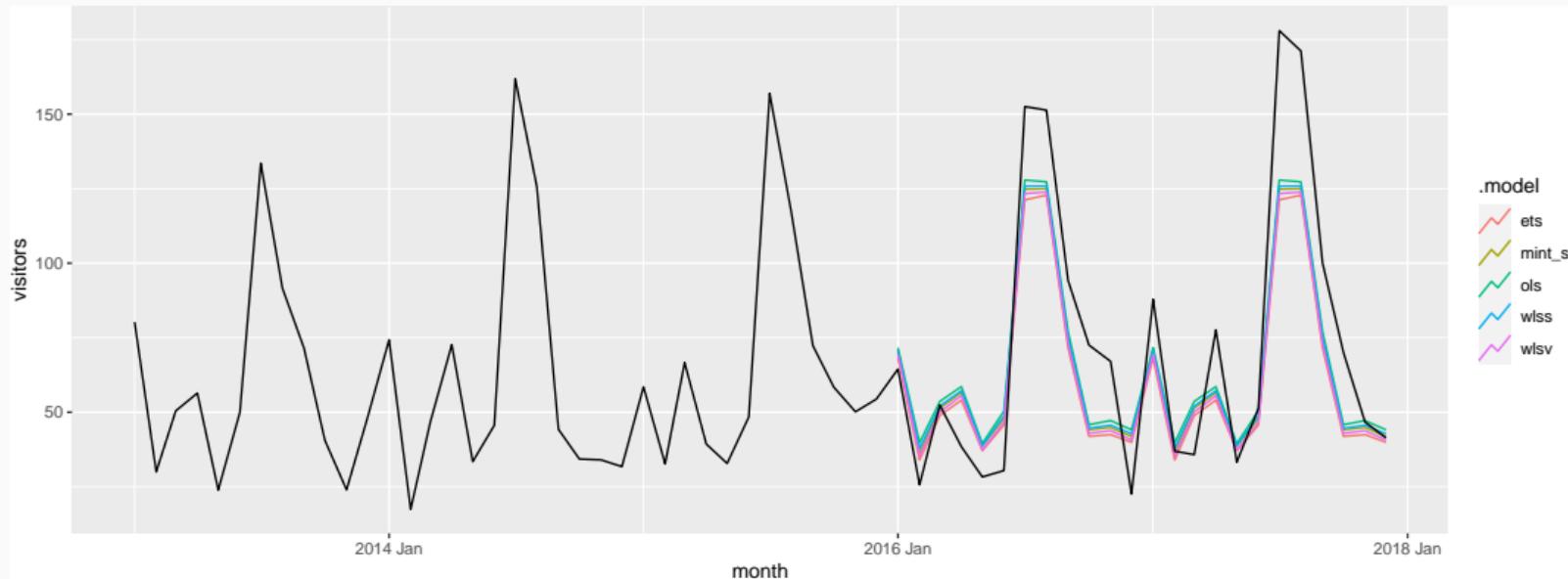
Example: Australian tourism

```
fc |>  
  filter(region == "Melbourne") |>  
  autoplot(filter(tourism_agg, year(month) > 2012), level = NULL)
```



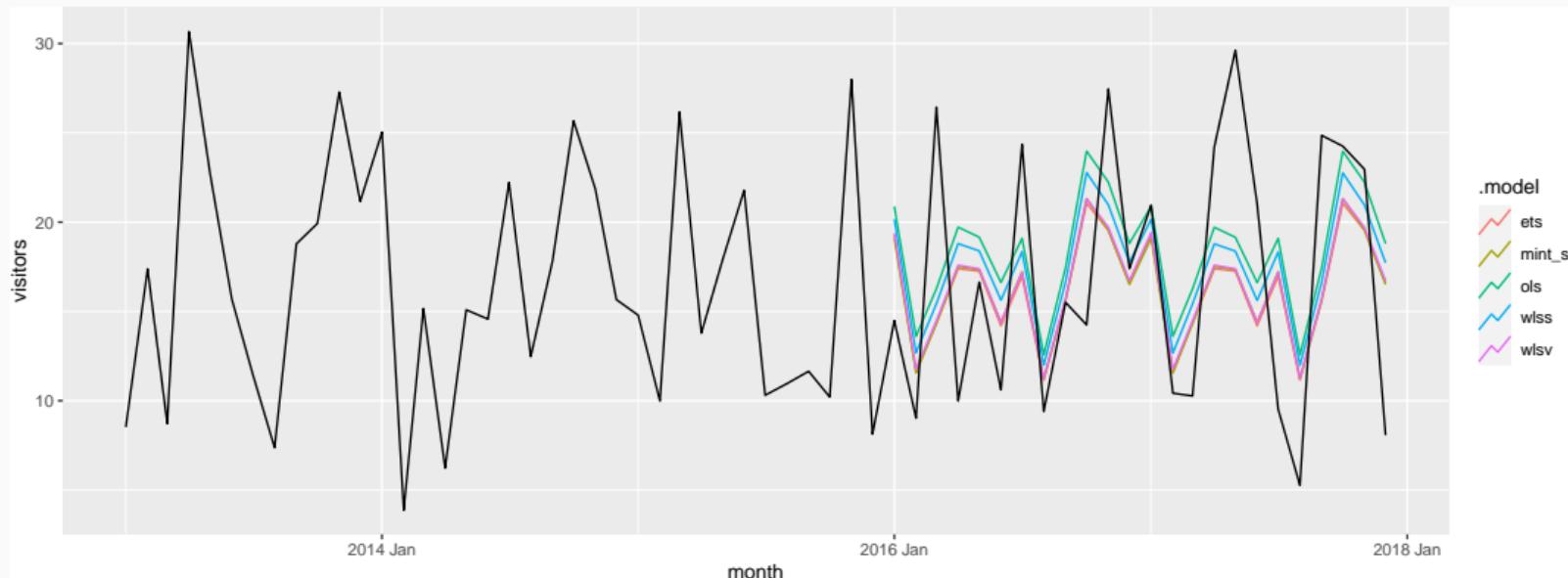
Example: Australian tourism

```
fc |>  
  filter(region == "Snowy Mountains") |>  
  autoplot(filter(tourism_agg, year(month) > 2012), level = NULL)
```



Example: Australian tourism

```
fc |>  
  filter(region == "Barossa") |>  
  autoplot(filter(tourism_agg, year(month) > 2012), level = NULL)
```



Example: Australian tourism

```
fc |>  
accuracy(  
  data = tourism_agg,  
  measures = list(rmsse = RMSSE)  
)
```

```
# A tibble: 550 x 6  
  .model state zone          region      .type rmsse  
  <chr>   <chr> <chr>        <chr>      <chr> <dbl>  
1 ets     NSW    ACT         Canberra    Test    0.835  
2 ets     NSW    ACT         <aggregated> Test    0.835  
3 ets     NSW    Metro NSW Central Coast Test    0.747  
4 ets     NSW    Metro NSW Sydney      Test    1.16  
5 ets     NSW    Metro NSW <aggregated> Test    1.18  
6 ets     NSW    North Coast NSW Hunter    Test    1.21  
7 ets     NSW    North Coast NSW North Coast NSW Test    0.884  
8 ets     NSW    North Coast NSW <aggregated> Test    1.02
```

Example: Australian tourism

```
fc |>
  accuracy(tourism_agg,
    measures = list(mase = MASE, rmsse = RMSSE)
  ) |>
  group_by(.model) |>
  summarise(mase = mean(mase), rmsse = sqrt(mean(rmsse^2))) |>
  arrange(rmsse)
```

```
# A tibble: 5 x 3
  .model   mase   rmsse
  <chr>   <dbl>   <dbl>
1 ols     0.930  0.926
2 wlss    0.949  0.948
3 mint_s  0.953  0.954
4 wlsv    0.964  0.965
5 ets     0.968  0.968
```

Example: Australian tourism

```
fc |>  
  accuracy(tourism_agg,  
    measures = list(mase = MASE, rmsse = RMSSE)  
) |>  
  group_by(.model) |>  
  summarise(mase = mean(mase), rmsse = sqrt(mean(rmsse^2))) |>  
  arrange(rmsse)
```

```
# A tibble: 5 x 3  
  .model    mase   rmsse  
  <chr>    <dbl>  <dbl>  
1 ols      0.930  0.926  
2 wlss     0.949  0.948  
3 mint_s   0.953  0.954  
4 wlsv     0.964  0.965  
5 ets      0.968  0.968
```

- Overall, every reconciliation method is better than the base ETS forecasts.

Example: Australian tourism

```
# A tibble: 20 x 4
# Groups:   .model [5]
  .model level      mase rmsse
  <chr>  <fct>    <dbl> <dbl>
1 ets     National  1.44  1.27
2 ols     National  1.46  1.29
3 wlss    National  1.61  1.43
4 mint_s  National  1.64  1.45
5 wlsv    National  1.69  1.49
6 ols     State     1.07  1.08
7 ets     State     1.10  1.11
8 wlss    State     1.13  1.14
9 mint_s  State     1.15  1.15
10 wlsv   State     1.18  1.17
11 ols     Zone      0.954 0.948
12 wlss   Zone      0.987 0.980
13 mint_s Zone      0.995 0.988
14 ets     Zone      1.01  0.999
15 wlsv   Zone      1.01  1.00
16 ols     Region    0.901 0.895
17 wlss   Region    0.910 0.907
18 mint_s Region    0.911 0.911
19 wlsv   Region    0.917 0.919
20 ets     Region    0.935 0.938
```

Example: Australian tourism

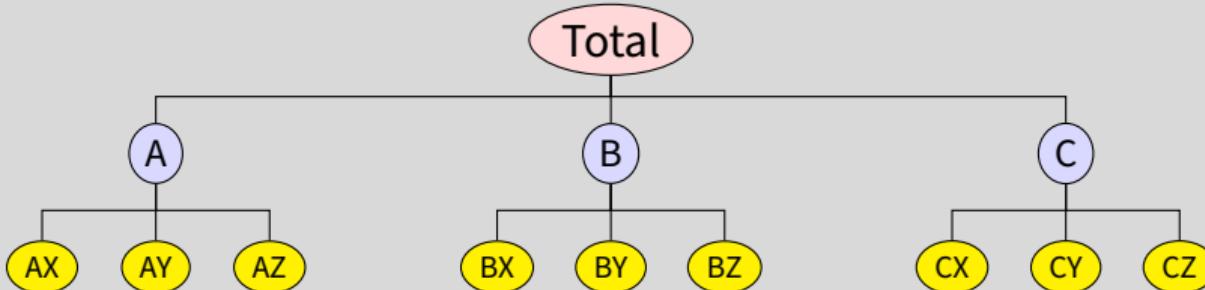
```
# A tibble: 20 x 4
# Groups:   .model [5]
  .model level      mase rmsse
  <chr>  <fct>    <dbl> <dbl>
1 ets     National  1.44  1.27
2 ols     National  1.46  1.29
3 wlss    National  1.61  1.43
4 mint_s  National  1.64  1.45
5 wlsv    National  1.69  1.49
6 ols     State     1.07  1.08
7 ets     State     1.10  1.11
8 wlss    State     1.13  1.14
9 mint_s  State     1.15  1.15
10 wlsv   State     1.18  1.17
11 ols     Zone      0.954 0.948
12 wlss   Zone      0.987 0.980
13 mint_s Zone      0.995 0.988
14 ets     Zone      1.01  0.999
15 wlsv   Zone      1.01  1.00
16 ols     Region    0.901 0.895
17 wlss   Region    0.910 0.907
18 mint_s Region    0.911 0.911
19 wlsv   Region    0.917 0.919
20 ets     Region    0.935 0.938
```

- OLS is best for all levels except national.
- Improvements due to reconciliation are greater at lower levels.

Outline

- 1 Hierarchical time series data
- 2 Hierarchical forecasting using single-level approaches
- 3 Linear forecast reconciliation
- 4 Example: Australian tourism
- 5 Fast computational tricks

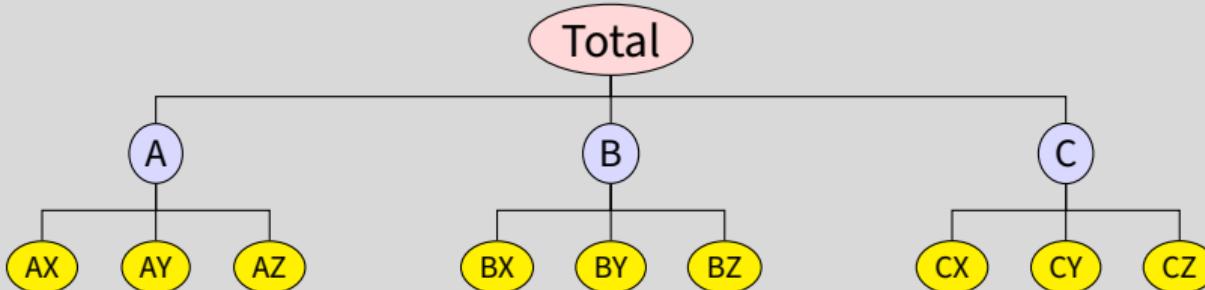
Fast computation: hierarchical data



$$\begin{aligned}
 \mathbf{y}_t = & \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{C,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix} = \\
 & = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix}
 \end{aligned}$$

$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t$$

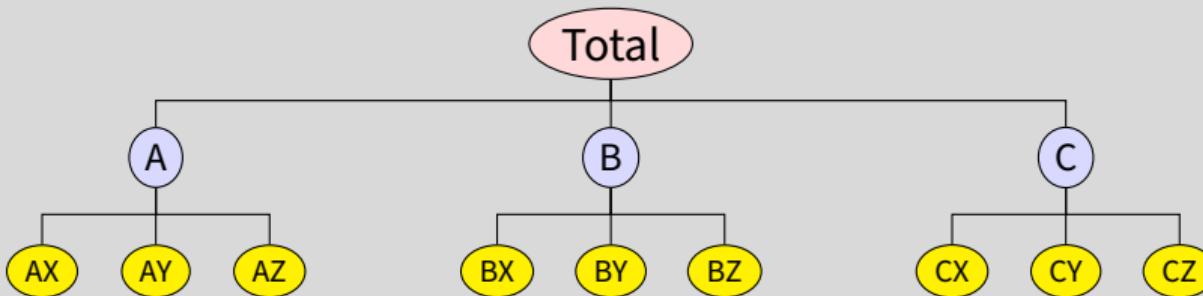
Fast computation: hierarchical data



$$\begin{aligned}
 \mathbf{y}_t = & \begin{pmatrix} y_t \\ y_{A,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{B,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{C,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix} = \\
 & = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix}
 \end{aligned}$$

$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t$$

Fast computation: hierarchical data

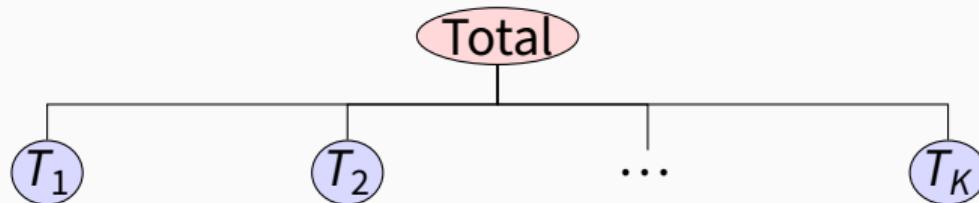


$$\begin{aligned}
 \mathbf{y}_t = & \begin{pmatrix} y_t \\ y_{A,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{B,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{C,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix} = \\
 & = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix}
 \end{aligned}$$

$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t$$

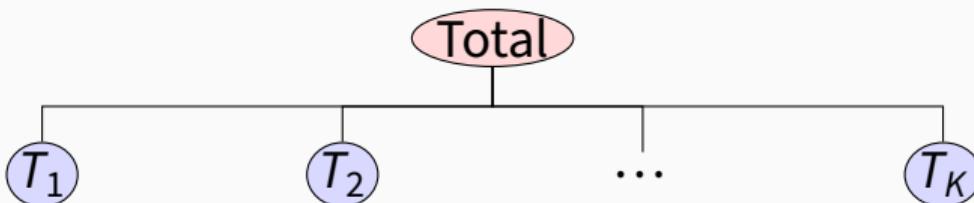
Fast computation: hierarchies

Think of the hierarchy as a tree of trees:



Fast computation: hierarchies

Think of the hierarchy as a tree of trees:



Then the summing matrix contains k smaller summing matrices:

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

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

Fast computation: hierarchies

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

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

Fast computation: hierarchies

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

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

Now apply the Sherman-Morrison formula ...

Fast computation: hierarchies

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

- \mathbf{S}_0 can be partitioned into K^2 blocks, with the (k, ℓ) block (of dimension $n_k \times n_\ell$) being $(\mathbf{S}'_k \Lambda_k \mathbf{S}_k)^{-1} \mathbf{J}_{n_k, n_\ell} (\mathbf{S}'_\ell \Lambda_\ell \mathbf{S}_\ell)^{-1}$
- \mathbf{J}_{n_k, n_ℓ} is a $n_k \times n_\ell$ matrix of ones
- $c^{-1} = \lambda_0^{-1} + \sum_k \mathbf{1}'_{n_k} (\mathbf{S}'_k \Lambda_k \mathbf{S}_k)^{-1} \mathbf{1}_{n_k}$
- Each $\mathbf{S}'_k \Lambda_k \mathbf{S}_k$ can be inverted similarly
- $\mathbf{S}' \Lambda \mathbf{y}$ can also be computed recursively

Fast computation: hierarchies

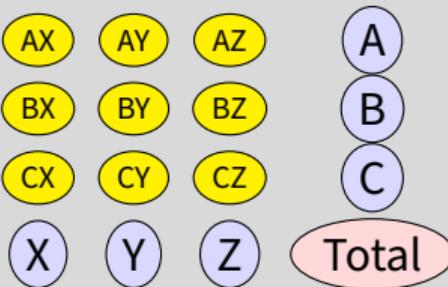
$$(\mathbf{S}'\Lambda\mathbf{S})^{-1} = \begin{bmatrix} (\mathbf{S}'_1\Lambda_1\mathbf{S}_1)^{-1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & (\mathbf{S}'_2\Lambda_2\mathbf{S}_2)^{-1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & (\mathbf{S}'_K\Lambda_K\mathbf{S}_K)^{-1}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

- \mathbf{S}_0 can be partitioned into K^2 blocks, with the (k, ℓ) block (of dimension $n_k \times n_\ell$) being $(\mathbf{S}'_k\Lambda_k\mathbf{S}_k)^{-1}\mathbf{J}_{n_k, n_\ell}(\mathbf{S}'_\ell\Lambda_\ell\mathbf{S}_\ell)^{-1}$
- \mathbf{J}_{n_k, n_ℓ} is a $n_k \times n_\ell$ matrix of ones
- $c^{-1} = \lambda_0^{-1} + \sum_k \mathbf{1}'_{n_k} (\mathbf{S}'_k\Lambda_k\mathbf{S}_k)^{-1} \mathbf{1}_{n_k}$
- Each $\mathbf{S}'_k\Lambda_k\mathbf{S}_k$ can be inverted similarly
- $\mathbf{S}'\Lambda\mathbf{y}$ can also be computed recursively



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

Fast computation: grouped data



$$\mathbf{y}_t = \begin{pmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{C,t} \\ y_{X,t} \\ y_{Y,t} \\ y_{Z,t} \\ y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{AX,t} \\ y_{AY,t} \\ y_{AZ,t} \\ y_{BX,t} \\ y_{BY,t} \\ y_{BZ,t} \\ y_{CX,t} \\ y_{CY,t} \\ y_{CZ,t} \end{pmatrix}$$

$$\mathbf{y}_t = \mathbf{S}\mathbf{b}_t$$

Fast computation: grouped data

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

m = number of rows
 n = number of columns

Fast computation: grouped data

$$\mathbf{S} = \begin{bmatrix} \mathbf{1}'_m \otimes \mathbf{1}'_n \\ \mathbf{1}'_m \otimes \mathbf{I}_n \\ \mathbf{I}_m \otimes \mathbf{1}'_n \\ \mathbf{I}_m \otimes \mathbf{I}_n \end{bmatrix} \quad \begin{aligned} m &= \text{number of rows} \\ n &= \text{number of columns} \end{aligned}$$

$$\mathbf{S}' \boldsymbol{\Lambda} \mathbf{S} = \lambda_{00} \mathbf{J}_{mn} + (\boldsymbol{\Lambda}_R \otimes \mathbf{J}_n) + (\mathbf{J}_m \otimes \boldsymbol{\Lambda}_C) + \boldsymbol{\Lambda}_U$$

- $\boldsymbol{\Lambda}_R$, $\boldsymbol{\Lambda}_C$ and $\boldsymbol{\Lambda}_U$ are diagonal matrices corresponding to rows, columns and unaggregated series;
- λ_{00} corresponds to aggregate.

Fast computation: grouped data

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

$\mathbf{A} = \Lambda_U^{-1} - \Lambda_U^{-1} (\mathbf{J}_m \otimes \mathbf{D}) \Lambda_U^{-1} - \mathbf{E} \mathbf{M}^{-1} \mathbf{E}'$. \(\mathbf{D}\) is diagonal with elements $d_k = \lambda_{0j}/\lambda_{j0}$.
has $m \times m$ blocks where \mathbf{e}_{ij} has k th element

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

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

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

Fast computation: grouped data

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

$\mathbf{A} = \Lambda_U^{-1} - \Lambda_U^{-1} (\mathbf{J}_m \otimes \mathbf{D}) \Lambda_U^{-1} - \mathbf{E} \mathbf{M}^{-1} \mathbf{E}'$. \(\mathbf{D}\) is diagonal with elements \(\lambda_{0j}/\lambda_{00}\).
has $m \times m$ blocks where \mathbf{e}_{ij} has k th element

$$\begin{pmatrix} \sqrt{1/2} & -1 & \sqrt{1/2} & -2 \end{pmatrix} \quad i = j,$$

$$i \neq j.$$

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

\mathbf{M} is $m \times m$ the large matrices involved.



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

Fast forecast reconciliation using linear model

References

-  Ashouri, M, RJ Hyndman, and G Shmueli (2022). Fast forecast reconciliation using linear models. *J Computational & Graphical Statistics* **31**(1), 263–282.
-  Hyndman, RJ, RA Ahmed, G Athanasopoulos, and HL Shang (2011). Optimal combination forecasts for hierarchical time series. *Computational Statistics & Data Analysis* **55**(9), 2579–2589.
-  Hyndman, RJ, A Lee, and E Wang (2016). Fast computation of reconciled forecasts for hierarchical and grouped time series. *Computational Statistics & Data Analysis* **97**, 16–32.
-  Wickramasuriya, SL, G Athanasopoulos, and RJ Hyndman (2019). Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *J American Statistical Association* **114**(526), 804–819.