Chapter 11: Forecasting hierarchical and grouped time series

Ankit Gupta

10/04/2023'

Time series can often be naturally disaggregated by various attributes of interest. For example, the total number of bicycles sold by a cycling manufacturer can be disaggregated by product type such as road bikes, mountain bikes and hybrids. Each of these can be disaggregated into finer categories. For example hybrid bikes can be divided into city, commuting, comfort, and trekking bikes; and so on. These categories are nested within the larger group categories, and so the collection of time series follows a hierarchical aggregation structure. Therefore we refer to these as "hierarchical time series".

Hierarchical time series often arise due to geographic divisions. For example, the total bicycle sales can be disaggregated by country, then within each country by state, within each state by region, and so on down to the outlet level.

In this chapter we discuss forecasting large collections of time series that aggregate in some way. The challenge is that we require forecasts that are coherent across the entire aggregation structure. That is, we require forecasts to add up in a manner that is consistent with the aggregation structure of the hierarchy or group that defines the collection of time series.

11.1 Hierarchical and grouped time series

Hierarchical time series

Figure 11.1 shows a simple hierarchical structure. At the top of the hierarchy is the "Total", the most aggregate level of the data. The tth observation of the Total series is denoted by y_t for $t = 1, ..., T$. The Total is disaggregated into two series, which in turn are divided into three and two series respectively at the bottom level of the hierarchy. Below the top level, we use $y_{j,t}$ to denote the tth observation of the series corresponding to node j. For example, $y_{A,t}$ denotes the tth observation of the series corresponding to node A, $y_{AB,t}$ denotes the tth observation of the series corresponding to node AB, and so on.

In this small example, the total number of series in the hierarchy is $n = 1 + 2 + 5 = 8$, while the number of series at the bottom level is $m = 5$. Note that $n > m$ in all hierarchies.

For any time
t , the observations at the bottom level of the hierarchy will sum to the observations of the series above. For example,

$y_t = y_{AA,t} + y_{AB,t} + y_{AC,t} + y_{BA,t} + y_{BB,t}$, (11.1)

$y_{A,t} = y_{AA,t} + y_{AB,t} + y_{AC,t}$ and

$y_{B,t} = y_{BA,t} + y_{BB,t}$. (11.2)

Substituting (11.2) into (11.1), we also get $y_t = y_{A,t} + y_{B,t.}$

Example: Australian tourism hierarchy

Australia is divided into six states and two territories, with each one having its own government and some economic and administrative autonomy. For simplicity, we refer to both states and territories as "states". Each of these states can be further subdivided into regions as shown in Figure 11.2 and Table 11.1. In total
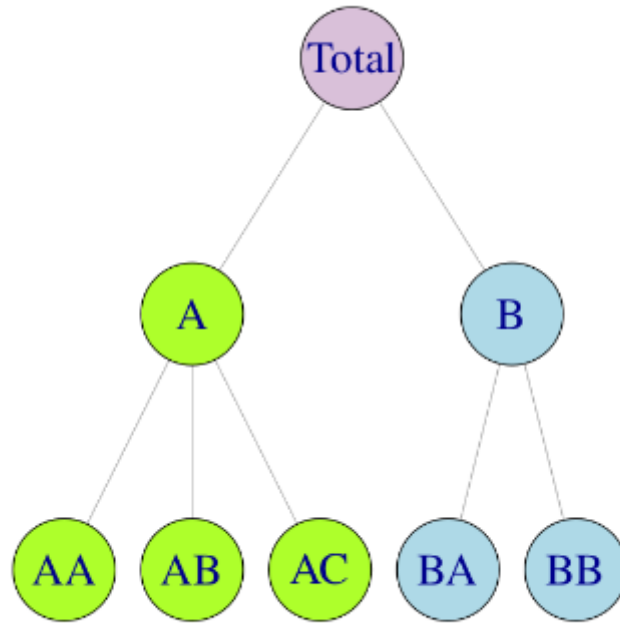
Figure 1: Figure 11.1: A two level hierarchical tree diagram.

there are 76 such regions. Business planners and tourism authorities are interested in forecasts for the whole of Australia, for each of the states and territories, and also for the regions.

The tourism tsibble contains data on quarterly domestic tourism demand, measured as the number of overnight trips Australians spend away from home. The key variables State and Region denote the geographical areas, while a further key Purpose describes the purpose of travel. For now, we will ignore the purpose of travel and just consider the geographic hierarchy. To make the graphs and tables simpler, we will recode State to use abbreviations.

```
library(fpp3)
```

```
## -- Attaching packages ---------------------------------------------- fpp3 0.5 --
## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.3     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.1
## v lubridate   1.9.3     v fable       0.3.3
## v ggplot2     3.4.4     v fabletools  0.3.4

## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()      masks base::date()
## x dplyr::filter()        masks stats::filter()
## x tsibble::intersect()   masks base::intersect()
## x tsibble::interval()    masks lubridate::interval()
## x dplyr::lag()           masks stats::lag()
## x tsibble::setdiff()     masks base::setdiff()
## x tsibble::union()       masks base::union()
```

```
tourism <- tsibble::tourism |>
  mutate(State = recode(State,
    `New South Wales` = "NSW",
```

```
    `Northern Territory` = "NT",
    `Queensland` = "QLD",
    `South Australia` = "SA",
    `Tasmania` = "TAS",
    `Victoria` = "VIC",
    `Western Australia` = "WA"
 ))
```

Using the aggregate_key() function, we can create the hierarchical time series with overnight trips in regions at the bottom level of the hierarchy, aggregated to states, which are aggregated to the national total. A hierarchical time series corresponding to the nested structure is created using a parent/child specification.

```
tourism_hts <- tourism |>
  aggregate_key(State / Region, Trips = sum(Trips))
tourism_hts
```

```
## # A tsibble: 6,800 x 4 [1Q]
## # Key:       State, Region [85]
##    Quarter State        Region        Trips
##      <qtr> <chr*>       <chr*>        <dbl>
##  1 1998 Q1 <aggregated> <aggregated> 23182.
##  2 1998 Q2 <aggregated> <aggregated> 20323.
##  3 1998 Q3 <aggregated> <aggregated> 19827.
##  4 1998 Q4 <aggregated> <aggregated> 20830.
##  5 1999 Q1 <aggregated> <aggregated> 22087.
##  6 1999 Q2 <aggregated> <aggregated> 21458.
##  7 1999 Q3 <aggregated> <aggregated> 19914.
##  8 1999 Q4 <aggregated> <aggregated> 20028.
##  9 2000 Q1 <aggregated> <aggregated> 22339.
## 10 2000 Q2 <aggregated> <aggregated> 19941.
## # i 6,790 more rows
```

The new tsibble now has some additional rows corresponding to state and national aggregations for each quarter. Figure 11.3 shows the aggregate total overnight trips for the whole of Australia as well as the states, revealing diverse and rich dynamics. For example, there is noticeable national growth since 2010 and for some states such as the ACT, New South Wales, Queensland, South Australia, and Victoria. There seems to be a significant jump for Western Australia in 2014.

```
tourism_hts |>
  filter(is_aggregated(Region)) |>
  autoplot(Trips) +
  labs(y = "Trips ('000)",
       title = "Australian tourism: national and states") +
  facet_wrap(vars(State), scales = "free_y", ncol = 3) +
  theme(legend.position = "none")
```

Australian tourism: national and states
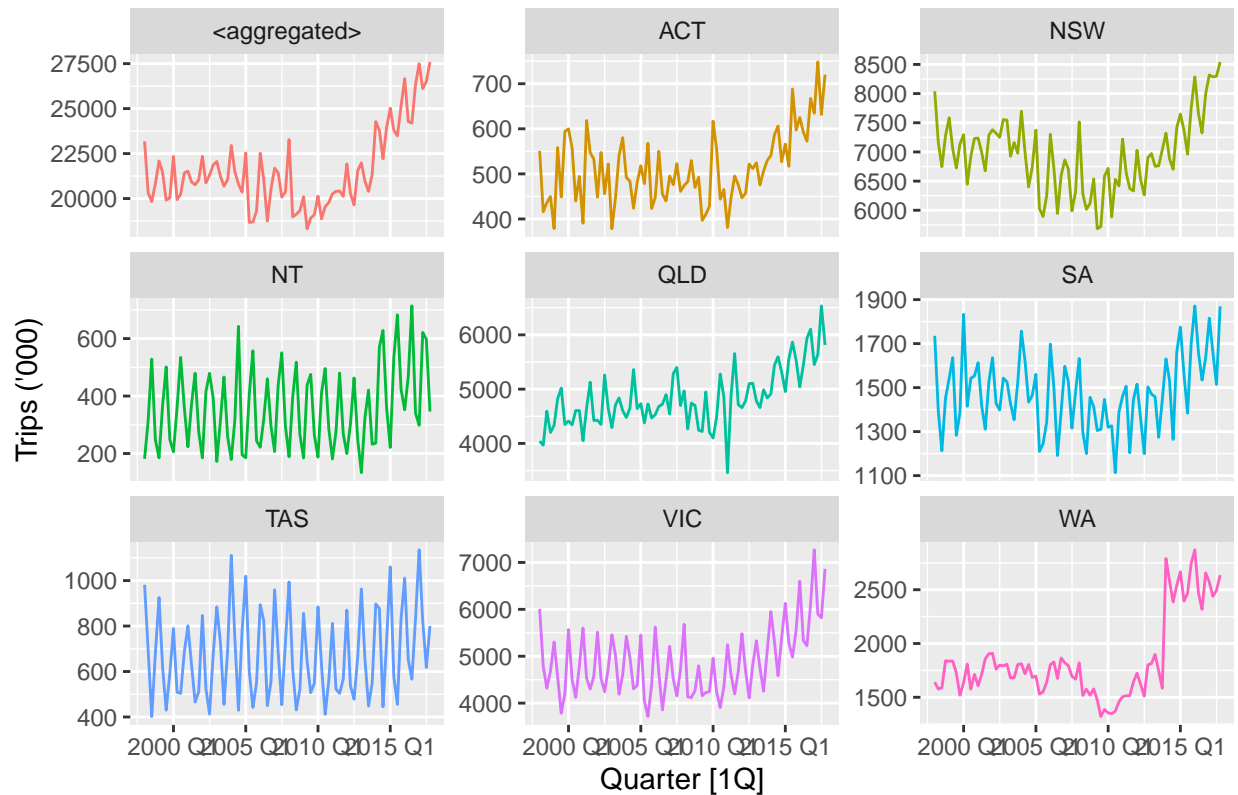

```
tourism_hts |>
  filter(State == "NT" | State == "QLD" |
          State == "TAS" | State == "VIC", is_aggregated(Region)) |>
  select(-Region) |>
  mutate(State = factor(State, levels=c("QLD","VIC","NT","TAS"))) |>
  gg_season(Trips) +
  facet_wrap(vars(State), nrow = 2, scales = "free_y")+
  labs(y = "Trips ('000)")
```
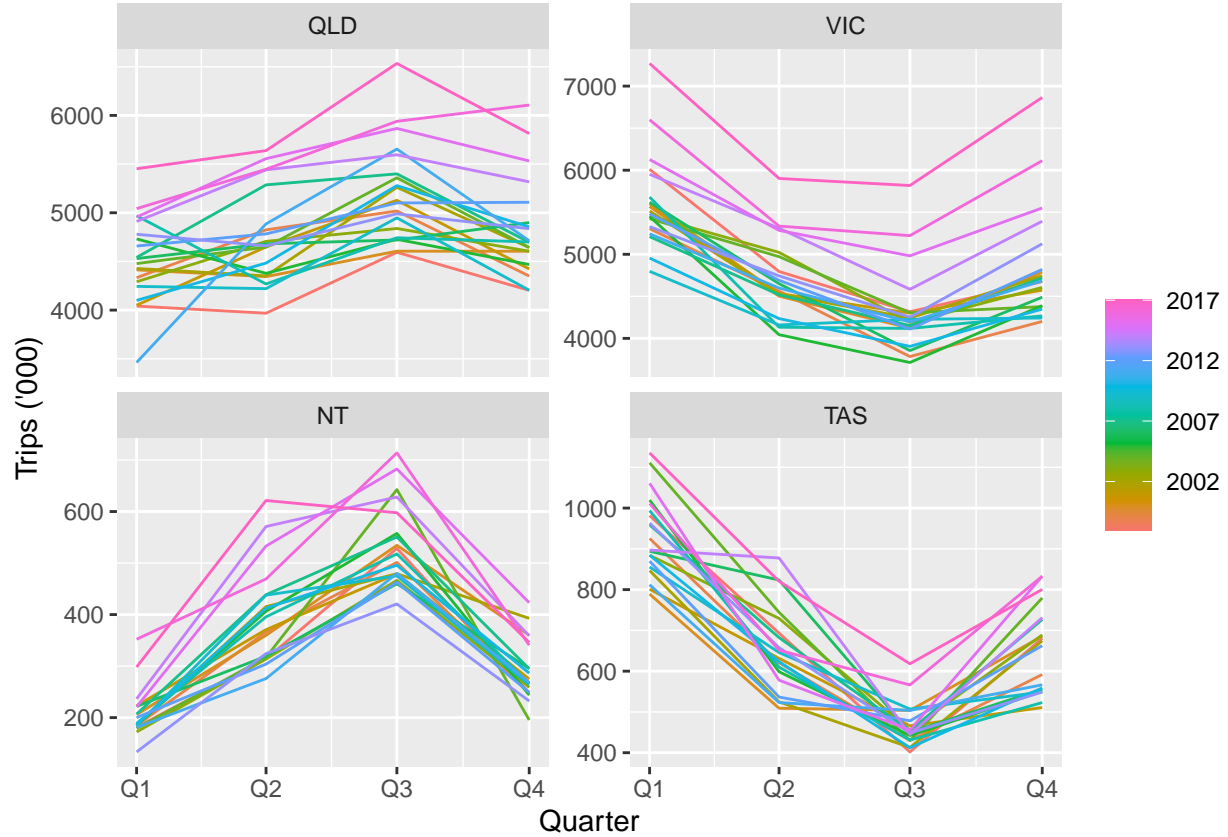
The seasonal pattern of the northern states, such as Queensland and the Northern Territory, leads to peak visits in winter (corresponding to Q3) due to the tropical climate and rainy summer months. In contrast, the southern states tend to peak in summer (corresponding to Q1). This is highlighted in the seasonal plots shown in Figure 11.4 for Queensland and the Northern Territory (shown in the left column) versus the most southern states of Victoria and Tasmania (shown in the right column).

The plots in Figure 11.5 shows data for some selected regions. These help us visualise the diverse regional dynamics within each state, with some series showing strong trends or seasonality, some showing contrasting seasonality, while some series appear to be just noise.

Grouped time series

With grouped time series, the data structure does not naturally disaggregate in a unique hierarchical manner. Figure 11.6 shows a simple grouped structure. At the top of the grouped structure is the Total, the most aggregate level of the data, again represented by y_t}. The Total can be disaggregated by attributes (A, B) forming series $y_{A,t}$ and $y_{B,t}$, or by attributes (X, Y) forming series $y_{X,t}$ and $y_{Y,t}$. At the bottom level, the data are disaggregated by both attributes.

Grouped time series can sometimes be thought of as hierarchical time series that do not impose a unique hierarchical structure, in the sense that the order by which the series can be grouped is not unique.

Example: Australian prison population   In this example we consider the Australia prison population data introduced in Chapter 2. The top panel in Figure 11.7 shows the total number of prisoners in Australia over the period 2005Q1–2016Q4. This represents the top-level series in the grouping structure. The panels below show the prison population disaggregated or grouped by (a) state (b) legal status (whether prisoners have already been sentenced or are in remand waiting for a sentence), and (c) gender. The three factors are crossed, but none are nested within the others.
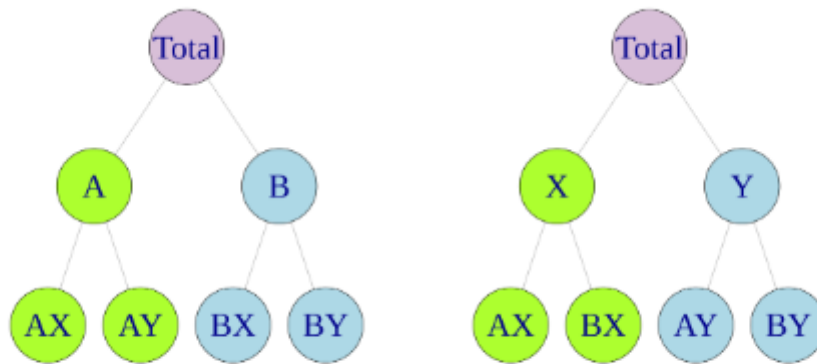
Figure 11.6: Alternative representations of a two level grouped structure.

Figure 2: Figure 11.6: Alternative representations of a two level grouped structure..

```r
prison <- readr::read_csv("https://OTexts.com/fpp3/extrafiles/prison_population.csv") |>
  mutate(Quarter = yearquarter(Date)) |>
  select(-Date)  |>
  as_tsibble(key = c(Gender, Legal, State, Indigenous),
             index = Quarter) |>
  relocate(Quarter)
```

```
## Rows: 3072 Columns: 6
## -- Column specification ------------------------------------------------------
## Delimiter: ","
## chr  (4): State, Gender, Legal, Indigenous
## dbl  (1): Count
## date (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

We create a grouped time series using aggregate_key() with attributes or groupings of interest now being crossed using the syntax attribute1*attribute2 (in contrast to the parent/child syntax used for hierarchical time series). The following code builds a grouped tsibble for the prison data with crossed attributes: gender, legal status and state.

```r
prison_gts <- prison |>
  aggregate_key(Gender * Legal * State, Count = sum(Count)/1e3)
```

Using is_aggregated() within filter() is helpful for exploring or plotting the main groups shown in the bottom panels of Figure 11.7. For example, the following code plots the total numbers of female and male prisoners across Australia.

```r
prison_gts |>
  filter(!is_aggregated(Gender), is_aggregated(Legal),
         is_aggregated(State)) |>
  autoplot(Count) +
```

6

```
labs(y = "Number of prisoners ('000)")
```
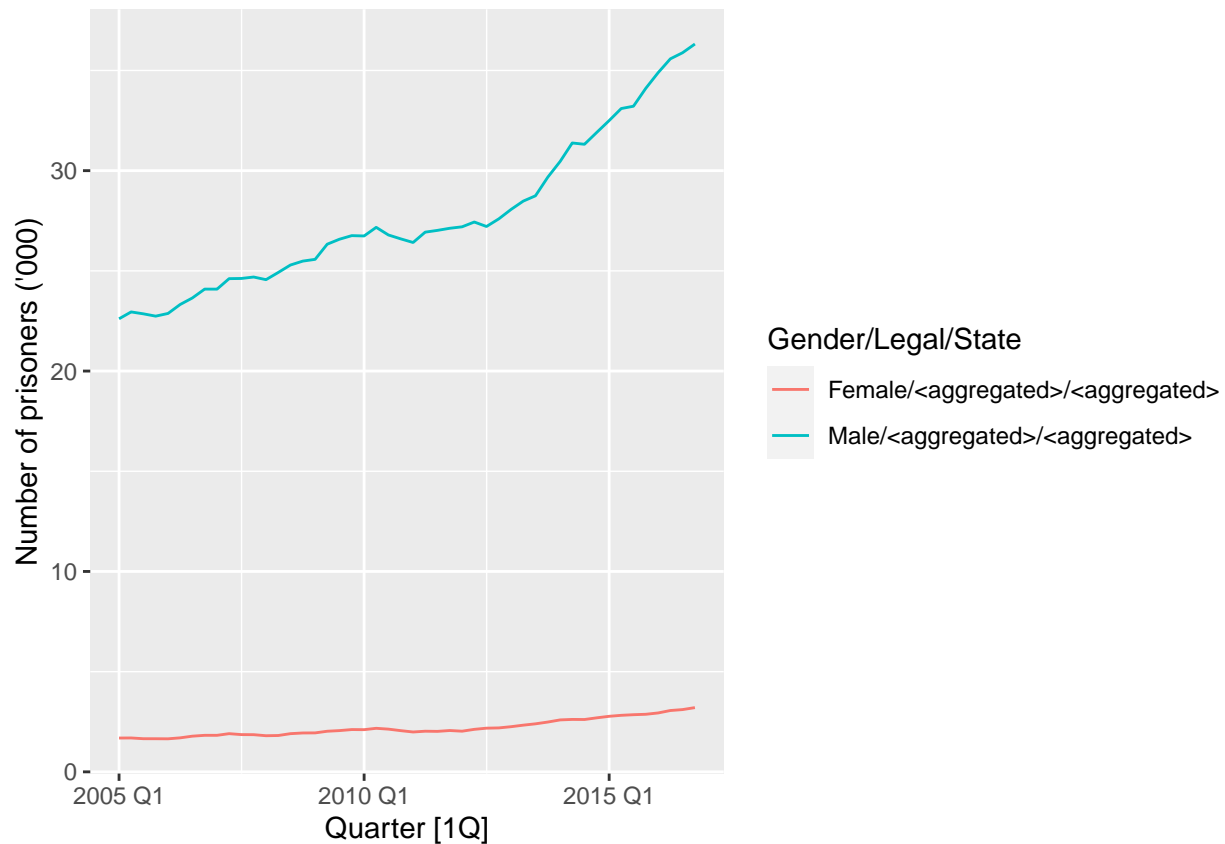

Plots of other group combinations can be obtained in a similar way. Figure 11.8 shows the Australian prison population grouped by all possible combinations of two attributes at a time: state and gender, state and legal status, and legal status and gender. The following code will reproduce the first plot in Figure 11.8.

```
prison_gts |>
  filter(!is_aggregated(Gender), !is_aggregated(Legal),
         !is_aggregated(State)) |>
  mutate(Gender = as.character(Gender)) |>
  ggplot(aes(x = Quarter, y = Count,
             group = Gender, colour=Gender)) +
  stat_summary(fun = sum, geom = "line") +
  labs(title = "Prison population by state and gender",
      y = "Number of prisoners ('000)") +
  facet_wrap(~ as.character(State),
             nrow = 1, scales = "free_y") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
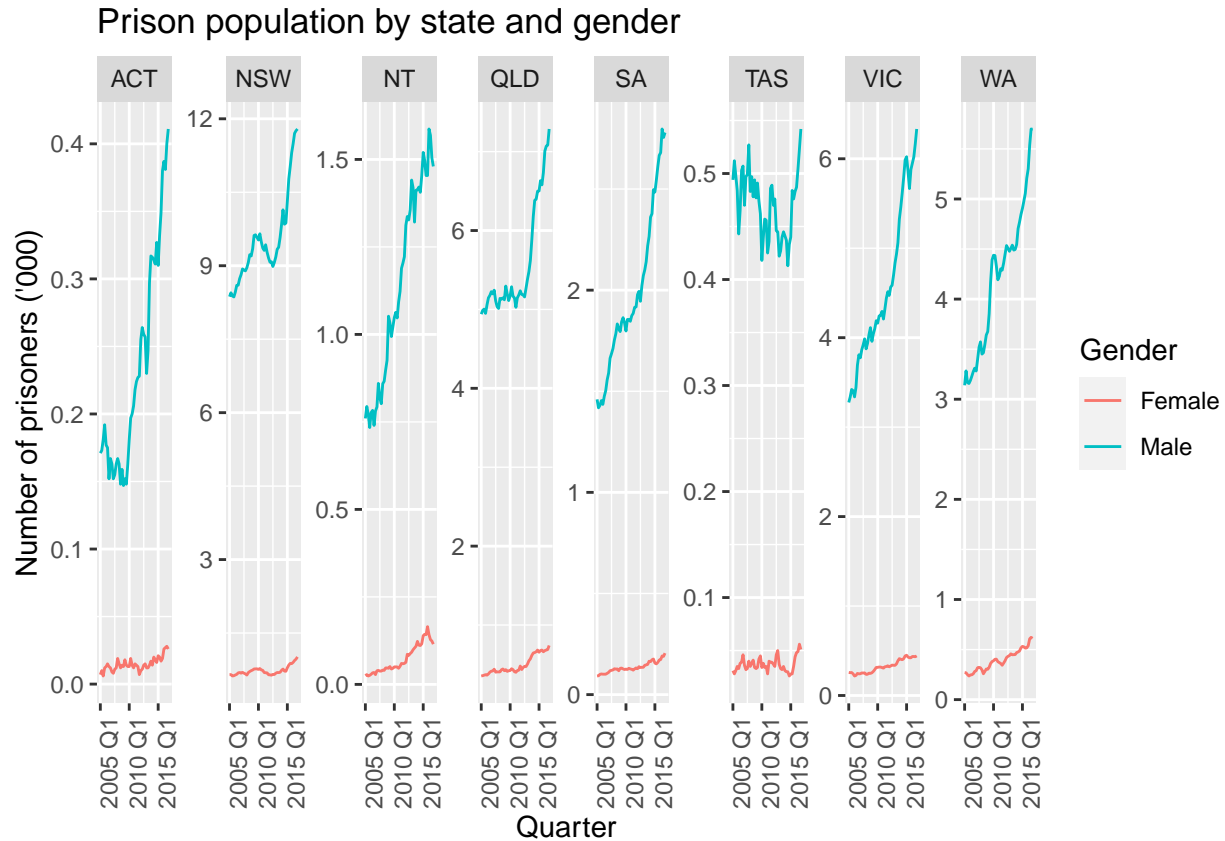
Figure 11.9 shows the Australian adult prison population disaggregated by all three attributes: state, legal status and gender. These form the bottom-level series of the grouped structure.

Mixed hierarchical and grouped structure

Often disaggregating factors are both nested and crossed. For example, the Australian tourism data can also be disaggregated by the four purposes of travel: holiday, business, visiting friends and relatives, and other. This grouping variable does not nest within any of the geographical variables. In fact, we could consider overnight trips split by purpose of travel for the whole of Australia, and for each state, and for each region. We describe such a structure as a "nested" geographic hierarchy "crossed" with the purpose of travel. Using aggregate_key() this can be specified by simply combining the factors.

```
tourism_full <- tourism |>
  aggregate_key((State/Region) * Purpose, Trips = sum(Trips))
```

The tourism_full tsibble contains 425 series, including the 85 series from the hierarchical structure, as well as another 340 series obtained when each series of the hierarchical structure is crossed with the purpose of travel.

11.2 Single level approaches

Traditionally, forecasts of hierarchical or grouped time series involved selecting one level of aggregation and generating forecasts for that level. These are then either aggregated for higher levels, or disaggregated for lower levels, to obtain a set of coherent forecasts for the rest of the structure.

The bottom-up approach

A simple method for generating coherent forecasts is the "bottom-up" approach. This approach involves first generating forecasts for each series at the bottom level, and then summing these to produce forecasts for all the series in the structure.

For example, for the hierarchy of Figure 11.1, we first generate
h-step-ahead forecasts for each of the bottom-level series: $\hat{y}_{AA,h}, \hat{y}_{AB,h}, \hat{y}_{AC,h}, \hat{y}_{BA,h}$

and $\hat{y}_{BB,h}$.

(We have simplified the previously used notation of $\hat{y}_{T+h|T}$ for brevity.)

Summing these, we get h-step-ahead coherent forecasts for the rest of the series:

$$\tilde{y}_h = \hat{y}_{AA,h} + \hat{y}_{AB,h} + \hat{y}_{AC,h} + \hat{y}_{BA,h} + \hat{y}_{BB,h}, \tilde{y}_{A,h} = \hat{y}_{AA,h} + \hat{y}_{AB,h} + \hat{y}_{AC,h}, \text{and } \tilde{y}_{B,h} = \hat{y}_{BA,h} + \hat{y}_{BB,h}.$$

(In this chapter, we will use the "tilde" notation to indicate coherent forecasts.)

An advantage of this approach is that we are forecasting at the bottom level of a structure, and therefore no information is lost due to aggregation. On the other hand, bottom-level data can be quite noisy and more challenging to model and forecast.

Example: Generating bottom-up forecasts

Suppose we want national and state forecasts for the Australian tourism data, but we aren't interested in disaggregations using regions or the purpose of travel. So we first create a simple tsibble object containing only state and national trip totals for each quarter.

```
tourism_states <- tourism |>
  aggregate_key(State, Trips = sum(Trips))
```

We could generate the bottom-level state forecasts first, and then sum them to obtain the national forecasts.

```
fcasts_state <- tourism_states |>
  filter(!is_aggregated(State)) |>
  model(ets = ETS(Trips)) |>
  forecast()

# Sum bottom-level forecasts to get top-level forecasts
fcasts_national <- fcasts_state |>
  summarise(value = sum(Trips), .mean = mean(value))
```

However, we want a more general approach that will work with all the forecasting methods discussed in this chapter. So we will use the reconcile() function to specify how we want to compute coherent forecasts.

```
tourism_states |>
  model(ets = ETS(Trips)) |>
  reconcile(bu = bottom_up(ets)) |>
  forecast()
```

```
## # A fable: 144 x 5 [1Q]
## # Key:     State, .model [18]
##     State  .model Quarter         Trips .mean
##     <chr*> <chr>   <qtr>          <dist> <dbl>
## 1 ACT      ets     2018 Q1  N(701, 7651)  701.
## 2 ACT      ets     2018 Q2  N(717, 8032)  717.
## 3 ACT      ets     2018 Q3  N(734, 8440)  734.
## 4 ACT      ets     2018 Q4  N(750, 8882)  750.
## 5 ACT      ets     2019 Q1  N(767, 9368)  767.
```

```
##  6 ACT     ets    2019 Q2  N(784, 9905)   784.
##  7 ACT     ets    2019 Q3 N(800, 10503)   800.
##  8 ACT     ets    2019 Q4 N(817, 11171)   817.
##  9 ACT     bu     2018 Q1  N(701, 7651)   701.
## 10 ACT     bu     2018 Q2  N(717, 8032)   717.
## # i 134 more rows
```

The reconcile() step has created a new "model" to produce bottom-up forecasts. The fable object contains the ets forecasts as well as the coherent bu forecasts, for the 8 states and the national aggregate. At the state level, these forecasts are identical, but the national ets forecasts will be different from the national bu forecasts.

For bottom-up forecasting, this is rather inefficient as we are not interested in the ETS model for the national total, and the resulting fable contains a lot of duplicates. But later we will introduce more advanced methods where we will need models for all levels of aggregation, and where the coherent forecasts are different from any of the original forecasts.

Workflow for forecasting aggregation structures

The above code illustrates the general workflow for hierarchical and grouped forecasts. We use the following pipeline of functions.

```
#data |> aggregate_key() |> model() |>
#reconcile() |> forecast()
```

1. Begin with a tsibble object (here labelled data) containing the individual bottom-level series.

2. Define in aggregate_key() the aggregation structure and build a tsibble object that also contains the aggregate series.

3. Identify a model() for each series, at all levels of aggregation.

4. Specify in reconcile() how the coherent forecasts are to be generated from the selected models.

5. Use the forecast() function to generate forecasts for the whole aggregation structure.

Top-down approaches

Top-down approaches involve first generating forecasts for the Total series y_t, and then disaggregating these down the hierarchy.

Let $p_1, ..., p_m$ denote a set of disaggregation proportions which determine how the forecasts of the Total series are to be distributed to obtain forecasts for each series at the bottom level of the structure. For example, for the hierarchy of Figure 11.1, using proportions $p_1, ..., p_5$

we get $\tilde{y}_{AA,t} = p_1 \hat{y}_t, \tilde{y}_{AB,t} = p_2 \hat{y}_t, \tilde{y}_{AC,t} = p_3 \hat{y}_t, \tilde{y}_{BA,t} = p_4 \hat{y}_t$ and $\tilde{y}_{BB,t} = p_5 \hat{y}_t$.

Once the bottom-level h-step-ahead forecasts have been generated, these are aggregated to generate coherent forecasts for the rest of the series.

Top-down forecasts can be generated using top_down() within the reconcile() function.

There are several possible top-down methods that can be specified. The two most common top-down approaches specify disaggregation proportions based on the historical proportions of the data. These performed well in the study of Gross & Sohl (1990).

Average historical proportions $p_j = \frac{1}{T} \Sigma_{t=1}^{T} \frac{y_{j,t}}{y_t}$

for j=1,…,m

Each proportion p_j reflects the average of the historical proportions of the bottom-level series $y_{j,t}$ over the period $t = 1, ..., T$ relative to the total aggregate y_t.

This approach is implemented in the top_down() function by setting method = "average_proportions".

Proportions of the historical averages $p_j = \Sigma_{t=1}^{T} \frac{y_{j,t}}{T} / \Sigma_{t=1}^{T} \frac{y_t}{T}$

for j=1,…,m. Each proportion p_j captures the average historical value of the bottom-level series y_{j,t}$ relative to the average value of the total aggregate y_t.

This approach is implemented in the top_down() function by setting method = "proportion_averages".

A convenient attribute of such top-down approaches is their simplicity. One only needs to model and generate forecasts for the most aggregated top-level series. In general, these approaches seem to produce quite reliable forecasts for the aggregate levels and they are useful with low count data. On the other hand, one disadvantage is the loss of information due to aggregation. Using such top-down approaches, we are unable to capture and take advantage of individual series characteristics such as time dynamics, special events, different seasonal patterns, etc.

Forecast proportions   Because historical proportions used for disaggregation do not take account of how those proportions may change over time, top-down approaches based on historical proportions tend to produce less accurate forecasts at lower levels of the hierarchy than bottom-up approaches. To address this issue, proportions based on forecasts rather than historical data can be used (G. Athanasopoulos et al., 2009).

Consider a one level hierarchy. We first generate h-step-ahead forecasts for all of the series. We don't use these forecasts directly, and they are not coherent (they don't add up correctly). Let's call these "initial" forecasts. We calculate the proportion of each h-step-ahead initial forecast at the bottom level, to the aggregate of all the h-step-ahead initial forecasts at this level. We refer to these as the forecast proportions, and we use them to disaggregate the top-level h-step-ahead initial forecast in order to generate coherent forecasts for the whole of the hierarchy.

Middle-out approach

The middle-out approach combines bottom-up and top-down approaches. Again, it can only be used for strictly hierarchical aggregation structures.

First, a "middle" level is chosen and forecasts are generated for all the series at this level. For the series above the middle level, coherent forecasts are generated using the bottom-up approach by aggregating the "middle-level" forecasts upwards. For the series below the "middle level", coherent forecasts are generated using a top-down approach by disaggregating the "middle level" forecasts downwards.

This approach is implemented in the middle_out() function by specifying the appropriate middle level via the level argument and selecting the top-down approach with the method argument.

11.3 Forecast reconciliation

11.4 Forecasting Australian domestic tourism

We will compute forecasts for the Australian tourism data that was described in Section 11.1. We use the data up to the end of 2015 as a training set, withholding the final two years (eight quarters, 2016Q1–2017Q4) as a test set for evaluation. The code below demonstrates the full workflow for generating coherent forecasts using the bottom-up, OLS and MinT methods.

```
tourism_full <- tourism |>
  aggregate_key((State/Region) * Purpose, Trips = sum(Trips))

fit <- tourism_full |>
```

level h-step-ahead initial forecast in order to generate coherent forecasts for the whole of the hierarchy.

For a K-level hierarchy, this process is repeated for each node, going from the top to the bottom level. Applying this process leads to the following general rule for obtaining the forecast proportions:

$$p_j = \prod_{\ell=0}^{K-1} \frac{\hat{y}_{j,h}^{(\ell)}}{\hat{S}_{j,h}^{(\ell+1)}}$$

where $j = 1, 2, \ldots, m$, $\hat{y}_{j,h}^{(\ell)}$ is the h-step-ahead initial forecast of the series that corresponds to the node which is ℓ levels above j, and $\hat{S}_{j,h}^{(\ell)}$ is the sum of the h-step-ahead initial forecasts below the node that is ℓ levels above node j and are directly connected to that node. These forecast proportions disaggregate the h-step-ahead initial forecast of the Total series to get h-step-ahead coherent forecasts of the bottom-level series.

We will use the hierarchy of Figure 11.1 to explain this notation and to demonstrate how this general rule is reached. Assume we have generated initial forecasts for each series in the hierarchy. Recall that for the top-level "Total" series, $\tilde{y}_h = \hat{y}_h$, for any top-down approach. Here are some examples using the above notation:

Figure 3: Figure1

Here are some examples using the above notation:

- $\hat{y}_{A,h}^{(1)} = \hat{y}_{B,h}^{(1)} = \hat{y}_h = \tilde{y}_h$;
- $\hat{y}_{AA,h}^{(1)} = \hat{y}_{AB,h}^{(1)} = \hat{y}_{AC,h}^{(1)} = \hat{y}_{A,h}$;
- $\hat{y}_{AA,h}^{(2)} = \hat{y}_{AB,h}^{(2)} = \hat{y}_{AC,h}^{(2)} = \hat{y}_{BA,h}^{(2)} = \hat{y}_{BB,h}^{(2)} = \hat{y}_h = \tilde{y}_h$;
- $\hat{S}_{AA,h}^{(1)} = \hat{S}_{AB,h}^{(1)} = \hat{S}_{AC,h}^{(1)} = \hat{y}_{AA,h} + \hat{y}_{AB,h} + \hat{y}_{AC,h}$;
- $\hat{S}_{AA,h}^{(2)} = \hat{S}_{AB,h}^{(2)} = \hat{S}_{AC,h}^{(2)} = \hat{S}_{A,h}^{(1)} = \hat{S}_{B,h}^{(1)} = \hat{S}_h = \hat{y}_{A,h} + \hat{y}_{B,h}$.

Moving down the farthest left branch of the hierarchy, coherent forecasts are given by

$$\tilde{y}_{A,h} = \left(\frac{\hat{y}_{A,h}}{\hat{S}_{A,h}^{(1)}} \right) \tilde{y}_h = \left(\frac{\hat{y}_{AA,h}^{(1)}}{\hat{S}_{AA,h}^{(2)}} \right) \tilde{y}_h$$

and

$$\tilde{y}_{AA,h} = \left(\frac{\hat{y}_{AA,h}}{\hat{S}_{AA,h}^{(1)}} \right) \tilde{y}_{A,h} = \left(\frac{\hat{y}_{AA,h}}{\hat{S}_{AA,h}^{(1)}} \right) \left(\frac{\hat{y}_{AA,h}^{(1)}}{\hat{S}_{AA,h}^{(2)}} \right) \tilde{y}_h.$$

Figure 4: Figure2

^{AA,h} ^{AA,h} ^{AA,h}

Consequently,

$$p_1 = \left(\frac{\hat{y}_{AA,h}}{\hat{S}_{AA,h}^{(1)}} \right) \left(\frac{\hat{y}_{AA,h}^{(1)}}{\hat{S}_{AA,h}^{(2)}} \right).$$

The other proportions can be obtained similarly.

This approach is implemented in the `top_down()` function by setting `method = "forecast_proportions"`. Because this approach tends to work better than other top-down methods, it is the default choice in the `top_down()` function when no `method` argument is specified.

One disadvantage of all top-down approaches, is that they do not produce unbiased coherent forecasts (Hyndman et al., 2011) even if the base forecasts are unbiased.

Figure 5: Figure..

Warning: the rest of this chapter is more advanced and assumes a knowledge of some basic matrix algebra.

Matrix notation

Recall that Equations (11.1) and (11.2) represent how data, that adhere to the hierarchical structure of Figure 11.1, aggregate. Similarly (11.3) and (11.4) represent how data, that adhere to the grouped structure of Figure 11.6, aggregate. These equations can be thought of as aggregation constraints or summing equalities, and can be more efficiently represented using matrix notation.

For any aggregation structure we construct an $n \times m$ matrix S (referred to as the "summing matrix") which dictates the way in which the bottom-level series aggregate.

For the hierarchical structure in Figure 11.1, we can write

Figure 6: Figure..

For the hierarchical structure in Figure 11.1, we can write

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

or in more compact notation

$$y_t = Sb_t, \tag{11.5}$$

where y_t is an n-dimensional vector of all the observations in the hierarchy at time t, S is the summing matrix, and b_t is an m-dimensional vector of all the observations in the bottom level of the hierarchy at time t. Note that the first row in the summing matrix S represents Equation (11.1), the second and third rows represent (11.2). The rows below these comprise an m-dimensional identity matrix I_m so that each bottom-level observation on the right hand side of the equation is equal to itself on the left hand side.

Figure 7: Figure..

14

Similarly for the grouped structure of Figure 11.6 we write

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

or

$$
\boldsymbol{y}_t = \boldsymbol{S}\boldsymbol{b}_t, \tag{11.6}
$$

where the second and third rows of $\boldsymbol{S}$ represent Equation (11.3) and the fourth and fifth rows represent (11.4).

Figure 8: Figure..

Mapping matrices

This matrix notation allows us to represent all forecasting methods for hierarchical or grouped time series using a common notation.

Suppose we forecast all series ignoring any aggregation constraints. We call these the **base forecasts** and denote them by $\hat{\boldsymbol{y}}_h$ where h is the forecast horizon. They are stacked in the same order as the data $\boldsymbol{y}_t$.

Then all coherent forecasting approaches for either hierarchical or grouped structures can be represented as[23]

$$
\tilde{\boldsymbol{y}}_h = \boldsymbol{S}\boldsymbol{G}\hat{\boldsymbol{y}}_h, \tag{11.7}
$$

where $\boldsymbol{G}$ is a matrix that maps the base forecasts into the bottom level, and the summing matrix $\boldsymbol{S}$ sums these up using the aggregation structure to produce a set of **coherent forecasts** $\tilde{\boldsymbol{y}}_h$.

The $\boldsymbol{G}$ matrix is defined according to the approach implemented. For example if the bottom–up approach is used to forecast the hierarchy of Figure 11.1, then

Figure 9: Figure..

The G matrix is defined according to the approach implemented. For example if the bottom-up approach is used to forecast the hierarchy of Figure 11.1, then

$$G = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Notice that G contains two partitions. The first three columns zero out the base forecasts of the series above the bottom level, while the m-dimensional identity matrix picks only the base forecasts of the bottom level. These are then summed by the S matrix.

If any of the top-down approaches were used then

$$G = \begin{bmatrix} p_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The first column includes the set of proportions that distribute the base forecasts of the top level to the bottom level. These are then summed up by the S matrix. The rest of the columns

Figure 10: Figure..

The first column includes the set of proportions that distribute the base forecasts of the top level to the bottom level. These are then summed up by the S matrix. The rest of the columns zero out the base forecasts below the highest level of aggregation.

For a middle out approach, the G matrix will be a combination of the above two. Using a set of proportions, the base forecasts of some pre-chosen level will be disaggregated to the bottom level, all other base forecasts will be zeroed out, and the bottom-level forecasts will then be summed up the hierarchy via the summing matrix.

Forecast reconciliation

Equation (11.7) shows that pre-multiplying any set of base forecasts with SG will return a set of coherent forecasts.

The traditional methods considered so far are limited in that they only use base forecasts from a single level of aggregation which have either been aggregated or disaggregated to obtain forecasts at all other levels. Hence, they use limited information. However, in general, we could use other G matrices, and then SG combines and reconciles all the base forecasts in order to produce coherent forecasts.

In fact, we can find the optimal G matrix to give the most accurate reconciled forecasts.

Figure 11: Figure..

The MinT optimal reconciliation approach

Wickramasuriya et al. (2019) found a $\boldsymbol{G}$ matrix that minimises the total forecast variance of the set of coherent forecasts, leading to the MinT (Minimum Trace) optimal reconciliation approach.

Suppose we generate coherent forecasts using Equation (11.7). First we want to make sure we have unbiased forecasts. If the base forecasts $\hat{\boldsymbol{y}}_h$ are unbiased, then the coherent forecasts $\tilde{\boldsymbol{y}}_h$ will be unbiased provided[24] $\boldsymbol{SGS} = \boldsymbol{S}$. This provides a constraint on the matrix $\boldsymbol{G}$. Interestingly, no top-down method satisfies this constraint, so all top-down approaches result in biased coherent forecasts.

Next we need to find the errors in our forecasts. Wickramasuriya et al. (2019) show that the variance-covariance matrix of the h-step-ahead coherent forecast errors is given by

$$\boldsymbol{V}_h = \mathrm{Var}[\boldsymbol{y}_{T+h} - \tilde{\boldsymbol{y}}_h] = \boldsymbol{SGW}_h\boldsymbol{G}'\boldsymbol{S}'$$

where $\boldsymbol{W}_h = \mathrm{Var}[(\boldsymbol{y}_{T+h} - \hat{\boldsymbol{y}}_h)]$ is the variance-covariance matrix of the corresponding base forecast errors.

The objective is to find a matrix $\boldsymbol{G}$ that minimises the error variances of the coherent forecasts. These error variances are on the diagonal of the matrix $\boldsymbol{V}_h$, and so the sum of all the error variances is given by the trace of the matrix $\boldsymbol{V}_h$. Wickramasuriya et al. (2019) show that the

Figure 12: Figure..

The objective is to find a matrix G that minimises the error variances of the coherent forecasts. These error variances are on the diagonal of the matrix V_h, and so the sum of all the error variances is given by the trace of the matrix V_h. Wickramasuriya et al. (2019) show that the matrix G which minimises the trace of V_h such that $SGS = S$, is given by

$$G = (S'W_h^{-1}S)^{-1}S'W_h^{-1}.$$

Therefore, the optimally reconciled forecasts are given by

$$\tilde{y}_h = S(S'W_h^{-1}S)^{-1}S'W_h^{-1}\hat{y}_h. \tag{11.8}$$

We refer to this as the MinT (or Minimum Trace) optimal reconciliation approach. MinT is implemented by `min_trace()` within the `reconcile()` function.

To use this in practice, we need to estimate W_h, the forecast error variance of the h-step-ahead base forecasts. This can be difficult, and so we provide four simplifying approximations that have been shown to work well in both simulations and in practice.

1. Set $W_h = k_h I$ for all h, where $k_h > 0$.[25] This is the most simplifying assumption to make, and means that G is independent of the data, providing substantial computational savings. The disadvantage, however, is that this specification does not account for the differences in scale between the levels of the structure, or for relationships between series.

   Setting $W_h = k_h I$ in (11.8) gives the ordinary least squares (OLS) estimator we introduced

Figure 13: Figure..

19

Setting $\boldsymbol{W}_h = k_h\boldsymbol{I}$ in (11.8) gives the ordinary least squares (OLS) estimator we introduced in Section 7.9 with $\boldsymbol{X} = \boldsymbol{S}$ and $\boldsymbol{y} = \hat{\boldsymbol{y}}$. Hence this approach is usually referred to as OLS reconciliation. It is implemented in `min_trace()` by setting `method = "ols"`.

. Set $\boldsymbol{W}_h = k_h\text{diag}(\hat{\boldsymbol{W}}_1)$ for all h, where $k_h > 0$,

$$\hat{\boldsymbol{W}}_1 = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{e}_t\boldsymbol{e}_t',$$

and $\boldsymbol{e}_t$ is an n-dimensional vector of residuals of the models that generated the base forecasts stacked in the same order as the data.

This specification scales the base forecasts using the variance of the residuals and it is therefore referred to as the WLS (weighted least squares) estimator using *variance scaling*. The approach is implemented in `min_trace()` by setting `method = "wls_var"`.

. Set $\boldsymbol{W}_h = k_h\boldsymbol{\Lambda}$ for all h, where $k_h > 0$, $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{S1})$, and $\boldsymbol{1}$ is a unit vector of dimension m (the number of bottom-level series). This specification assumes that the bottom-level base forecast errors each have variance k_h and are uncorrelated between nodes. Hence each element of the diagonal $\boldsymbol{\Lambda}$ matrix contains the number of forecast error variances contributing to each node. This estimator only depends on the structure of the aggregations, and not on the actual data. It is therefore referred to as *structural scaling*.

Figure 14: Figure..

aggregations, and not on the actual data. It is therefore referred to as *structural scaling*. Applying the structural scaling specification is particularly useful in cases where residuals are not available, and so variance scaling cannot be applied; for example, in cases where the base forecasts are generated by judgmental forecasting (Chapter 6). The approach is implemented in `min_trace()` by setting `method = "wls_struct"`.

4. Set $W_h = k_h W_1$ for all h, where $k_h > 0$. Here we only assume that the error covariance matrices are proportional to each other, and we directly estimate the full one-step covariance matrix W_1. The most obvious and simple way would be to use the sample covariance. This is implemented in `min_trace()` by setting `method = "mint_cov"`.

However, for cases where the number of bottom-level series m is large compared to the length of the series T, this is not a good estimator. Instead we use a shrinkage estimator which shrinks the sample covariance to a diagonal matrix. This is implemented in `min_trace()` by setting `method = "mint_shrink"`.

In summary, unlike any other existing approach, the optimal reconciliation forecasts are generated using all the information available within a hierarchical or a grouped structure. This is important, as particular aggregation levels or groupings may reveal features of the data that are of interest to the user and are important to be modelled. These features may be completely hidden or not easily identifiable at other levels.

For example, consider the Australian tourism data introduced in Section 11.1, where the

Figure 15: Figure..

are of interest to the user and are important to be modelled. These features may be completely hidden or not easily identifiable at other levels.

For example, consider the Australian tourism data introduced in Section 11.1, where the hierarchical structure followed the geographic division of a country into states and regions. Some areas will be largely summer destinations, while others may be winter destinations. We saw in Figure 11.4 the contrasting seasonal patterns between the northern and the southern states. These differences will be smoothed at the country level due to aggregation.

Figure 16: Figure..

```
  filter(year(Quarter) <= 2015) |>
  model(base = ETS(Trips)) |>
  reconcile(
    bu = bottom_up(base),
    ols = min_trace(base, method = "ols"),
    mint = min_trace(base, method = "mint_shrink")
  )
```

Here, fit contains the base ETS model (discussed in Chapter 8) for each series in tourism_full, along with the three methods for producing coherent forecasts as specified in the reconcile() function.

```
fc <- fit |> forecast(h = "2 years")
```

Passing fit into forecast() generates base and coherent forecasts across all the series in the aggregation structure. Figures 11.12 and 11.13 plot the four point forecasts for the overnight trips for the Australian total, the states, and the purposes of travel, along with the actual observations of the test set.

```
fc |>
  filter(is_aggregated(Region), is_aggregated(Purpose)) |>
  autoplot(
    tourism_full |> filter(year(Quarter) >= 2011),
    level = NULL
  ) +
  labs(y = "Trips ('000)") +
  facet_wrap(vars(State), scales = "free_y")
```



```
fc |>
```

22
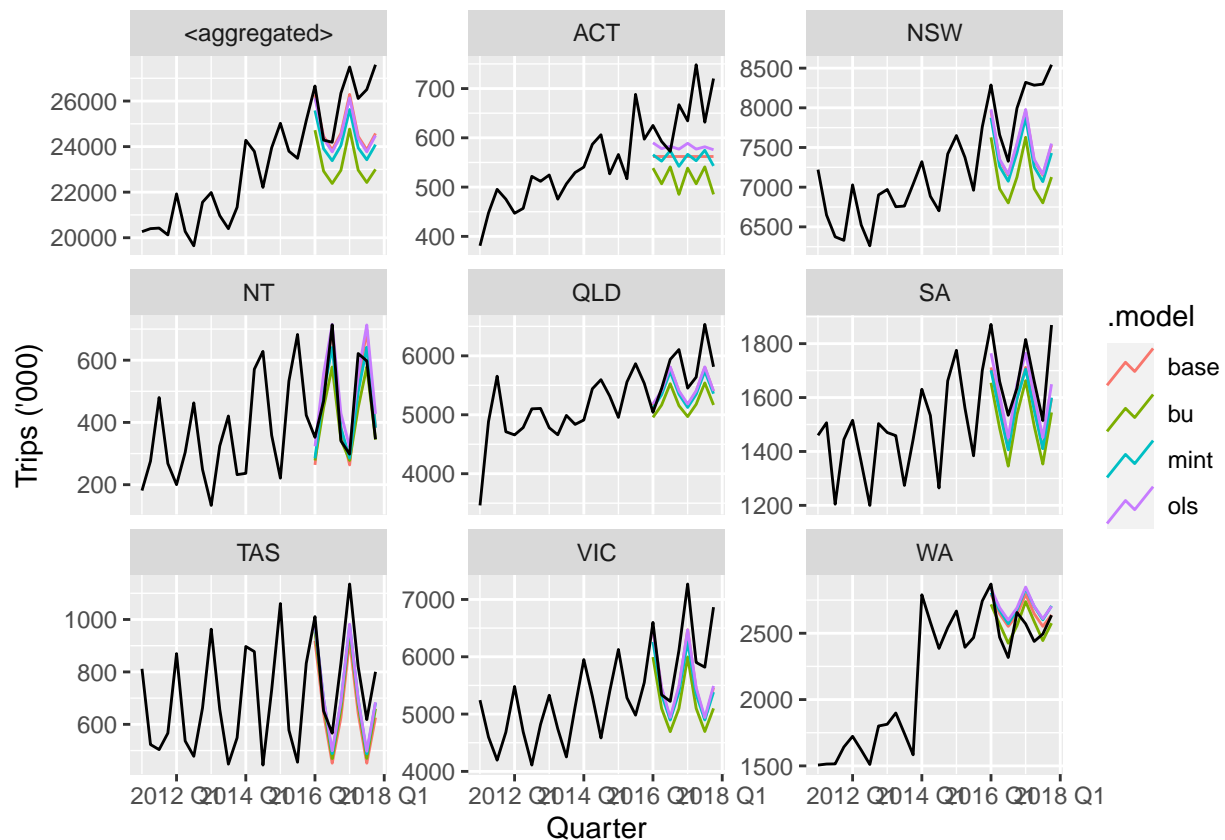
```
  filter(is_aggregated(State), !is_aggregated(Purpose)) |>
  autoplot(
    tourism_full |> filter(year(Quarter) >= 2011),
    level = NULL
  ) +
  labs(y = "Trips ('000)") +
  facet_wrap(vars(Purpose), scales = "free_y")
```


To make it easier to see the differences, we have included only the last five years of the training data, and have omitted the prediction intervals. In most panels, the increase in overnight trips, especially in the second half of the test set, is higher than what is predicted by the point forecasts. This is particularly noticeable for the mainland eastern states of ACT, New South Wales, Queensland and Victoria, and across all purposes of travel.

The accuracy of the forecasts over the test set can be evaluated using the accuracy() function. We summarise some results in Table 11.2 using RMSE and MASE.

The scales of the series at different levels of aggregation are quite different, due to aggregation. Hence, we need to be cautious when comparing or calculating scale dependent error measures, such as the RMSE, across levels as the aggregate series will dominate. Therefore, we compare error measures across each level of aggregation, before providing the error measures across all the series in the bottom-row. Notice, that the RMSE increases as we go from the bottom level to the aggregate levels above.

The following code generates the accuracy measures for the aggregate series shown in the first row of the table. Similar code is used to evaluate forecasts for other levels.

```
fc |>
  filter(is_aggregated(State), is_aggregated(Purpose)) |>
```

```
accuracy(
  data = tourism_full,
  measures = list(rmse = RMSE, mase = MASE)
) |>
group_by(.model) |>
summarise(rmse = mean(rmse), mase = mean(mase))
```

```
## # A tibble: 4 x 3
##   .model  rmse  mase
##   <chr>  <dbl> <dbl>
## 1 base   1721.  1.53
## 2 bu     3070.  3.16
## 3 mint   2158.  2.09
## 4 ols    1804.  1.63
```

```
#> # A tibble: 4 × 3
#>   .model   rmse  mase
#>   <chr>   <dbl> <dbl>
#> 1 base    1721.  1.53
#> 2 bu      3071.  3.17
#> 3 mint    2158.  2.09
#> 4 ols     1804.  1.63
```

Reconciling the base forecasts using OLS and MinT results in more accurate forecasts compared to the bottom-up approach. This result is commonly observed in applications as reconciliation approaches use information from all levels of the structure, resulting in more accurate coherent forecasts compared to the older traditional methods which use limited information. Furthermore, reconciliation usually improves the incoherent base forecasts for almost all levels.

11.5 Reconciled distributional forecasts

So far we have only discussed the reconciliation of point forecasts. However, we are usually also interested in the forecast distributions so that we can compute prediction intervals.

Panagiotelis et al. (2023) present several important results for generating reconciled probabilistic forecasts. We focus here on two fundamental results that are implemented in the reconcile() function.

11.6 Forecasting Australian prison population

Returning to the Australian prison population data (Section 11.1), we will compare the forecasts from bottom-up and MinT methods applied to base ETS models, using a test set comprising the final two years or eight quarters 2015Q1–2016Q4 of the available data.

```
fit <- prison_gts |>
  filter(year(Quarter) <= 2014) |>
  model(base = ETS(Count)) |>
  reconcile(
    bottom_up = bottom_up(base),
    MinT = min_trace(base, method = "mint_shrink")
  )
fc <- fit |> forecast(h = 8)
fc |>
  filter(is_aggregated(State), is_aggregated(Gender),
         is_aggregated(Legal)) |>
  autoplot(prison_gts, alpha = 0.7, level = 90) +
```

1. If the base forecasts are normally distributed, i.e.,

$$\hat{\boldsymbol{y}}_h \sim N(\hat{\boldsymbol{\mu}}_h, \hat{\boldsymbol{\Sigma}}_h),$$

then the reconciled forecasts are also normally distributed,

$$\tilde{\boldsymbol{y}}_h \sim N(\boldsymbol{SG}\hat{\boldsymbol{\mu}}_h, \boldsymbol{SG}\hat{\boldsymbol{\Sigma}}_h\boldsymbol{G'S'}).$$

2. If it is unreasonable to assume normality for the base forecasts, we can use bootstrapping. Bootstrapped prediction intervals were introduced in Section 5.5. The same idea can be used here. We can simulate future sample paths from the model(s) that produce the base forecasts, and then reconcile these sample paths. Coherent prediction intervals can be computed from the reconciled sample paths.

   Suppose that $(\hat{\boldsymbol{y}}_h^{[1]}, \dots, \hat{\boldsymbol{y}}_h^{[B]})$ are a set of B simulated sample paths, generated independently from the models used to produce the base forecasts. Then $(\boldsymbol{SG}\hat{\boldsymbol{y}}_h^{[1]}, \dots, \boldsymbol{SG}\hat{\boldsymbol{y}}_h^{[B]})$ provides a set of reconciled sample paths, from which percentiles can be calculated in order to construct coherent prediction intervals.

   To generate bootstrapped prediction intervals in this way, we simply set `bootstrap = TRUE` in the `forecast()` function.

Figure 17: Figure..

```
labs(y = "Number of prisoners ('000)",
     title = "Australian prison population (total)")
```


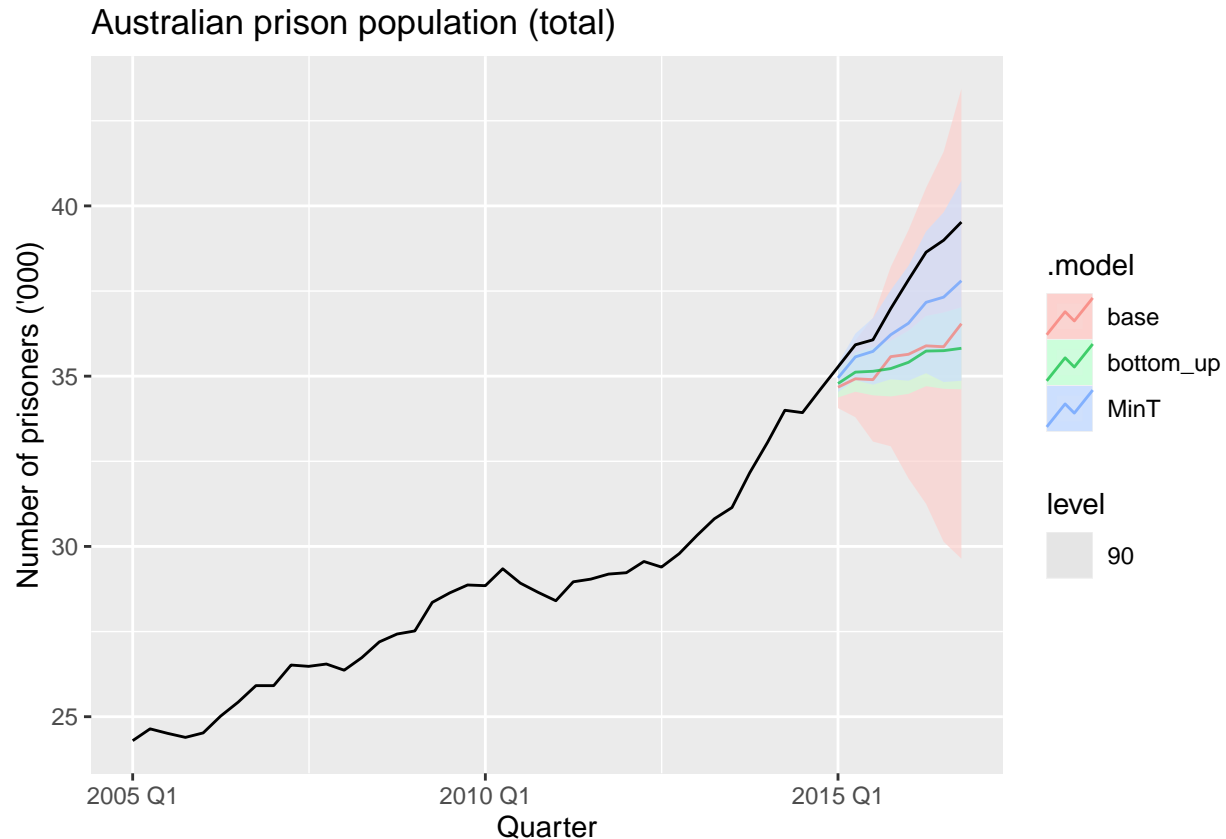Australian prison population (total)

Figure 11.14 shows the three sets of forecasts for the aggregate Australian prison population. The base and bottom-up forecasts from the ETS models seem to underestimate the trend over the test period. The MinT approach combines information from all the base forecasts in the aggregation structure; in this case, the base forecasts at the top level are adjusted upwards.

The MinT reconciled prediction intervals are much tighter than the base forecasts, due to MinT being based on an estimator that minimizes variances. The base forecast distributions are also incoherent, and therefore carry with them the extra uncertainty of the incoherency error.

We exclude the bottom-up forecasts from the remaining plots in order to simplify the visual exploration. However, we do revisit their accuracy in the evaluation results presented later.

Figures 11.15–11.17 show the MinT and base forecasts at various levels of aggregation. To make it easier to see the effect, we only show the last five years of training data. In general, MinT adjusts the base forecasts in the direction of the test set, hence improving the forecast accuracy. There is no guarantee that MinT reconciled forecasts will be more accurate than the base forecasts for every series, but they will be more accurate on average (see Panagiotelis et al., 2021).

```
fc |>
  filter(
    .model %in% c("base", "MinT"),
    !is_aggregated(State), is_aggregated(Legal),
    is_aggregated(Gender)
  ) |>
  autoplot(
```

```
        prison_gts |> filter(year(Quarter) >= 2010),
        alpha = 0.7, level = 90
    ) +
    labs(title = "Prison population (by state)",
        y = "Number of prisoners ('000)") +
    facet_wrap(vars(State), scales = "free_y", ncol = 4) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```
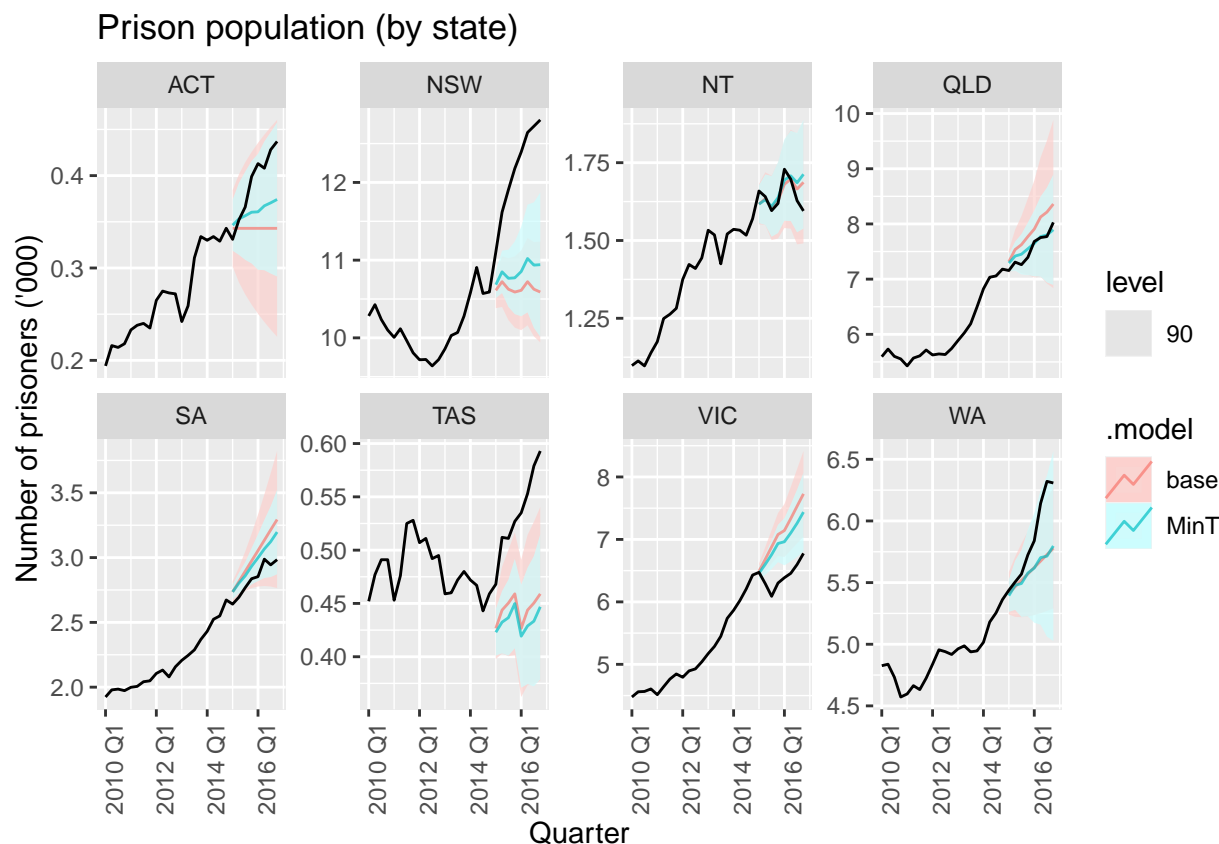

Prison population (by state)

Figure 11.15 shows forecasts for each of the eight states. There is a general upward trend during the test set period across all the states. However, there appears to be a relatively large and sudden surge in New South Wales and Tasmania, which means the test set observations are well outside the upper bound of the forecast intervals for both these states. Because New South Wales is the state with the largest prison population, this surge will have a substantial impact on the total. In contrast, Victoria shows a substantial dip in 2015Q2–2015Q3, before returning to an upward trend. This dip is not captured in any of the Victorian forecasts.

Figure 11.17 shows the forecasts for some selected bottom-level series of the Australian prison population. The four largest states are represented across the columns, with legal status and gender down the rows. These allow for some interesting analysis and observations that have policy implications. The large increase observed across the states during the 2015Q1–2016Q4 test period appears to be driven by large increases in the remand prison population. These increases seem to be generally missed by both forecasts. In contrast to the other states, for New South Wales there is also a substantial increase in the sentenced prison population. In particular, the increase in numbers of sentenced males in NSW contributes substantially to the rise in state and national prison numbers.

Using the accuracy() function, we evaluate the forecast accuracy across the grouped structure. The code below evaluates the forecast accuracy for only the top-level national aggregate of the Australian prison

population time series. Similar code is used for the rest of the results shown in Table 11.3.

```
fc |>
  filter(is_aggregated(State), is_aggregated(Gender),
         is_aggregated(Legal)) |>
  accuracy(data = prison_gts,
           measures = list(mase = MASE,
                           ss = skill_score(CRPS)
                           )
          ) |>
  group_by(.model) |>
  summarise(mase = mean(mase), sspc = mean(ss) * 100)
```

```
## # A tibble: 3 x 3
##   .model      mase  sspc
##   <chr>      <dbl> <dbl>
## 1 MinT       0.895  76.8
## 2 base       1.72   55.9
## 3 bottom_up  1.84   33.5
```

```
#> # A tibble: 3 × 3
#>   .model      mase  sspc
#>   <chr>      <dbl> <dbl>
#> 1 MinT       0.895  76.8
#> 2 base       1.72   55.9
#> 3 bottom_up  1.84   33.5
```

We use scaled measures because the numbers of prisoners vary substantially across the groups. The MASE gives a scaled measure of point-forecast accuracy (see Section 5.8), while the CRPS skill score gives a scaled measure of distributional forecast accuracy (see Section 5.9). A low value of MASE indicates a good forecast, while a high value of the skill score indicates a good forecast.

The results show that the MinT reconciled forecasts improve on the accuracy of the base forecasts and are also more accurate than the bottom-up forecasts. As the MinT optimal reconciliation approach uses information from all levels in the structure, it generates more accurate forecasts than the traditional approaches (such as bottom-up) which use limited information.