

Time Series Analysis & Forecasting Using R

bit.ly/fable2023

10. Forecast reconciliation



Outline

- 1 Hierarchical and grouped time series
- 2 Forecast reconciliation
- 3 Example: Australian tourism
- 4 Lab Session 20

Outline

1 Hierarchical and grouped time series

2 Forecast reconciliation

3 Example: Australian tourism

4 Lab Session 20

Australian Pharmaceutical Benefits Scheme



PBS sales

PBS

```
# A tibble: 67,596 x 9 [1M]
```

```
# Key:      Concession, Type, ATC1, ATC2 [336]
```

	Month	Concession	Type	ATC1	ATC1_desc	ATC2	ATC2_desc	Scripts
	<mth>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>
1	1991 Jul	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	18228
2	1991 Aug	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	15327
3	1991 Sep	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	14775
4	1991 Oct	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	15380
5	1991 Nov	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	14371
6	1991 Dec	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	15028
7	1992 Jan	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	11040
8	1992 Feb	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	15165
9	1992 Mar	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	16898
10	1992 Apr	Concessional	Co-pa~	A	Alimenta~	A01	STOMATOL~	18141

```
# i 67,586 more rows
```

```
# i 1 more variable: Cost <dbl>
```

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

ATC drug classification

ATC1: 14 classes

A

Alimentary tract and metabolism

ATC2: 84 classes

A10

Drugs used in diabetes

A10B

Blood glucose lowering drugs

A10BA

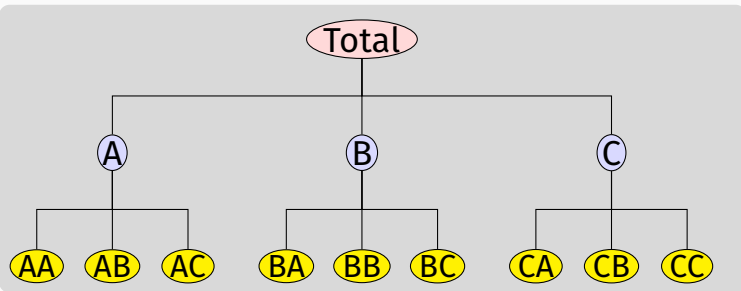
Biguanides

A10BA02

Metformin

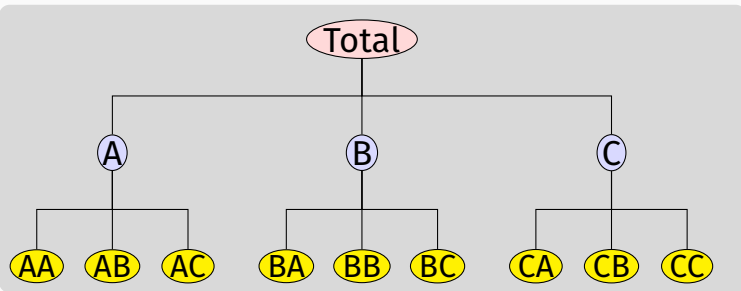
Hierarchical time series

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



Hierarchical time series

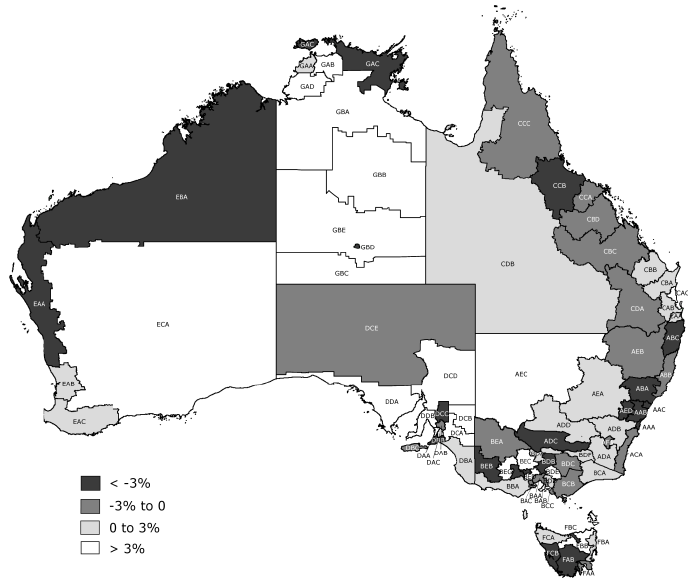
A **hierarchical time series** is a collection of several time series that are linked together in a hierarchical structure.



Examples

- PBS sales by ATC groups
- Tourism demand by states, zones, regions

Australian tourism



Australian tourism

tourism

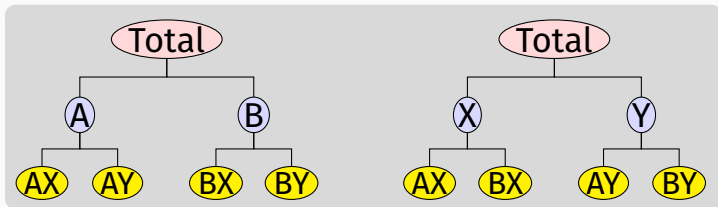
```
# A tsibble: 24,320 x 5 [1Q]
# Key:           Region, State, Purpose [304]
   Quarter Region   State Purpose   Trips
   <qtr> <chr>      <chr> <chr>      <dbl>
1 1998 Q1 Adelaide SA      Business 135.
2 1998 Q2 Adelaide SA      Business 110.
3 1998 Q3 Adelaide SA      Business 166.
4 1998 Q4 Adelaide SA      Business 127.
5 1999 Q1 Adelaide SA      Business 137.
6 1999 Q2 Adelaide SA      Business 200.
7 1999 Q3 Adelaide SA      Business 169.
8 1999 Q4 Adelaide SA      Business 134.
9 2000 Q1 Adelaide SA      Business 154.
10 2000 Q2 Adelaide SA      Business 169.
```

Australian tourism

- Quarterly data on visitor night from 1998:Q1 – 2013:Q4
- From: *National Visitor Survey*, based on annual interviews of 120,000 Australians aged 15+, collected by Tourism Research Australia.
- Split by 7 states, 27 zones and 76 regions (a geographical hierarchy)
- Also split by purpose of travel
 - ▶ Holiday
 - ▶ Visiting friends and relatives (VFR)
 - ▶ Business
 - ▶ Other
- 304 bottom-level series

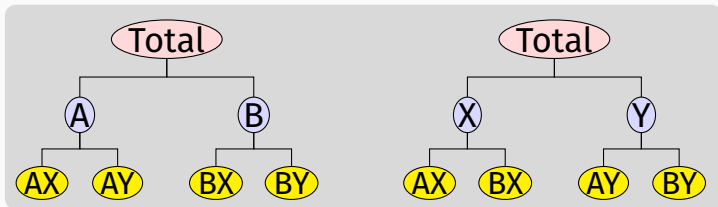
Grouped time series

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

Creating aggregates

```
PBS |>  
  aggregate_key(ATC1 / ATC2, Scripts = sum(Scripts)) |>  
  filter(Month == yearmonth("1991 Jul")) |>  
  print(n = 18)
```

```
# A tsibble: 98 x 4 [1M]  
# Key:      ATC1, ATC2 [98]  
  Month ATC1      ATC2      Scripts  
  <mt> <chr*>    <chr*>    <dbl>  
1 1991 Jul <aggregated> <aggregated> 8090395  
2 1991 Jul A      <aggregated> 799025  
3 1991 Jul B      <aggregated> 109227  
4 1991 Jul C      <aggregated> 1794995  
5 1991 Jul D      <aggregated> 299779  
6 1991 Jul G      <aggregated> 300931  
7 1991 Jul H      <aggregated> 112114  
8 1991 Jul J      <aggregated> 1151681  
9 1991 Jul L      <aggregated> 24580  
10 1991 Jul M      <aggregated> 562956  
11 1991 Jul N      <aggregated> 1546023  
12 1991 Jul P      <aggregated> 47661  
13 1991 Jul R      <aggregated> 859273  
14 1991 Jul S      <aggregated> 391639  
15 1991 Jul V      <aggregated> 38705  
16 1991 Jul Z      <aggregated> 51806
```

Creating aggregates

```
tourism |>
  aggregate_key(Purpose * (State / Region), Trips = sum(Trips)) |>
  filter(Quarter == yearquarter("1998 Q1")) |>
  print(n = 15)
```

```
# A tsibble: 425 x 5 [1Q]
```

```
# Key:      Purpose, State, Region [425]
```

	Quarter	Purpose	State	Region	Trips
	<qtr>	<chr*>	<chr*>	<chr*>	<dbl>
1	1998 Q1	<aggregated>	<aggregated>	<aggregated>	23182.
2	1998 Q1	Business	<aggregated>	<aggregated>	3599.
3	1998 Q1	Holiday	<aggregated>	<aggregated>	11806.
4	1998 Q1	Other	<aggregated>	<aggregated>	680.
5	1998 Q1	Visiting	<aggregated>	<aggregated>	7098.
6	1998 Q1	<aggregated>	ACT	<aggregated>	551.
7	1998 Q1	<aggregated>	NSW	<aggregated>	8040.
8	1998 Q1	<aggregated>	NT	<aggregated>	181.
9	1998 Q1	<aggregated>	QLD	<aggregated>	4041.
10	1998 Q1	<aggregated>	SA	<aggregated>	1735.
11	1998 Q1	<aggregated>	TAS	<aggregated>	982.
12	1998 Q1	<aggregated>	VIC	<aggregated>	6010.
13	1998 Q1	<aggregated>	WA	<aggregated>	1641.
14	1998 Q1	<aggregated>	ACT	Canberra	551.

Creating aggregates

- Similar to `summarise()` but using the key structure
- A grouped structure is specified using `grp1 * grp2`
- A nested structure is specified via `parent / child`.
- Groups and nesting can be mixed:

```
(country/region/city) * (brand/product)
```
- All possible aggregates are produced.
- These are useful when forecasting at different levels of aggregation.

Outline

1 Hierarchical and grouped time series

2 Forecast reconciliation

3 Example: Australian tourism

4 Lab Session 20

The problem

- 1 How to forecast time series at all nodes such that the forecasts add up in the same way as the original data?
- 2 Can we exploit relationships between the series to improve the forecasts?

The problem

- 1 How to forecast time series at all nodes such that the forecasts add up in the same way as the original data?
- 2 Can we exploit relationships between the series to improve the forecasts?

The solution

- 1 Forecast all series at all levels of aggregation using an automatic forecasting algorithm.
(e.g., ETS, ARIMA, ...)
- 2 Reconcile the resulting forecasts so they add up correctly using least squares optimization (i.e., find closest reconciled forecasts to the original forecasts).
- 3 This is available using `reconcile()`.

Forecast reconciliation

```
tourism |>
  aggregate_key(Purpose * (State / Region), Trips = sum(Trips)) |>
  model(ets = ETS(Trips)) |>
  reconcile(ets_adjusted = min_trace(ets)) |>
  forecast(h = 2)
```

```
# A fable: 1,700 x 7 [1Q]
```

```
# Key:      Purpose, State, Region, .model [850]
```

	Purpose	State	Region	.model	Quarter	Trips	.mean
	<chr*>	<chr*>	<chr*>	<chr>	<qtr>	<dist>	<dbl>
1	Business	ACT	Canberra	ets	2018 Q1	N(144, 1119)	144.
2	Business	ACT	Canberra	ets	2018 Q2	N(203, 2260)	203.
3	Business	ACT	Canberra	ets_adjusted	2018 Q1	N(157, 539)	157.
4	Business	ACT	Canberra	ets_adjusted	2018 Q2	N(214, 951)	214.
5	Business	ACT	<aggregated>	ets	2018 Q1	N(144, 1119)	144.
6	Business	ACT	<aggregated>	ets	2018 Q2	N(203, 2260)	203.
7	Business	ACT	<aggregated>	ets_adjusted	2018 Q1	N(157, 539)	157.
8	Business	ACT	<aggregated>	ets_adjusted	2018 Q2	N(214, 951)	214.
9	Business	NSW	Blue Mountains	ets	2018 Q1	N(20, 140)	19.7

Hierarchical and grouped time series

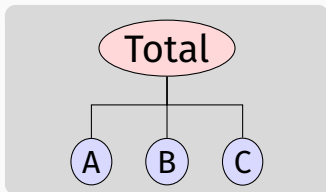
Every collection of time series with aggregation constraints can be written as

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

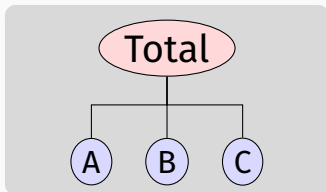
where

- \mathbf{y}_t is a vector of all series at time t
- \mathbf{b}_t is a vector of the most disaggregated series at time t
- \mathbf{S} is a “summing matrix” containing the aggregation constraints.

Hierarchical time series



Hierarchical time series

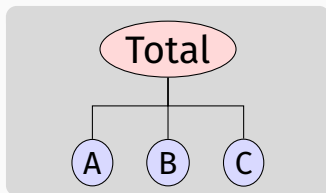


y_t : observed aggregate of all series at time t .

$y_{X,t}$: observation on series X at time t .

b_t : vector of all series at bottom level in time t .

Hierarchical time series



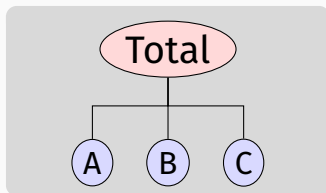
y_t : observed aggregate of all series at time t .

$y_{X,t}$: observation on series X at time t .

b_t : vector of all series at bottom level in time t .

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

Hierarchical time series



y_t : observed aggregate of all series at time t .

$y_{X,t}$: observation on series X at time t .

b_t : vector of all series at bottom level in time t .

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

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

Forecasting notation

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

Forecasting notation

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

Forecasting notation

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

(In general, they will not “add up”.)

Reconciled forecasts must be of the form:

$$\tilde{\mathbf{y}}_n(h) = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_n(h)$$

for some matrix \mathbf{G} .

Forecasting notation

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

Reconciled forecasts must be of the form:

$$\tilde{\mathbf{y}}_n(h) = \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_n(h)$$

for some matrix \mathbf{G} .

- \mathbf{G} extracts and combines base forecasts $\hat{\mathbf{y}}_n(h)$ to get bottom-level forecasts.

- \mathbf{S} adds them up

Optimal combination forecasts

Main result

The best (minimum sum of variances) unbiased forecasts are obtained when $\mathbf{G} = (\mathbf{S}'\Sigma_h^{-1}\mathbf{S})^{-1}\mathbf{S}'\Sigma_h^{-1}$, where Σ_h is the h -step base forecast error covariance matrix.

Optimal combination forecasts

Main result

The best (minimum sum of variances) unbiased forecasts are obtained when $\mathbf{G} = (\mathbf{S}'\Sigma_h^{-1}\mathbf{S})^{-1}\mathbf{S}'\Sigma_h^{-1}$, where Σ_h is the h -step base forecast error covariance matrix.

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

Problem: Σ_h hard to estimate, especially for $h > 1$.

Solutions:

- Ignore Σ_h (OLS) [`min_trace(method='ols')`]
- Assume $\Sigma_h = k_h \Sigma_1$ is diagonal (WLS)
[`min_trace(method='wls')`]
- Assume $\Sigma_h = k_h \Sigma_1$ and estimate it (GLS)

Features

- Covariates can be included in initial forecasts.
- Adjustments can be made to initial forecasts at any level.
- Very simple and flexible method. Can work with *any* hierarchical or grouped time series.
- Conceptually easy to implement: regression of base forecasts on structure matrix.

Outline

1 Hierarchical and grouped time series

2 Forecast reconciliation

3 Example: Australian tourism

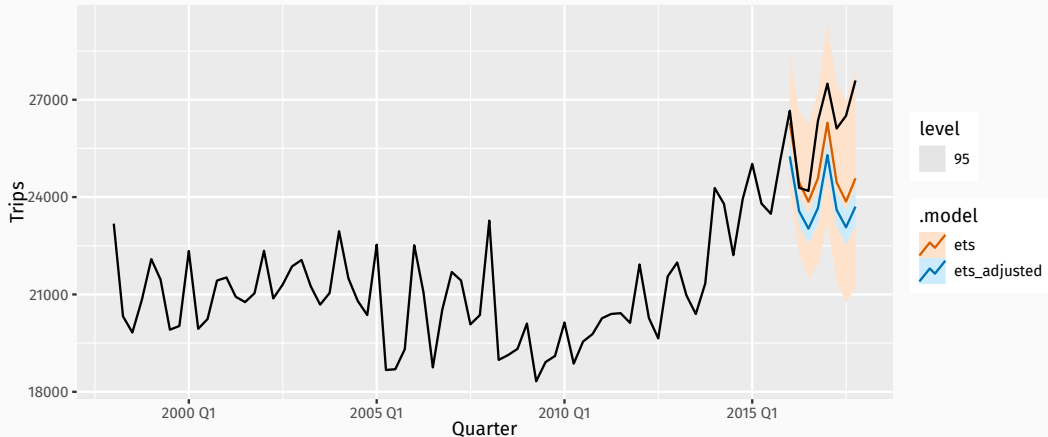
4 Lab Session 20

Example: Australian tourism

```
tourism_agg <- tourism |>
  aggregate_key(Purpose * (State / Region),
    Trips = sum(Trips)
  )
fc <- tourism_agg |>
  filter_index(. ~ "2015 Q4") |>
  model(ets = ETS(Trips)) |>
  reconcile(ets_adjusted = min_trace(ets)) |>
  forecast(h = "2 years")
```

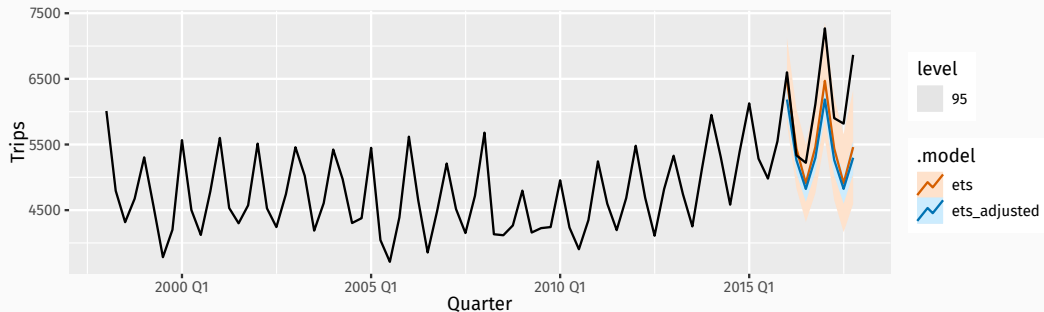
Example: Australian tourism

```
fc |>  
  filter(is_aggregated(Purpose) & is_aggregated(State)) |>  
  autoplot(tourism_agg, level = 95)
```



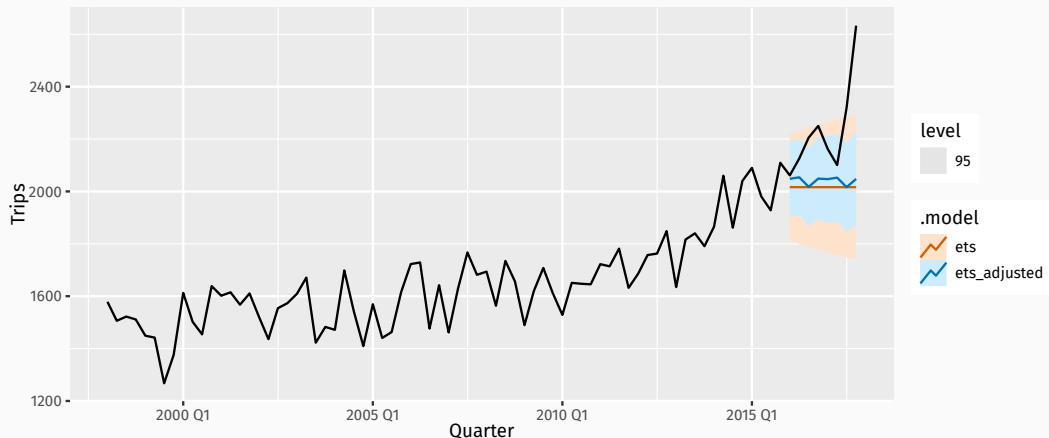
Example: Australian tourism

```
fc |>  
  filter(is_aggregated(Purpose) & State == "VIC" &  
         is_aggregated(Region)) |>  
  autoplot(tourism_agg, level = 95)
```



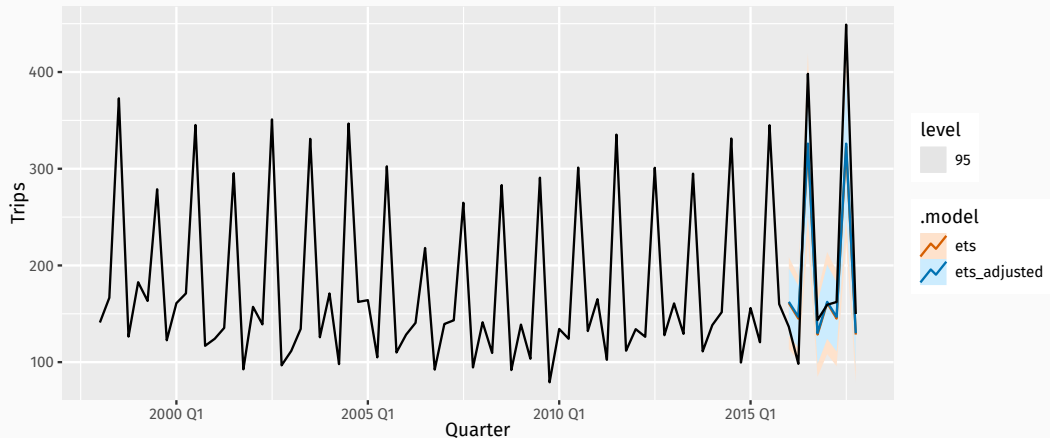
Example: Australian tourism

```
fc |>  
  filter(is_aggregated(Purpose) & Region == "Melbourne") |>  
  autoplot(tourism_agg, level = 95)
```



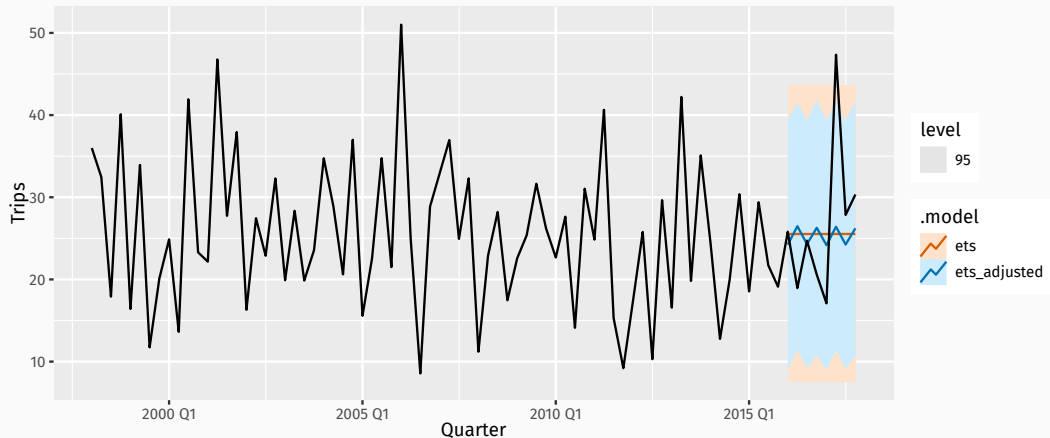
Example: Australian tourism

```
fc |>  
  filter(is_aggregated(Purpose) & Region == "Snowy Mountains") |>  
  autoplot(tourism_agg, level = 95)
```



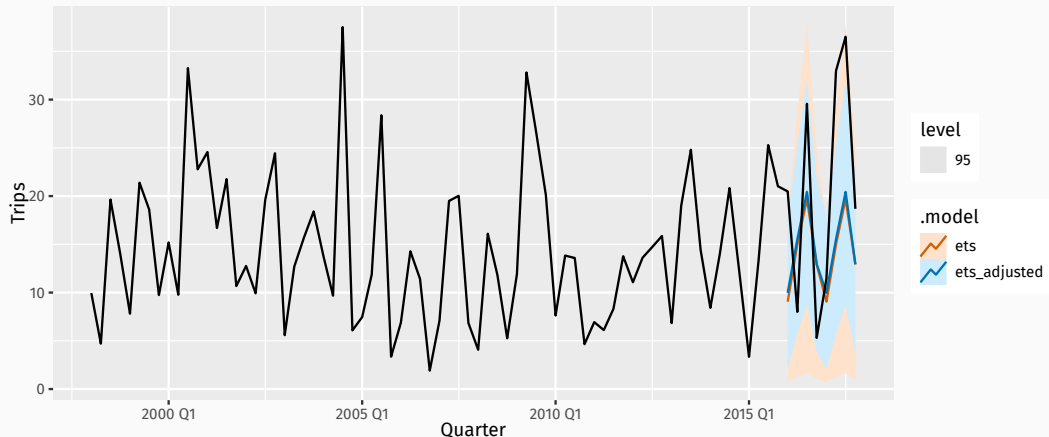
Example: Australian tourism

```
fc |>  
  filter(Purpose == "Holiday" & Region == "Barossa") |>  
  autoplot(tourism_agg, level = 95)
```



Example: Australian tourism

```
fc |>  
  filter(is_aggregated(Purpose) & Region == "MacDonnell") |>  
  autoplot(tourism_agg, level = 95)
```



Example: Australian tourism

```
fc <- tourism_agg |>
  filter_index(. ~ "2015 Q4") |>
  model(
    ets = ETS(Trips),
    arima = ARIMA(Trips)
  ) |>
  mutate(
    comb = (ets + arima) / 2
  ) |>
  reconcile(
    ets_adj = min_trace(ets),
    arima_adj = min_trace(arima),
    comb_adj = min_trace(comb)
  ) |>
  forecast(h = "2 years")
```

Forecast evaluation

```
fc |> accuracy(tourism_agg)
```

```
# A tibble: 2,550 x 13
```

	.model	Purpose	State	Region	.type	ME	RMSE	MAE	MPE	MAPE		
	<chr>	<chr*>	<chr*>	<chr*>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	arma	Business	ACT	Canberra	~ Test	35.9	45.7	35.9	16.9	16.9	0	
2	arma	Business	ACT	<aggregated>	Test	35.9	45.7	35.9	16.9	16.9	0	
3	arma	Business	NSW	Blue Mountains	~ Test	1.93	10.6	8.52	-			
18.0	48.6	0.644										
4	arma	Business	NSW	Capital Country	~ Test	8.08	15.6	10.4	11.8	19.0	0	
5	arma	Business	NSW	Central Coast	~ Test	10.0	14.5	10.8	26.9	32.2	0	
6	arma	Business	NSW	Central NSW	~ Test	17.7	31.9	28.2	12.0	24.1	1	
7	arma	Business	NSW	Hunter	~ Test	35.3	43.9	35.3	24.2	24.2	1	
8	arma	Business	NSW	New England North W	~ Test	23.1	31.8	26.8	19.5	28.0	1	
9	arma	Business	NSW	North Coast NSW	~ Test	24.8	40.1	36.8	11.5	28.5	1	
10	arma	Business	NSW	Outback NSW	~ Test	6.87	11.0	7.76	13.7	16.5	0	

```
# i 2,540 more rows
```

Forecast evaluation

```
fc |>  
  accuracy(tourism_agg) |>  
  group_by(.model) |>  
  summarise(MASE = mean(MASE)) |>  
  arrange(MASE)
```

```
# A tibble: 6 x 2  
  .model      MASE  
  <chr>      <dbl>  
1 ets_adj    1.02  
2 comb_adj   1.02  
3 ets        1.04  
4 comb       1.04  
5 arima_adj  1.07  
6 arima      1.09
```

Outline

1 Hierarchical and grouped time series

2 Forecast reconciliation

3 Example: Australian tourism

4 Lab Session 20

Lab Session 20

- Prepare aggregations of the PBS data by Concession, Type, and ATC1.
- Use forecast reconciliation with the PBS data, using ETS, ARIMA and SNAIVE models, applied to all but the last 3 years of data.
- Which type of model works best?
- Does the reconciliation improve the forecast accuracy?
- Why doesn't the reconciliation make any difference to the SNAIVE forecasts?

Feedback form

bit.ly/fable2022feedback