MONASH University

MONASH BUSINESS SCHOOL

High dimensional time series analysis

tsibble feasts Fable

5. Forecast reconciliation

robjhyndman.com/hdtsa

Outline

Outline

1. Hierarchical and grouped time series
2. Forecast reconciliation
3. Example: Australian tourism
4. Lab Session 10

Australian Pharmaceutical Benefits Scheme

PBS sales

```
PBS
```

```
## # A tsibble: 65,219 x 9 [1M]
## # Key:       Concession, Type, ATC1, ATC2 [336]
##         Month Concession Type  ATC1  ATC1_desc ATC2
##         <mth> <chr>      <chr> <chr> <chr>     <chr>
##  1   1991 Jul Concessio~ Co-p~ A     Alimenta~ A01
##  2   1991 Aug Concessio~ Co-p~ A     Alimenta~ A01
##  3   1991 Sep Concessio~ Co-p~ A     Alimenta~ A01
##  4   1991 Oct Concessio~ Co-p~ A     Alimenta~ A01
##  5   1991 Nov Concessio~ Co-p~ A     Alimenta~ A01
##  6   1991 Dec Concessio~ Co-p~ A     Alimenta~ A01
##  7   1992 Jan Concessio~ Co-p~ A     Alimenta~ A01
##  8   1992 Feb Concessio~ Co-p~ A     Alimenta~ A01
##  9   1992 Mar Concessio~ Co-p~ A     Alimenta~ A01
## 10   1992 Apr Concessio~ Co-p~ A     Alimenta~ A01
## # ... with 65,209 more rows, and 3 more variables:
## #   ATC2_desc <chr>, Scripts <dbl>, Cost <dbl>
```

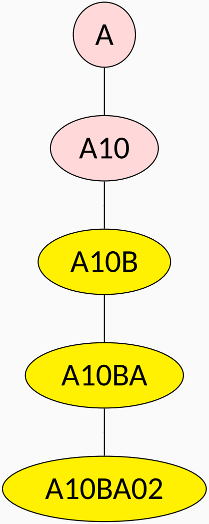
ATC drug classification

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

ATC2: 84 classes



Alimentary tract and metabolism

Drugs used in diabetes

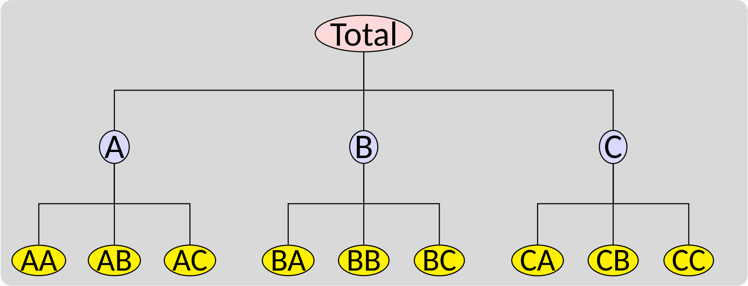
Blood glucose lowering drugs

Biguanides

Metformin

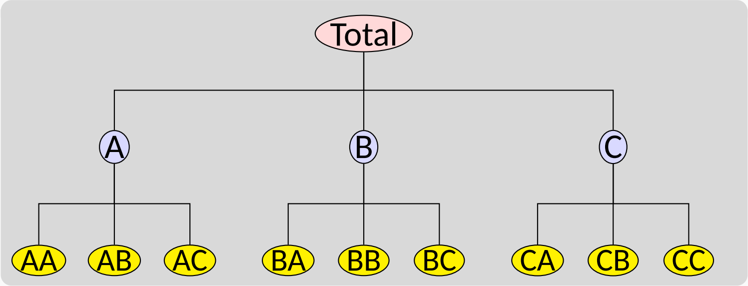
Hierarchical time series

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



Hierarchical time series

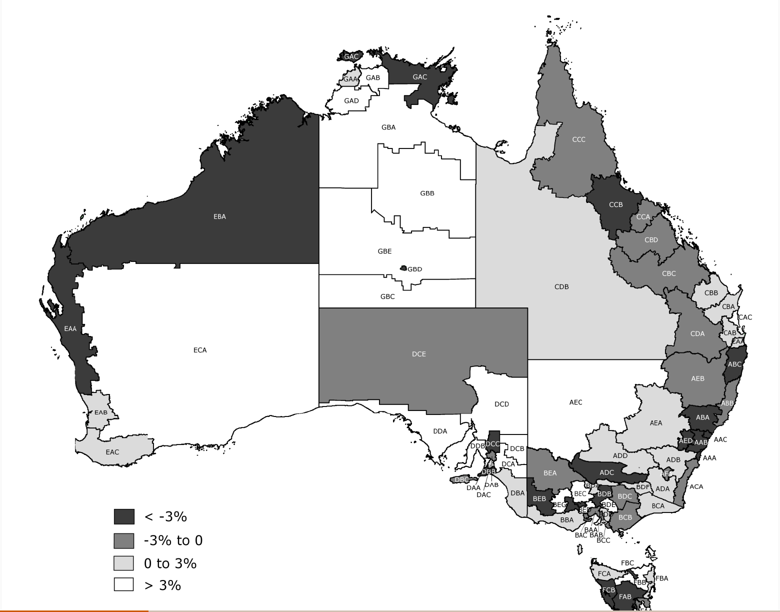
A **hierarchical time series** is a collection of several time series that are linked together in a hierarchical structure.



Examples

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

Australian tourism



Australian tourism

```
tourism
```

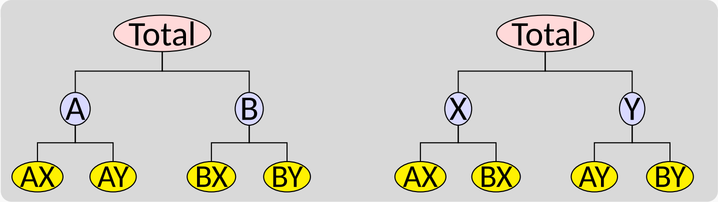
```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:       Region, State, Purpose [304]
##    Quarter Region   State           Purpose  Trips
##      <qtr> <chr>    <chr>           <chr>    <dbl>
##  1 1998 Q1 Adelaide South Australia Business  135.
##  2 1998 Q2 Adelaide South Australia Business  110.
## 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

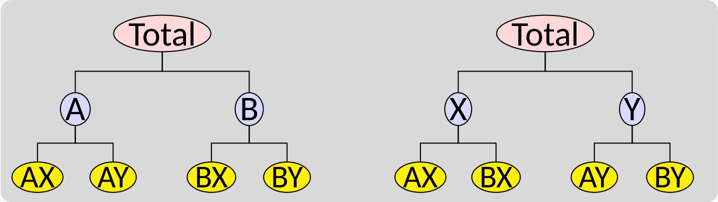
Grouped time series

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

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

```
## # A tsibble: 98 x 4 [?]
## # Key:       ATC1, ATC2 [98]
##    ATC1         ATC2            Month Scripts
##    <chr>        <chr>           <mth>   <dbl>
##  1 <aggregated> <aggregated> 1991 Jul 8090395
##  2 A            <aggregated> 1991 Jul  799025
##  3 B            <aggregated> 1991 Jul  109227
##  4 C            <aggregated> 1991 Jul 1794995
##  5 D            <aggregated> 1991 Jul  299779
##  6 G            <aggregated> 1991 Jul  300931
##  7 H            <aggregated> 1991 Jul  112114
##  8 J            <aggregated> 1991 Jul 1151681
##  9 L            <aggregated> 1991 Jul   24580
## 10 M            <aggregated> 1991 Jul  562956
## 11 N            <aggregated> 1991 Jul 1546023
## 12 P            <aggregated> 1991 Jul   47661
## 13 R            <aggregated> 1991 Jul  859273
## 14 S            <aggregated> 1991 Jul  391639
## 15 V            <aggregated> 1991 Jul   38705
## 16 Z            <aggregated> 1991 Jul   51806
## 17 A            A01          1991 Jul   22615
## 18 A            A02          1991 Jul  299251
## # ... with 80 more rows
```

Creating aggregates

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

```
## # A tsibble: 425 x 5 [?]
## # Key:       Purpose, State, Region [425]
##    Purpose      State             Region         Quarter   Trips
##    <chr>        <chr>             <chr>            <qtr>   <dbl>
##  1 <aggregated> <aggregated>      <aggregated>   1998 Q1 23182.
##  2 Business     <aggregated>      <aggregated>   1998 Q1  3599.
##  3 Holiday      <aggregated>      <aggregated>   1998 Q1 11806.
##  4 Other        <aggregated>      <aggregated>   1998 Q1   680.
##  5 Visiting     <aggregated>      <aggregated>   1998 Q1  7098.
##  6 <aggregated> ACT             ~ <aggregated>   1998 Q1   551.
##  7 <aggregated> New South Wale~ <aggregated>   1998 Q1  8040.
##  8 <aggregated> Northern Terri~ <aggregated>   1998 Q1   181.
##  9 <aggregated> Queensland      ~ <aggregated>   1998 Q1  4041.
## 10 <aggregated> South Australi~ <aggregated>   1998 Q1  1735.
## 11 <aggregated> Tasmania        ~ <aggregated>   1998 Q1   982.
## 12 <aggregated> Victoria        ~ <aggregated>   1998 Q1  6010.
## 13 <aggregated> Western Austra~ <aggregated>   1998 Q1  1641.
## 14 <aggregated> ACT             ~ Canberra     ~ 1998 Q1   551.
## 15 <aggregated> New South Wale~ Blue Mounta~   1998 Q1   196.
## # ... with 410 more rows
```

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

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

```r
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
##    <chr>      <chr>      <chr>      <chr>         <qtr> <dbl>
##  1 Business   ACT      ~ Canberra ~ ets         2018 Q1 144.
##  2 Business   ACT      ~ Canberra ~ ets         2018 Q2 203.
##  3 Business   ACT      ~ <aggregat~ ets         2018 Q1 144.
##  4 Business   ACT      ~ <aggregat~ ets         2018 Q2 203.
##  5 Business   New South~ Blue Moun~ ets         2018 Q1  19.7
##  6 Business   New South~ Blue Moun~ ets         2018 Q2  19.7
##  7 Business   New South~ Capital C~ ets         2018 Q1  36.1
##  8 Business   New South~ Capital C~ ets         2018 Q2  36.1
```

Hierarchical and grouped time series

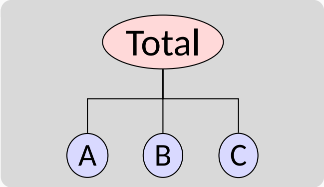
Every collection of time series with aggregation constraints can be written as

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

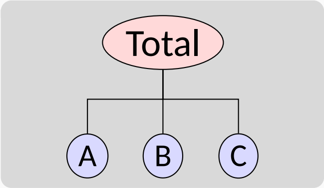
where

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

Hierarchical time series



Hierarchical time series

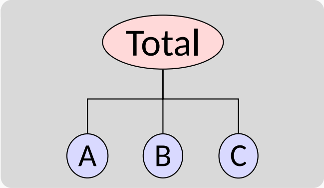


y_t : observed aggregate of all series at time t.

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

$\boldsymbol{b}_t$: vector of all series at bottom level in time t.

Hierarchical time series



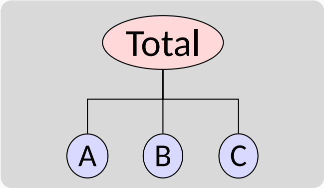
y_t : observed aggregate of all series at time t.

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

$\boldsymbol{b}_t$: vector of all series at bottom level in time t.

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

Hierarchical time series



y_t : observed aggregate of all series at time t.

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

$\boldsymbol{b}_t$: vector of all series at bottom level in time t.

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

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

Forecasting notation

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

Forecasting notation

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

Forecasting notation

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

Reconciled forecasts must be of the form:

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

for some matrix $\boldsymbol{G}$.

Forecasting notation

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

Reconciled forecasts must be of the form:

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

for some matrix $\boldsymbol{G}$.

- $\boldsymbol{G}$ extracts and combines base forecasts $\hat{\boldsymbol{y}}_n(h)$ to get bottom-level forecasts.
- $\boldsymbol{S}$ adds them up

Optimal combination forecasts

Main result

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

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

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) [`min_trace(method='shrink')` (the default)]

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.

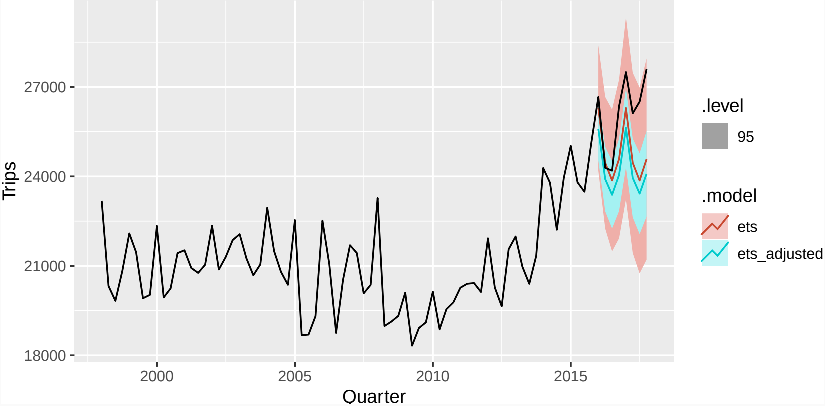
Outline

Example: Australian tourism

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

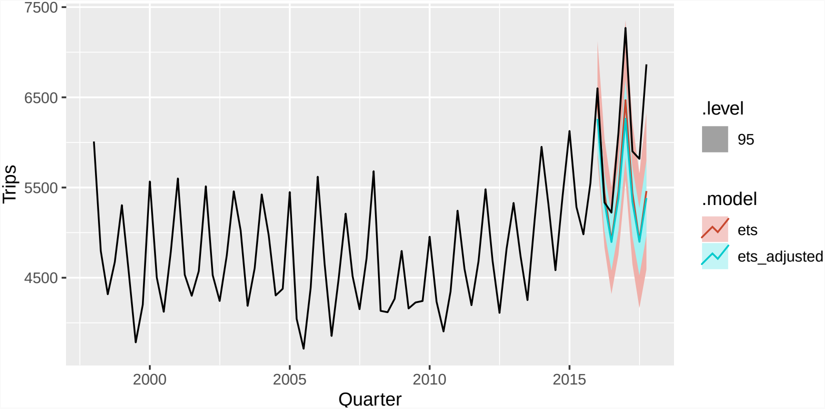
Example: Australian tourism

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



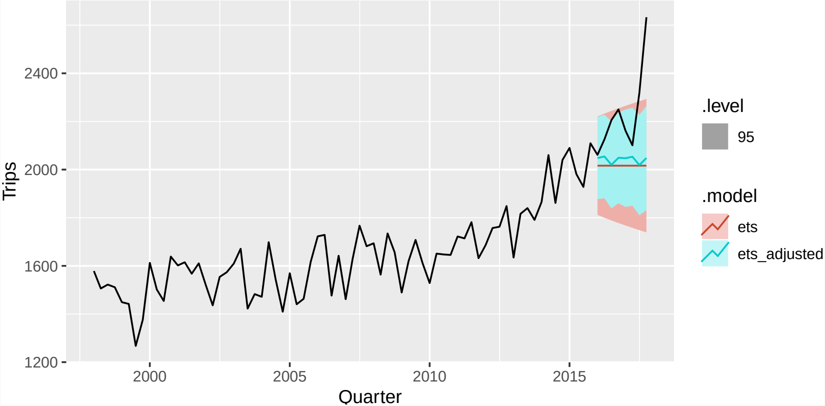
Example: Australian tourism

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



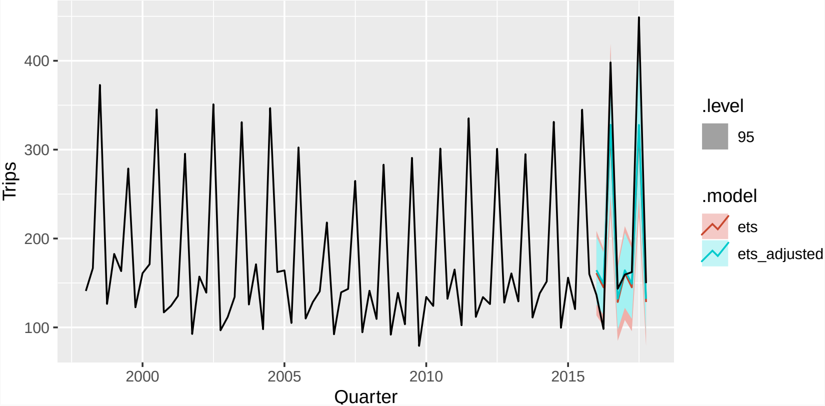
Example: Australian tourism

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



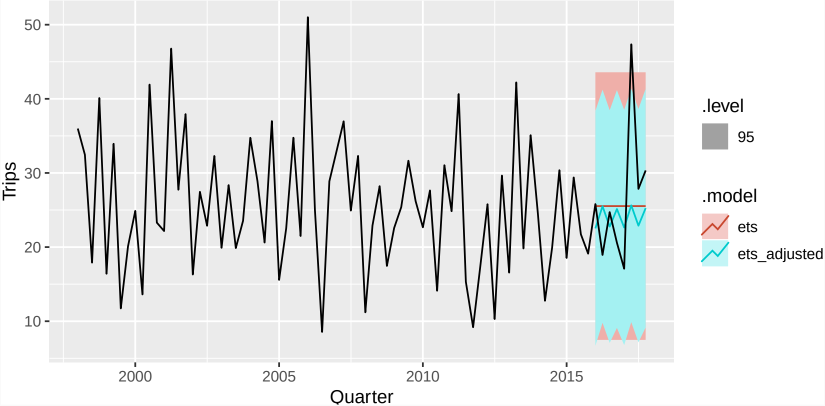
Example: Australian tourism

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



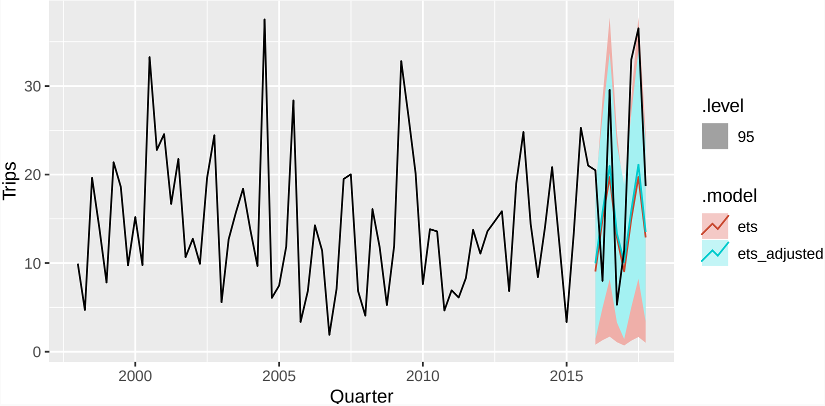
Example: Australian tourism

```
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

```r
fc <- tourism_agg %>%
  filter_index(. ~ yearquarter("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 12
##    .model Purpose    State      Region      .type    ME   RMSE
##    <chr>  <chr>      <chr>      <chr>       <chr> <dbl> <dbl>
##  1 arima  Business   ACT      ~ Canberra  ~ Test  35.9   45.7
##  2 arima  Business   ACT      ~ <aggregat~ Test  35.9   45.7
##  3 arima  Business   New South~ Blue Moun~ Test   1.93  10.6
##  4 arima  Business   New South~ Capital C~ Test   8.08  15.6
##  5 arima  Business   New South~ Central C~ Test  10.0   14.5
##  6 arima  Business   New South~ Central N~ Test  17.7   31.9
##  7 arima  Business   New South~ Hunter    ~ Test  35.3   43.9
##  8 arima  Business   New South~ New Engla~ Test  23.1   31.8
##  9 arima  Business   New South~ North Coa~ Test  24.8   40.1
## 10 arima  Business   New South~ Outback N~ Test   6.87  11.0
## # ... with 2,540 more rows, and 5 more variables:
## #   MAE <dbl>, MPE <dbl>, MAPE <dbl>, MASE <dbl>,
## #   ACF1 <dbl>
```

Forecast evaluation

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

```
## # A tibble: 6 x 2
##   .model      MASE
##   <chr>      <dbl>
## 1 comb_adj   0.981
## 2 ets_adj    0.984
## 3 arima_adj  1.01
## 4 ets        1.04
## 5 comb       1.04
## 6 arima      1.09
```

Outline

Lab Session 10

- 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 reconcililation make any difference to the SNAIVE forecasts?