Tidy Time Series & Forecasting in R

10. Forecast reconciliation

bit.ly/fable202

Outline

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

Outline

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  ATC2_desc Scripts
##       <mth> <chr>      <chr> <chr> <chr>     <chr> <chr>       <dbl>
##  1 1991 Jul Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   18228
##  2 1991 Aug Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   15327
##  3 1991 Sep Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   14775
##  4 1991 Oct Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   15380
##  5 1991 Nov Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   14371
##  6 1991 Dec Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   15028
##  7 1992 Jan Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   11040
##  8 1992 Feb Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   15165
##  9 1992 Mar Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   16898
## 10 1992 Apr Concessio~ Co-p~ A     Alimenta~ A01   STOMATOL~   18141
## # ... with 65,209 more rows, and 1 more variable: Cost <dbl>
```

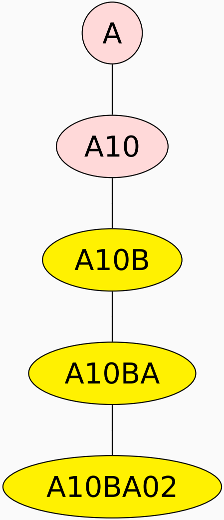
ATC drug classification

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

ATC2: 84 classes



Alimentary tract and metab

Drugs used in diabetes

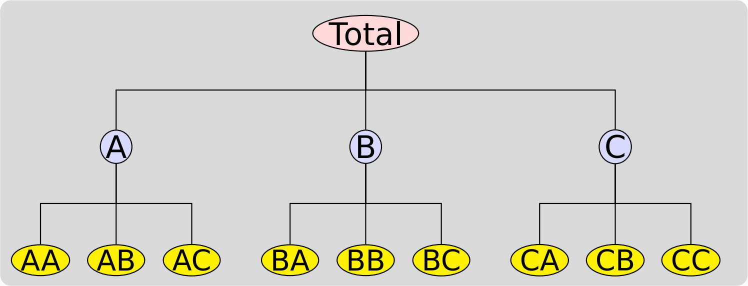
Blood glucose lowering dru

Biguanides

Metformin

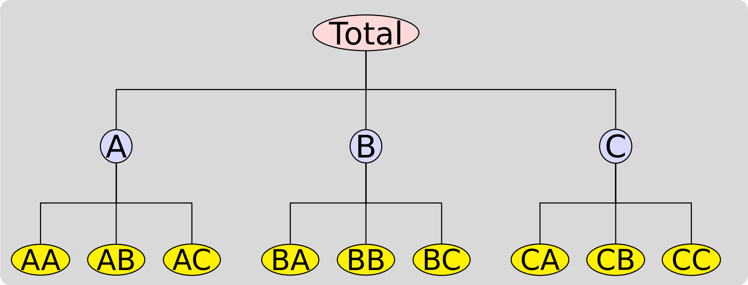
Hierarchical time series

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



Hierarchical time series

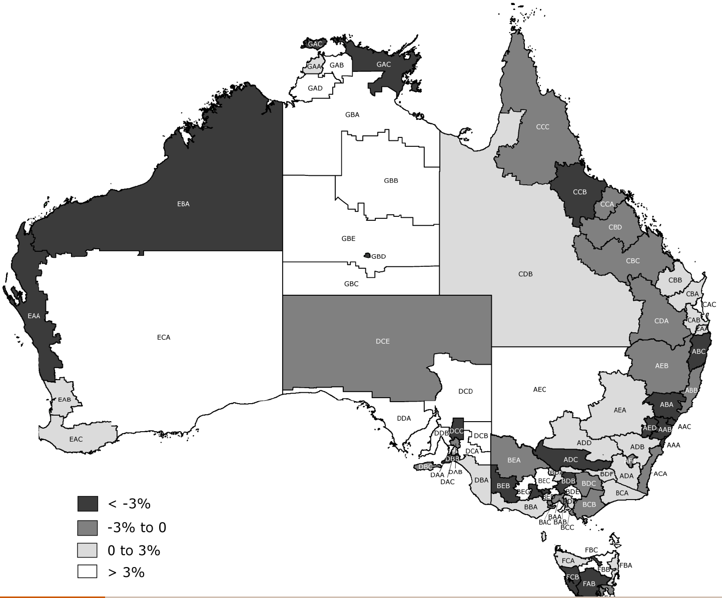
A **hierarchical time series** is a collection of several time series that are linked together in a hierarchical structure.



Examples

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

Australian tourism



Australian tourism

```
tourism
```

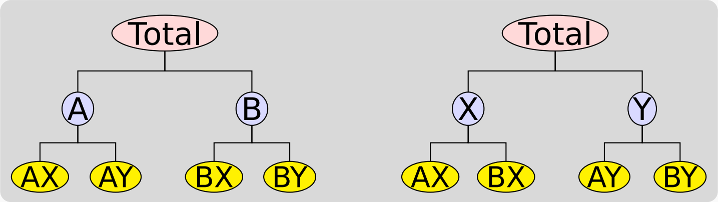
```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:       Region, State, Purpose [304]
##    Quarter Region   State           Purpose  Trips
##      <qtr> <chr>    <chr>           <chr>    <dbl>
##  1 1998 Q1 Adelaide South Australia Business  135.
##  2 1998 Q2 Adelaide South Australia Business  110.
##  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.
## # ... with 24,310 more rows
```

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

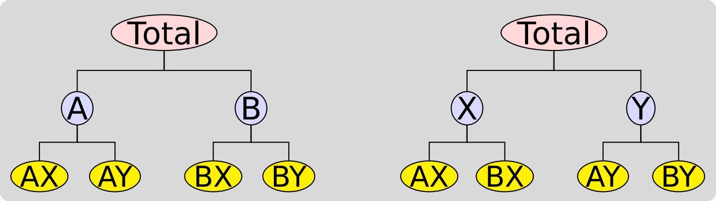
Grouped time series

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 [1M]
## # Key:       ATC1, ATC2 [98]
##        Month ATC1         ATC2         Scripts
##        <mth> <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
## 17 1991 Jul A            A01             22615
## 18 1991 Jul A            A02            299251
## # ... with 80 more rows
```

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> New South Wales~ <aggregated>  8040.
##  8 1998 Q1 <aggregated> Northern Territ~ <aggregated>   181.
##  9 1998 Q1 <aggregated> Queensland       ~ <aggregated>  4041.
## 10 1998 Q1 <aggregated> South Australia~ <aggregated>  1735.
## 11 1998 Q1 <aggregated> Tasmania         ~ <aggregated>   982.
## 12 1998 Q1 <aggregated> Victoria         ~ <aggregated>  6010.
## 13 1998 Q1 <aggregated> Western Austral~ <aggregated>  1641.
## 14 1998 Q1 <aggregated> ACT              ~ Canberra     ~   551.
## 15 1998 Q1 <aggregated> New South Wales~ Blue Mounta~   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
##    <chr>      <chr>       <chr>       <chr>    <qtr>
##  1 Business   <aggregat~ <aggregat~ ets     2018 Q1
##  2 Business   <aggregat~ <aggregat~ ets     2018 Q2
##  3 Business   <aggregat~ <aggregat~ ets_a~ 2018 Q1
##  4 Business   <aggregat~ <aggregat~ ets_a~ 2018 Q2
##  5 Business   South Aus~ <aggregat~ ets     2018 Q1
##  6 Business   South Aus~ <aggregat~ ets     2018 Q2
##  7 Business   South Aus~ <aggregat~ ets_a~ 2018 Q1
##  8 Business   South Aus~ <aggregat~ ets_a~ 2018 Q2
##  9 Business   Northern ~ <aggregat~ ets     2018 Q1
## 10 Business   Northern ~ <aggregat~ ets     2018 Q2
## # ... with 1,690 more rows, and 2 more variables:
```

Hierarchical and grouped time series

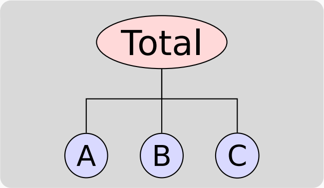
Every collection of time series with aggregation constraints can be written as

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

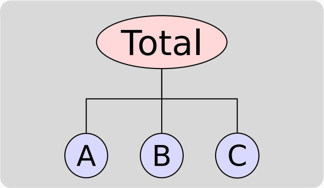
where

- $\boldsymbol{y}_t$ 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

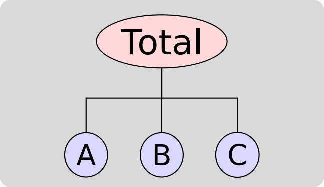


y_t : observed aggregate of all series at time t.

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

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

Hierarchical time series



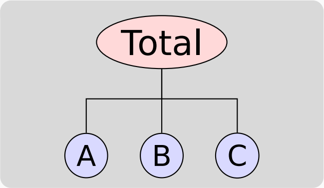
y_t : observed aggregate of all series at time t.

$y_{X,t}$: 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.

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

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

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)

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.

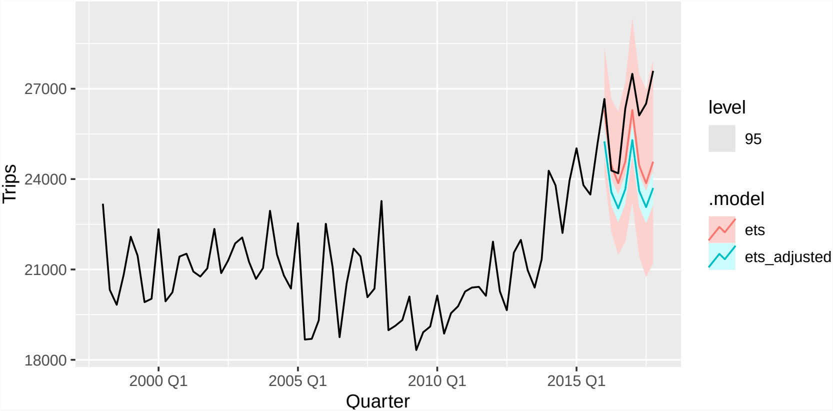
Outline

Example: Australian tourism

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

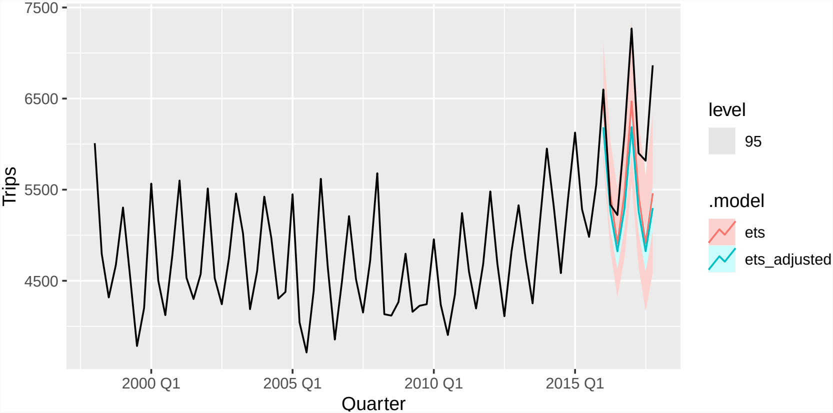
Example: Australian tourism

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



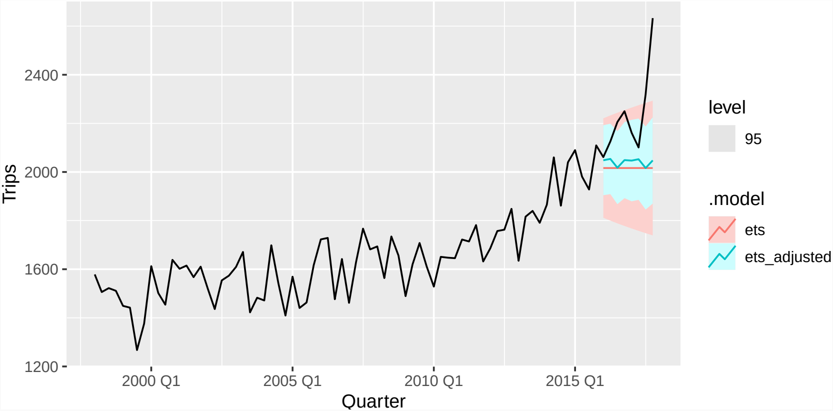
Example: Australian tourism

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



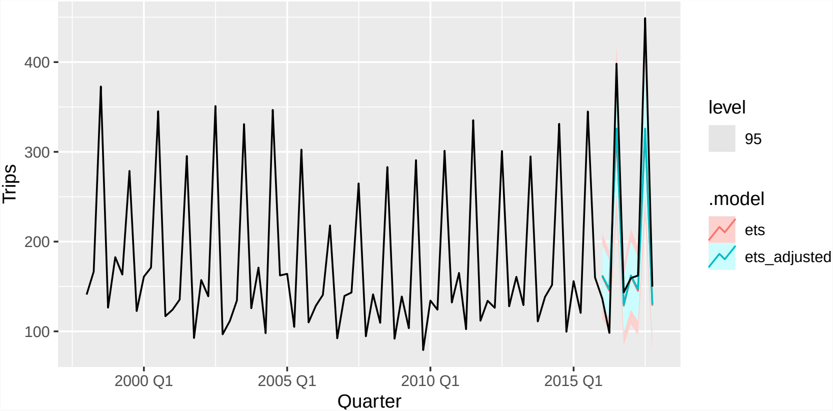
Example: Australian tourism

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



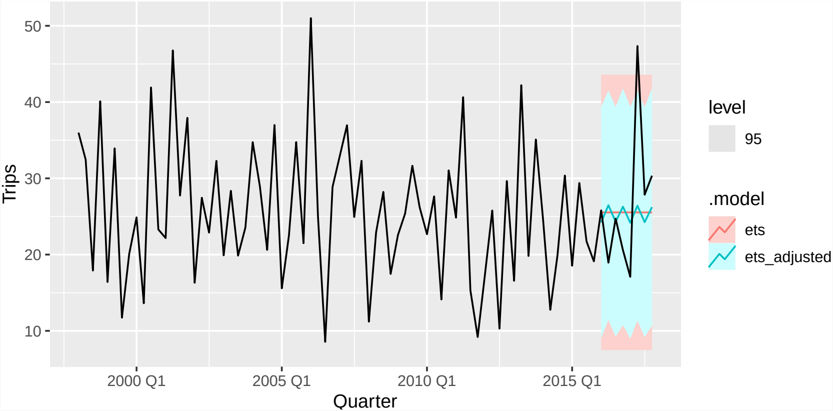
Example: Australian tourism

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



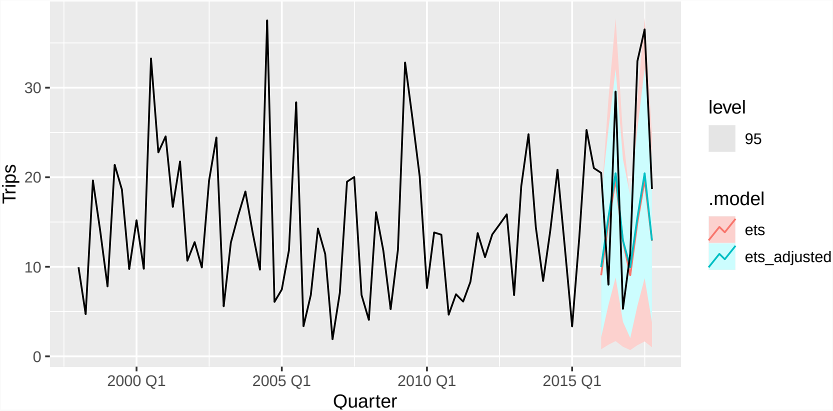
Example: Australian tourism

```
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

```r
fc %>% accuracy(tourism_agg)
```

```
## # A tibble: 2,550 x 12
##    .model Purpose  State        Region       .type     ME
##    <chr>  <chr>    <chr>        <chr>        <chr>  <dbl>
##  1 arima  Business <aggregat~ <aggregat~ Test   685.
##  2 arima  Business South Aus~ <aggregat~ Test    49.9
##  3 arima  Business Northern ~ <aggregat~ Test    22.2
##  4 arima  Business Western A~ <aggregat~ Test  -138.
##  5 arima  Business Victoria ~ <aggregat~ Test   232.
##  6 arima  Business New South~ <aggregat~ Test   153.
##  7 arima  Business Queenslan~ <aggregat~ Test    81.8
##  8 arima  Business ACT        ~ <aggregat~ Test    35.9
##  9 arima  Business Tasmania ~ <aggregat~ Test    28.8
## 10 arima  Business South Aus~ Adelaide   Test    20.8
## # ... with 2,540 more rows, and 6 more variables:
## #   RMSE <dbl>, 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 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

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 reconcililation make any difference to the SNAIVE forecasts?

Survey

rstd.io/ws-survey