workshop.nectric.com.au/tidyfc2024

Time Series Analysis & Forecasting Using R

11. Multivariate modelling

Outline

Outline

Multivariate modelling

Multivariate models jointly describes the dynamic interrelationships between two or more measured variables.

They are particularly useful for:

- understanding how variables influence each other over time
- analysing causality and cointegration
- investigating the effect of shocks in the system.

Outline

Autoregression (AR)

Recall the AR(p) model for a **univariate** time series y_t is:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_p y_{t-p} + \varepsilon_t$$

where:

- y_t is the time series at time t,
- $a_1, a_2, \ldots, a_p$ are the coefficients for the autoregressive lags,
- p is the order of the AR process,
- ε_t is the white noise error term at time t.

Endogeneity

Endogeneity occurs when an explanatory variable is correlated with the error term in a model. This can cause a ‘feedback loop’ in the model estimation which:

- Leads to biased and inconsistent estimates.
- Compromises hypothesis testing and forecasting.

Endogeneity

Endogeneity occurs when an explanatory variable is correlated with the error term in a model. This can cause a ‘feedback loop’ in the model estimation which:

- Leads to biased and inconsistent estimates.
- Compromises hypothesis testing and forecasting.

Common Causes of endogeneity in time series include:

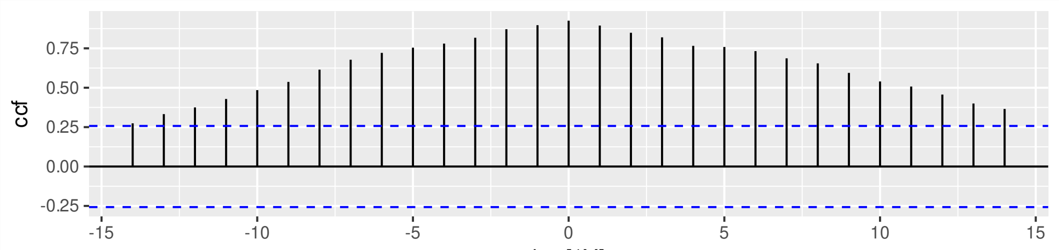
- Simultaneity: Variables with bidirectional contemporaneous effects.
- Omitted Variables: Missing relevant variables.
- Measurement Errors: Inaccuracy in data collection.

Cross correlations

The Cross-Correlation Function (CCF) measures the correlation between two time series at different time lags.

It indicates the relationship between two variables over time.

```
global_economy |>
  filter(Country == "Australia") |>
  CCF(Imports, Exports) |>
  autoplot()
```



Vector Autoregression (VAR)

The general form of a VAR model with k time series variables $Y_t = (y_{1t}, y_{2t}, \ldots, y_{kt})'$ is:

$$Y_t = A_1 Y_{t-1} + A_2 Y_{t-2} + \cdots + A_p Y_{t-p} + \varepsilon_t$$

where:

- Y_t is a vector of endogenous variables at time t,
- A_i are coefficient matrices (each of size $k \times k$),
- p is the lag order,
- ε_t is a vector of error terms (mean zero, covariance Σ).

Vector Autoregression (VAR)

In matrix form, the VAR(p) model is written as:

$$\Phi(B)Y_t = \varepsilon_t$$

where:

- Y_t is a $k \times 1$ vector of endogenous variables,
- $\Phi(B)$ is a matrix polynomial of AR coefficients,
- ε_t is a $k \times 1$ vector of error terms (white noise).

Vector Autoregression (VAR)

VAR (Vector Autoregression) models handle endogeneity by treating all variables in the system as endogenous.

Vector Autoregression (VAR)

VAR (Vector Autoregression) models handle endogeneity by treating all variables in the system as endogenous.

```
global_economy |>
  filter(Country == "Australia") |>
  model(var = VAR(vars(Imports, Exports)))
```

```
# A mable: 1 x 2
# Key:     Country [1]
  Country         var
  <fct>       <model>
1 Australia <VAR(1)>
```

Use `VAR()` to specify a model, specifying all response variables in the model with `vars()`.

Vector Autoregression (VAR)

```
global_economy |>
  filter(Country == "Australia") |>
  model(var = VAR(vars(Imports, Exports))) |>
  report()
```

```
Series: Imports, Exports
Model: VAR(1)

Coefficients for Imports:
      lag(Imports,1)  lag(Exports,1)
              0.6949           0.330
s.e.          0.0992           0.105

Coefficients for Exports:
      lag(Imports,1)  lag(Exports,1)
               0.406           0.577
s.e.           0.119           0.126

Residual covariance matrix:
```

Vector Autoregression (VAR)

Forecasts from VAR models are multivariate normal.

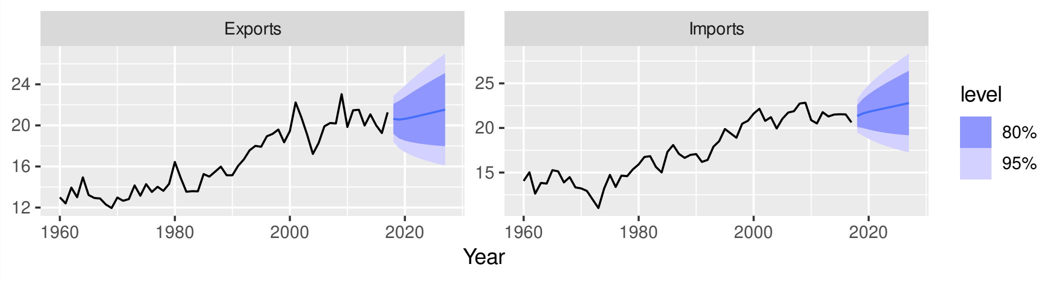
```r
global_economy |>
  filter(Country == "Australia") |>
  model(var = VAR(vars(Imports, Exports))) |>
  forecast(h = "10 years")
```

```
# A fable: 10 x 5 [1Y]
# Key:     Country, .model [1]
   Country   .model  Year .distribution .mean[,"Imports"] [,"Exports"]
   <fct>     <chr>  <dbl>        <dist>             <dbl>        <dbl>
 1 Australia var     2018        MVN[2]              21.3         20.6
 2 Australia var     2019        MVN[2]              21.6         20.6
 3 Australia var     2020        MVN[2]              21.8         20.6
 4 Australia var     2021        MVN[2]              22.0         20.8
 5 Australia var     2022        MVN[2]              22.1         20.9
 6 Australia var     2023        MVN[2]              22.2         21.0
 7 Australia var     2024        MVN[2]              22.4         21.1
 8 Australia var     2025        MVN[2]              22.5         21.3
```

Vector Autoregression (VAR)

Plotting multivariate forecasts shows intervals from marginal forecast distributions.

```
global_economy |>
  filter(Country == "Australia") |>
  model(var = VAR(vars(Imports, Exports))) |>
  forecast(h = "10 years") |>
  autoplot(global_economy)
```



Impulse Response Functions

Impulse Response Functions (IRFs) reveal the response of one variable to shocks in another.

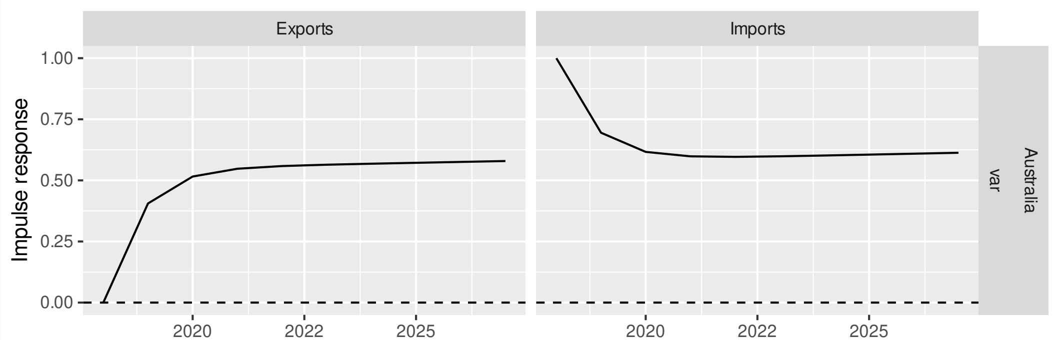
```
global_economy |>
  filter(Country == "Australia") |>
  model(var = VAR(vars(Imports, Exports))) |>
  IRF(h = 10, impulse = "Imports")
```

```
# A tsibble: 10 x 5 [1Y]
# Key:       Country, .model [1]
   Country   .model  Year Imports Exports
   <fct>     <chr>  <dbl>   <dbl>   <dbl>
 1 Australia var     2018   1       0
 2 Australia var     2019   0.695   0.405
 3 Australia var     2020   0.617   0.516
 4 Australia var     2021   0.598   0.548
 5 Australia var     2022   0.596   0.559
 6 Australia var     2023   0.598   0.564
```

Impulse Response Functions

They can be plotted using `gg_irf()`

```
global_economy |>
  filter(Country == "Australia") |>
  model(var = VAR(vars(Imports, Exports))) |>
  IRF(h = 10, impulse = "Imports") |>
  gg_irf()
```



Outline

Cointegration

Definition

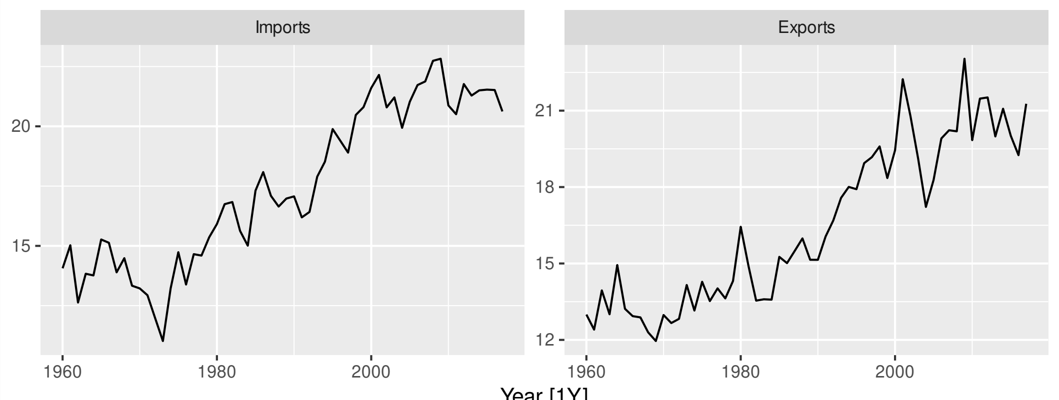
Cointegration is when two or more non-stationary time series, each integrated of the same order, are linked by a long-term equilibrium relationship.

For example: If x and y are non-stationary ($I(1)$), and a linear combination of them is stationary ($I(0)$), then x and y are cointegrated.

Cointegration

Is Australian imports and exports cointegrated?

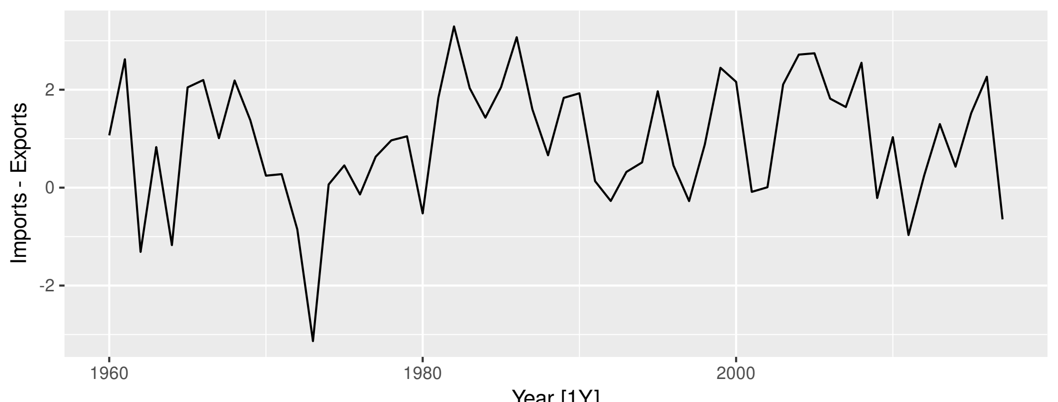
```
global_economy |>
  filter(Country == "Australia") |>
  autoplot(vars(Imports, Exports))
```



Cointegration

Yes, both are non-stationary but their difference is stationary.

```
global_economy |>
  filter(Country == "Australia") |>
  autoplot(Imports - Exports)
```



The Johansen test

The Johansen test helps to determine how many long-term equilibrium (cointegrating) relationships exist between the variables.

```
aus_economy <- global_economy |>
  filter(Country == "Australia")
aus_economy |>
  with(cointegration_johansen(cbind(Imports, Exports)))
```

```
$johansen_stat
  R<=1    R=0
 0.597 18.752

$johansen_pvalue
  R<=1    R=0
0.1000 0.0141
```

Vector Error Correction Models (VECM)

The VECM is used when variables are cointegrated.

The form of a VECM for Y_t is:

$$\Delta Y_t = \Pi Y_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta Y_{t-i} + \varepsilon_t$$

where:

- $\Delta Y_t = Y_t - Y_{t-1}$ is the first difference of Y_t,
- Π is the cointegration matrix (long-run coefficients)
- Γ_i are short-run adjustment coefficients,
- ε_t is the error term.

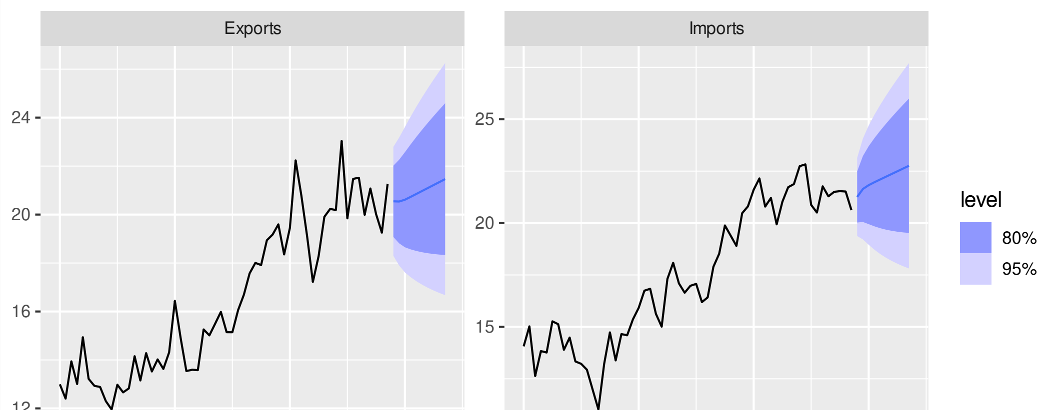
Estimating a VECM

```r
global_economy |>
  filter(Country == "Australia") |>
  model(vecm = VECM(vars(Imports, Exports) ~ 1 + AR(p = 1), r = 1)) |>
  report()
```

```
Series: Imports, Exports
Model: VECM(1, r=1) w/ mean

 Cointegrating vector:
           r1
Imports  1.00
Exports -1.08

Coefficients for Imports:
        ECT1  lag(Imports,1)  lag(Exports,1)  constant
      -0.347         -0.0107         -0.0943    0.0105
s.e.   0.125          0.1397          0.1292    0.1346

Coefficients for Exports:
```

Forecasting a VECM

```
global_economy |>
  filter(Country == "Australia") |>
  model(vecm = VECM(vars(Imports, Exports) ~ 1 + AR(p = 1), r = 1)) |>
  forecast(h = "10 years") |>
  autoplot(global_economy)
```



Outline

VARIMA Models

The VARIMA model extends an ARIMA model to capture relationships between multiple time series.

$$\Phi(B)\Delta^d Y_t = \Theta(B)\varepsilon_t$$

where:

- $\Phi(B)$ is the matrix polynomial for the AR part,
- $\Delta^d Y_t$ is the differenced series,
- $\Theta(B)$ is the matrix polynomial for the MA part,
- ε_t is the error term.

VARIMA Models

VARIMA models are useful for modelling multiple related non-stationary time series which are **not** cointegrated.

VARIMA Models

VARIMA models are useful for modelling multiple related non-stationary time series which are **not** cointegrated.

> **Handling non-stationarity**
>
> Much like ARIMA, we use transformations and differences to make non-stationary variables stationary.

Identifying VARIMA models

VARIMA models can be statistically difficult to estimate.

This is because their coefficients aren't always uniquely identified.

Identifying VARIMA models

VARIMA models can be statistically difficult to estimate.

This is because their coefficients aren't always uniquely identified.

❗ Constraints for identification

We must force some coefficients to zero (usually MA coefficients) to ensure identification.
Identification of VARIMA models is often achieved with estimation using Kronecker indices or scalar components.

Estimating VARIMA models

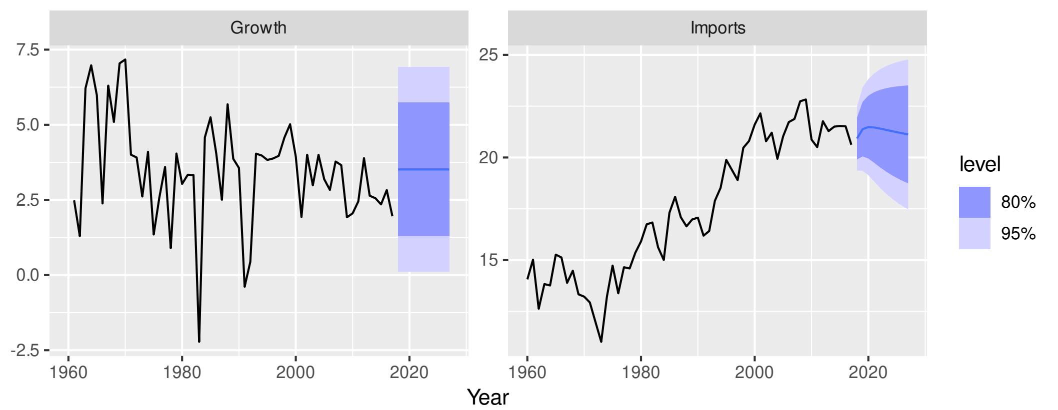
Much like `ARIMA()`, we use `VARIMA()` and `pdq()` to specify a VARIMA model.

```
aus_economy |>
  filter(!is.na(Growth)) |>
  model(varima = VARIMA(vars(Growth, Imports))) |>
  report()
```

```
Series: Growth, Imports
Model: VARIMA(2,0,2) w/ mean

Coefficients for Growth:
      constant  lag(Growth,1)  lag(Growth,2)  lag(Imports,1)
Est.      3.51              0              0               0
      lag(Imports,2)  lag(e_Growth,1)  lag(e_Growth,2)  lag(e_Imports,1)
Est.               0                0                0                 0
      lag(e_Imports,2)
Est.                 0
```

Forecasting VARIMA models

```
aus_economy |>
  filter(!is.na(Growth)) |>
  model(varima = VARIMA(vars(Growth, Imports))) |>
  forecast(h = "10 years") |>
  autoplot(aus_economy)
```



Outline

Lab Session 18

Outline

Outline

Australian tourism

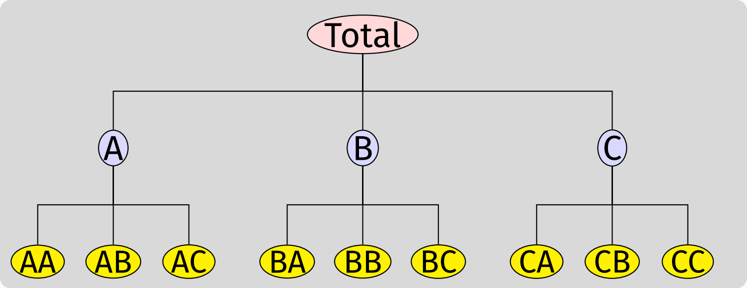
```
tourism
```

```
# A tsibble: 24,320 x 5 [1Q]
# Key:       Region, State, Purpose [304]
   Quarter Region   State           Purpose  T
     <qtr> <chr>    <chr>           <chr>    <
 1 1998 Q1 Adelaide South Australia Business
 2 1998 Q2 Adelaide South Australia Business
 3 1998 Q3 Adelaide South Australia Business
 4 1998 Q4 Adelaide South Australia Business
 5 1999 Q1 Adelaide South Australia Business
 6 1999 Q2 Adelaide South Australia Business
 7 1999 Q3 Adelaide South Australia Business
 8 1999 Q4 Adelaide South Australia Business
 9 2000 Q1 Adelaide South Australia Business
10 2000 Q2 Adelaide South Australia Business
# i 24,310 more rows
```

- Quarterly data on visitor nights, 1998:Q1 – 2017:Q4
- From: *National Visitor Survey*, based on annual interviews of 120,000 Australians aged 15+, collected by Tourism Research Australia.
- Split by 8 states and 76 regions
- Split by purpose of travel
  - Holiday
  - Visiting friends and relatives (VFR)
  - Business
  - Other
- 304 bottom-level series

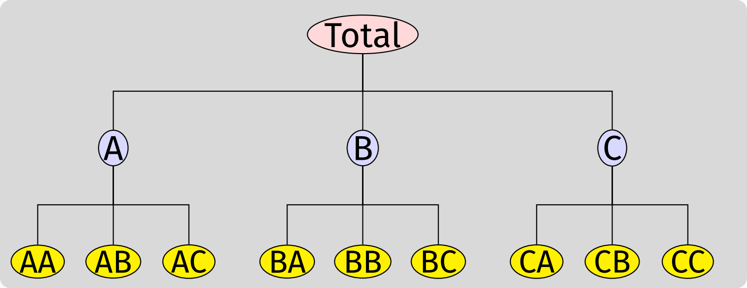
Hierarchical time series

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.

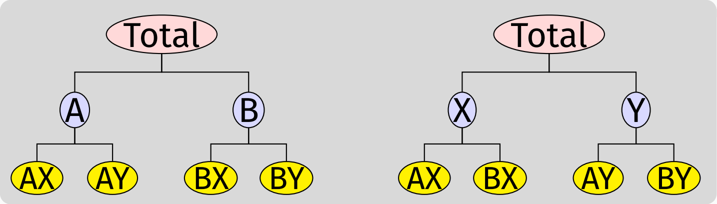


Examples

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

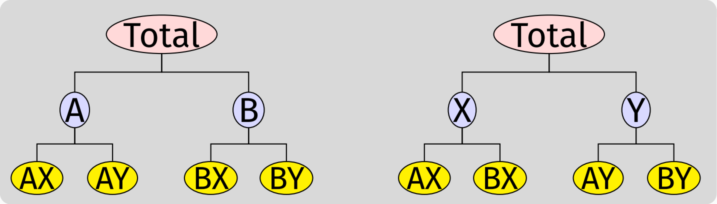
Grouped time series

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

```
tourism |>
  aggregate_key(Purpose * (State / Region), Trips = sum(Trips)) |>
  arrange(Quarter) |>
  print(n=20)
```

```
# A tsibble: 34,000 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 Territory <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 Australia  <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.

Forecast reconciliation: 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?

Forecast reconciliation: 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?

Forecast reconciliation: 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
   <chr*>   <chr*>          <chr*>          <chr>          <qtr>
 1 Business ACT             Canberra        ets          2018 Q1
 2 Business ACT             Canberra        ets          2018 Q2
 3 Business ACT             Canberra        ets_adjusted 2018 Q1
 4 Business ACT             Canberra        ets_adjusted 2018 Q2
 5 Business ACT             <aggregated>    ets          2018 Q1
 6 Business ACT             <aggregated>    ets          2018 Q2
 7 Business ACT             <aggregated>    ets_adjusted 2018 Q1
 8 Business ACT             <aggregated>    ets_adjusted 2018 Q2
 9 Business New South Wales Blue Mountains ets          2018 Q1
```

Hierarchical and grouped time series

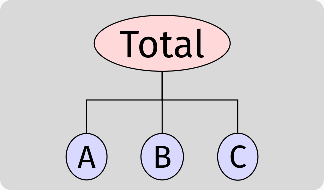
Every collection of time series with aggregation constraints can be written as

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

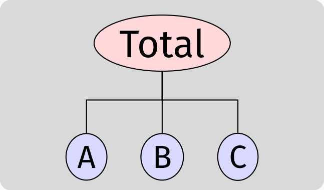
where

- $\boldsymbol{y}_t$ 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

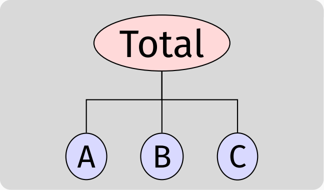


y_t : observed aggregate of all series at time t.

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

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

Hierarchical time series



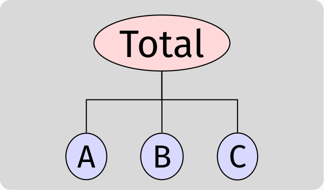
y_t : observed aggregate of all series at time t.

$y_{X,t}$: 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

Outline

Simple combination forecasts

There are several ways to adjust forecasts to be coherent.

- `bottom_up()`

  Simply aggregate the most disaggregated forecasts

- `top_down()`

  Disaggregate the top level forecast

- `middle_out()`

  Find a middle level of disaggregation, applying both bottom up and top down techniques.

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

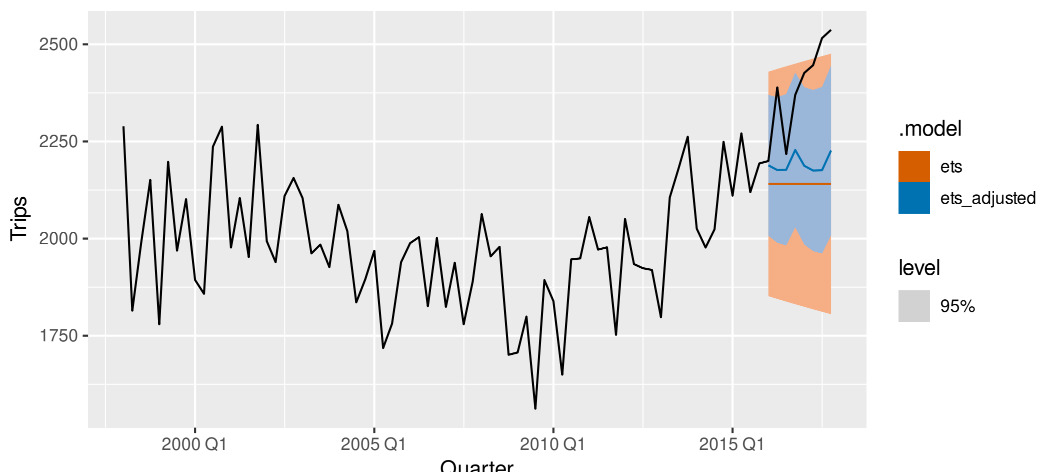
- Covariates can be included in initial forecasts.
- Adjustments can be made to initial forecasts at any level.
- Very simple and flexible method. Can work with *any* hierarchical or grouped time series.
- Conceptually easy to implement: regression of base forecasts on structure matrix.

Example: Australian tourism

```r
tourism_agg <- tourism |>
  aggregate_key(Purpose * (State / Region),
    Trips = sum(Trips)
  )
fc <- tourism_agg |>
  filter(Quarter < yearquarter("2016 Q1")) |>
  model(ets = ETS(Trips)) |>
  reconcile(ets_adjusted = min_trace(ets)) |>
  forecast(h = "2 years")
```

Example: Australian tourism

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



Example: Australian tourism

```
fc |>
  accuracy(tourism_agg) |>
  group_by(.model) |>
  summarise(across(where(is.numeric), mean))
```

```
# A tibble: 2 x 9
  .model          ME  RMSE   MAE   MPE  MAPE  MASE RMSSE    ACF1
  <chr>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
1 ets           28.4  46.0  38.0   NaN   Inf  1.04 0.976 -0.0825
2 ets_adjusted  31.8  48.9  41.2   NaN   Inf  1.02 0.958 -0.0830
```

Outline

Lab Session 19

Produce coherent forecasts of the total number of students.

1. Poduce aggregates of `students` by `State/Territory`, `Affiliation (Gov/Cath/Ind)` and `School Level`.
2. Estimate automatic ETS models on data before 2020.
3. Use optimal forecast reconciliation for these models.
4. Produce and plot 4 years of coherent and base forecasts.
5. Evaluate the test-set accuracy of these forecasts Are the coherent forecasts more accurate?