

Time series analysis using R



Rob J Hyndman

robjhyndman/fable2021

Outline

- 1 tsibble: Time series data
- 2 feasts: Data visualization
- 3 feasts: Time series features
- 4 fable: Forecasting
- 5 fable: Evaluating forecast accuracy
- 6 fable: Forecast reconciliation

Outline

- 1 tsibble: Time series data
- 2 feasts: Data visualization
- 3 feasts: Time series features
- 4 fable: Forecasting
- 5 fable: Evaluating forecast accuracy
- 6 fable: Forecast reconciliation





```
library(fpp3)
```

```
-- Attaching packages ----- fpp3 0.4.0 --
✓ tibble      3.1.4      ✓ tsibble      1.0.1
✓ dplyr       1.0.7      ✓ tsibbledata  0.3.0
✓ tidyr       1.1.3      ✓ feasts       0.2.2
✓ lubridate   1.7.10     ✓ fable        0.3.1
✓ ggplot2     3.3.5

-- 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()
```

tsibble objects

```
global_economy
```

```
## # A tsibble: 15,150 x 6 [1Y]
```

```
## # Key:      Country [263]
```

##	Year	Country	GDP	Imports	Exports	Population
##	<dbl>	<fct>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	1960	Afghanistan	537777811.	7.02	4.13	8996351
## 2	1961	Afghanistan	548888896.	8.10	4.45	9166764
## 3	1962	Afghanistan	546666678.	9.35	4.88	9345868
## 4	1963	Afghanistan	751111191.	16.9	9.17	9533954
## 5	1964	Afghanistan	800000044.	18.1	8.89	9731361
## 6	1965	Afghanistan	1006666638.	21.4	11.3	9938414
## 7	1966	Afghanistan	1399999967.	18.6	8.57	10152331
## 8	1967	Afghanistan	1673333418.	14.2	6.77	10372630
## 9	1968	Afghanistan	1373333367.	15.2	8.90	10604346
## 10	1969	Afghanistan	1408888922.	15.0	10.1	10854428

tsibble objects

```
global_economy
```

```
## # A tsibble: 15,150 x 6 [1Y]
```

```
## # Key:          Country [263]
```

##	Year	Country	GDP	Imports	Exports	Population
##	Index	<fct>	<dbl>	<dbl>	<dbl>	<dbl>
##	1	1960 Afghanistan	537777811.	7.02	4.13	8996351
##	2	1961 Afghanistan	548888896.	8.10	4.45	9166764
##	3	1962 Afghanistan	546666678.	9.35	4.88	9345868
##	4	1963 Afghanistan	751111191.	16.9	9.17	9533954
##	5	1964 Afghanistan	800000044.	18.1	8.89	9731361
##	6	1965 Afghanistan	1006666638.	21.4	11.3	9938414
##	7	1966 Afghanistan	1399999967.	18.6	8.57	10152331
##	8	1967 Afghanistan	1673333418.	14.2	6.77	10372630
##	9	1968 Afghanistan	1373333367.	15.2	8.90	10604346
##	10	1969 Afghanistan	1408888922.	15.0	10.1	10854428

tsibble objects

```
global_economy
```

```
## # A tsibble: 15,150 x 6 [1Y]
```

```
## # Key:          Country [263]
```

##	Year	Country	GDP	Imports	Exports	Population
##	Index	Key	<dbl>	<dbl>	<dbl>	<dbl>
##	1	1960 Afghanistan	537777811.	7.02	4.13	8996351
##	2	1961 Afghanistan	548888896.	8.10	4.45	9166764
##	3	1962 Afghanistan	546666678.	9.35	4.88	9345868
##	4	1963 Afghanistan	751111191.	16.9	9.17	9533954
##	5	1964 Afghanistan	800000044.	18.1	8.89	9731361
##	6	1965 Afghanistan	1006666638.	21.4	11.3	9938414
##	7	1966 Afghanistan	1399999967.	18.6	8.57	10152331
##	8	1967 Afghanistan	1673333418.	14.2	6.77	10372630
##	9	1968 Afghanistan	1373333367.	15.2	8.90	10604346
##	10	1969 Afghanistan	1408888922.	15.0	10.1	10854428

tsibble objects

```
global_economy
```

```
## # A tsibble: 15,150 x 6 [1Y]
```

```
## # Key:      Country [263]
```

```
##      Year Country      GDP Imports Exports Population
```

```
##      Index  Key      Measured variables
```

```
## 1  1960 Afghanistan 5377777811.    7.02    4.13    8996351
```

```
## 2  1961 Afghanistan 5488888896.    8.10    4.45    9166764
```

```
## 3  1962 Afghanistan 5466666678.    9.35    4.88    9345868
```

```
## 4  1963 Afghanistan 7511111191.   16.9    9.17    9533954
```

```
## 5  1964 Afghanistan 8000000044.   18.1    8.89    9731361
```

```
## 6  1965 Afghanistan 10066666638.   21.4   11.3    9938414
```

```
## 7  1966 Afghanistan 13999999967.   18.6    8.57   10152331
```

```
## 8  1967 Afghanistan 16733333418.   14.2    6.77   10372630
```

```
## 9  1968 Afghanistan 13733333367.   15.2    8.90   10604346
```

```
## 10 1969 Afghanistan 14088888922.   15.0   10.1   10854428
```

tsibble objects

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.
```

Domestic visitor
nights in thousands
by state/region and
purpose.

tsibble objects

tourism

```
## # A tsibble: 24,320 x 5 [1Q]
## # Key:           Region, State, Purpose [304]
##   Quarter Region  State Purpose  Trips
##   Index    <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.
```

Domestic visitor
nights in thousands
by state/region and
purpose.

tsibble objects

tourism

```
## # A tsibble: 24,320 x 5 [1Q]
```

```
## # Key:      Region, State, Purpose [304]
```

```
##   Quarter Region State Purpose Trips
```

```
##   Index      Keys      <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.
```

Domestic visitor
nights in thousands
by state/region and
purpose.

tsibble objects

tourism

```
## # A tsibble: 24,320 x 5 [1Q]
```

```
## # Key:      Region, State, Purpose [304]
```

```
##   Quarter Region State Purpose Trips
```

```
##   Index      Keys      Measure
```

```
## 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.
```

Domestic visitor
nights in thousands
by state/region and
purpose.

tsibble objects

- A `tsibble` allows storage and manipulation of multiple time series in R.
- It contains:
 - ▶ An index: time information about the observation
 - ▶ Measured variable(s): numbers of interest
 - ▶ Key variable(s): optional unique identifiers for each series
- It works with tidyverse functions.

Example: Australian prison population



Read a csv file and convert to a tibble

```
prison <- readr::read_csv("data/prison_population.csv")
```

```
## # A tibble: 3,072 x 6
```

##	date	state	gender	legal	indigenous	count
##	<date>	<chr>	<chr>	<chr>	<chr>	<dbl>
## 1	2005-03-01	ACT	Female	Remanded	ATSI	0
## 2	2005-03-01	ACT	Female	Remanded	Other	2
## 3	2005-03-01	ACT	Female	Sentenced	ATSI	0
## 4	2005-03-01	ACT	Female	Sentenced	Other	0
## 5	2005-03-01	ACT	Male	Remanded	ATSI	7
## 6	2005-03-01	ACT	Male	Remanded	Other	58
## 7	2005-03-01	ACT	Male	Sentenced	ATSI	0
## 8	2005-03-01	ACT	Male	Sentenced	Other	0
## 9	2005-03-01	NSW	Female	Remanded	ATSI	51
## 10	2005-03-01	NSW	Female	Remanded	Other	131
## #	... with 3,062 more rows					

Read a csv file and convert to a tibble

```
prison <- readr::read_csv("data/prison_population.csv") %>%  
  mutate(Quarter = yearquarter(date))
```

```
## # A tibble: 3,072 x 7
```

##	date	state	gender	legal	indigenous	count	Quarter
##	<date>	<chr>	<chr>	<chr>	<chr>	<dbl>	<qtr>
## 1	2005-03-01	ACT	Female	Remanded	ATSI	0	2005 Q1
## 2	2005-03-01	ACT	Female	Remanded	Other	2	2005 Q1
## 3	2005-03-01	ACT	Female	Sentenced	ATSI	0	2005 Q1
## 4	2005-03-01	ACT	Female	Sentenced	Other	0	2005 Q1
## 5	2005-03-01	ACT	Male	Remanded	ATSI	7	2005 Q1
## 6	2005-03-01	ACT	Male	Remanded	Other	58	2005 Q1
## 7	2005-03-01	ACT	Male	Sentenced	ATSI	0	2005 Q1
## 8	2005-03-01	ACT	Male	Sentenced	Other	0	2005 Q1
## 9	2005-03-01	NSW	Female	Remanded	ATSI	51	2005 Q1
## 10	2005-03-01	NSW	Female	Remanded	Other	131	2005 Q1

Read a csv file and convert to a tibble

```
prison <- readr::read_csv("data/prison_population.csv") %>%  
  mutate(Quarter = yearquarter(date)) %>%  
  select(-date)
```

```
## # A tibble: 3,072 x 6
```

```
##   state gender legal      indigenous count Quarter  
##   <chr> <chr>  <chr>      <chr>      <dbl>   <qtr>  
## 1 ACT    Female Remanded  ATSI          0 2005 Q1  
## 2 ACT    Female Remanded  Other          2 2005 Q1  
## 3 ACT    Female Sentenced ATSI          0 2005 Q1  
## 4 ACT    Female Sentenced Other          0 2005 Q1  
## 5 ACT    Male    Remanded  ATSI          7 2005 Q1  
## 6 ACT    Male    Remanded  Other         58 2005 Q1  
## 7 ACT    Male    Sentenced ATSI          0 2005 Q1  
## 8 ACT    Male    Sentenced Other          0 2005 Q1  
## 9 NSW    Female Remanded  ATSI         51 2005 Q1
```

Read a csv file and convert to a tsibble

```
prison <- readr::read_csv("data/prison_population.csv") %>%  
  mutate(Quarter = yearquarter(date)) %>%  
  select(-date) %>%  
  as_tsibble(  
    index = Quarter,  
    key = c(state, gender, legal, indigenous)  
  )
```

```
## # A tsibble: 3,072 x 6 [1Q]  
## # Key:      state, gender, legal, indigenous [64]  
##   state gender legal    indigenous count Quarter  
##   <chr> <chr>  <chr>      <chr>      <dbl>   <qtr>  
## 1 ACT   Female Remanded ATSI          0 2005 Q1  
## 2 ACT   Female Remanded ATSI          1 2005 Q2  
## 3 ACT   Female Remanded ATSI          0 2005 Q3  
## 4 ACT   Female Remanded ATSI          0 2005 Q4
```

The tsibble index

Common time index variables can be created with these functions:

Frequency	Function
Annual	<code>start:end</code>
Quarterly	<code>yearquarter()</code>
Monthly	<code>yearmonth()</code>
Weekly	<code>yearweek()</code>
Daily	<code>as_date()</code> , <code>ymd()</code>
Sub-daily	<code>as_datetime()</code>

Outline

- 1 tsibble: Time series data
- 2 feasts: Data visualization
- 3 feasts: Time series features
- 4 fable: Forecasting
- 5 fable: Evaluating forecast accuracy
- 6 fable: Forecast reconciliation

Australian holidays

```
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.
## # ... with 24,310 more rows
```

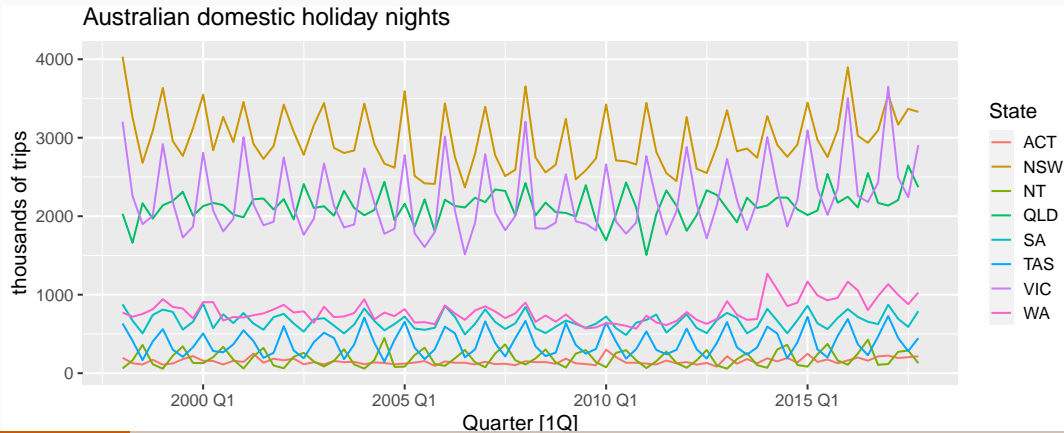
Australian holidays

```
holidays <- tourism %>%  
  filter(Purpose == "Holiday") %>%  
  group_by(State) %>%  
  summarise(Trips = sum(Trips))
```

```
## # A tsibble: 640 x 3 [1Q]  
## # Key:      State [8]  
##   State Quarter Trips  
##   <chr>    <qtr> <dbl>  
## 1 ACT     1998 Q1  196.  
## 2 ACT     1998 Q2  127.  
## 3 ACT     1998 Q3  111.  
## 4 ACT     1998 Q4  170.  
## 5 ACT     1999 Q1  108.  
## 6 ACT     1999 Q2  125.  
## 7 ACT     1999 Q3  178.  
## 8 ACT     1999 Q4  218.  
## 9 ACT     2000 Q1  158.
```

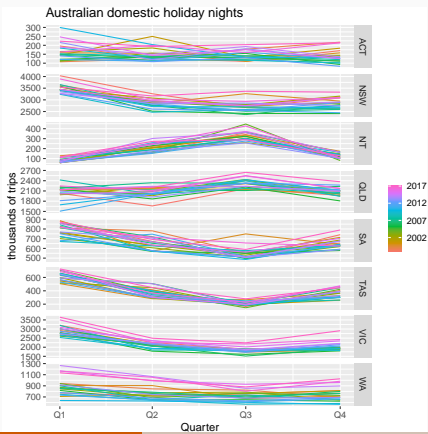
Australian holidays

```
holidays %>% autoplot(Trips) +  
  ylab("thousands of trips") +  
  ggtitle("Australian domestic holiday nights")
```



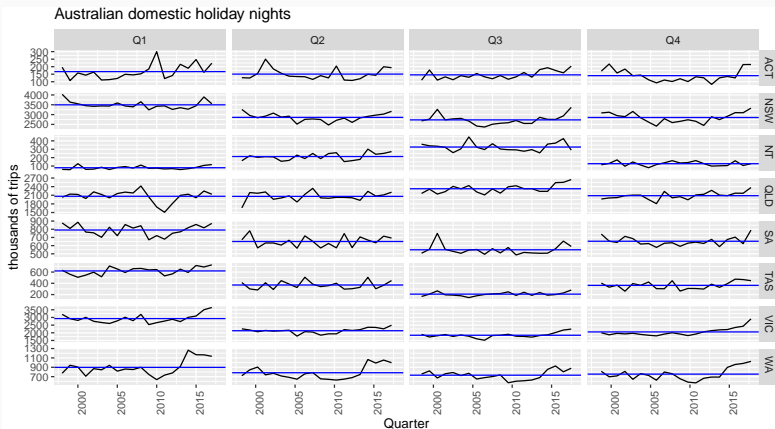
Seasonal plots

```
holidays %>% gg_season(Trips) +  
  ylab("thousands of trips") +  
  ggtitle("Australian domestic holiday nights")
```



Seasonal subseries plots

```
holidays %>% gg_subseries(Trips) +  
  ylab("thousands of trips") +  
  ggtitle("Australian domestic holiday nights")
```



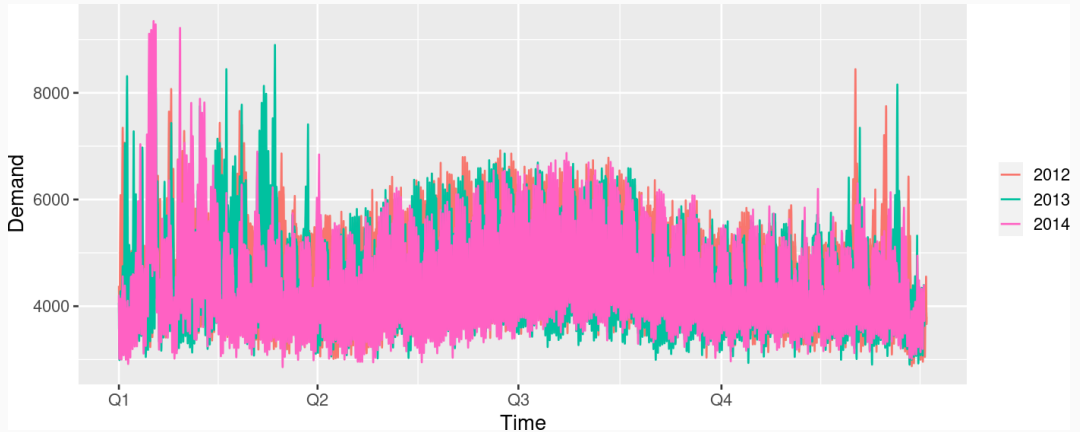
Victorian electricity demand

vic_elec

```
## # A tibble: 52,608 x 5 [30m] <Australia/Melbourne>
##   Time                Demand Temperature Date        Holiday
##   <dtm>                <dbl>         <dbl> <date>      <lgl>
## 1 2012-01-01 00:00:00  4383.          21.4 2012-01-01 TRUE
## 2 2012-01-01 00:30:00  4263.          21.0 2012-01-01 TRUE
## 3 2012-01-01 01:00:00  4049.          20.7 2012-01-01 TRUE
## 4 2012-01-01 01:30:00  3878.          20.6 2012-01-01 TRUE
## 5 2012-01-01 02:00:00  4036.          20.4 2012-01-01 TRUE
## 6 2012-01-01 02:30:00  3866.          20.2 2012-01-01 TRUE
## 7 2012-01-01 03:00:00  3694.          20.1 2012-01-01 TRUE
## 8 2012-01-01 03:30:00  3562.          19.6 2012-01-01 TRUE
## 9 2012-01-01 04:00:00  3433.          19.1 2012-01-01 TRUE
## 10 2012-01-01 04:30:00  3359.          19.0 2012-01-01 TRUE
## # ... with 52,598 more rows
```

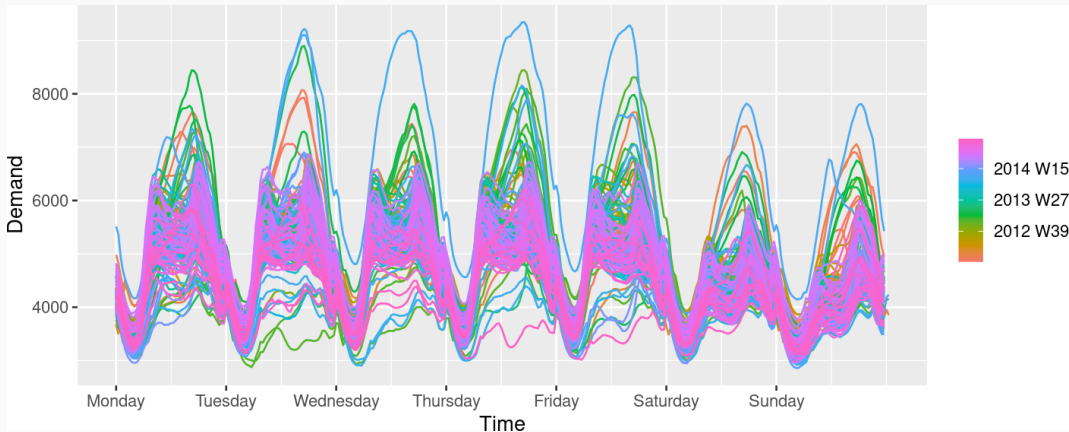
Multiple seasonal periods

```
vic_elec %>% gg_season(Demand)
```



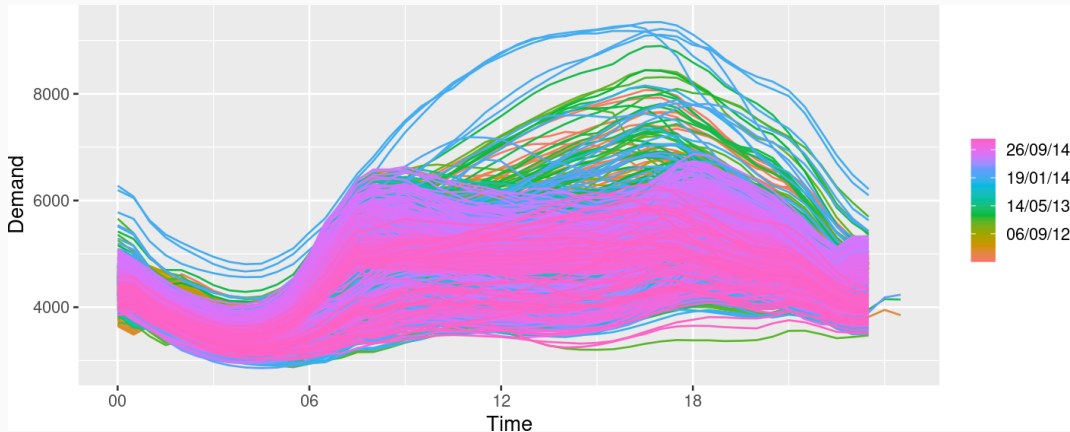
Multiple seasonal periods

```
vic_elec %>% gg_season(Demand, period = "week")
```



Multiple seasonal periods

```
vic_elec %>% gg_season(Demand, period = "day")
```



Time plots

- Plotted against time: `autoplot()` (each series overplotted)
- Plotted against season: `gg_season()` (facet by key)
- Plotted against time with seasonal facets: `gg_subseries()` (facet by key)

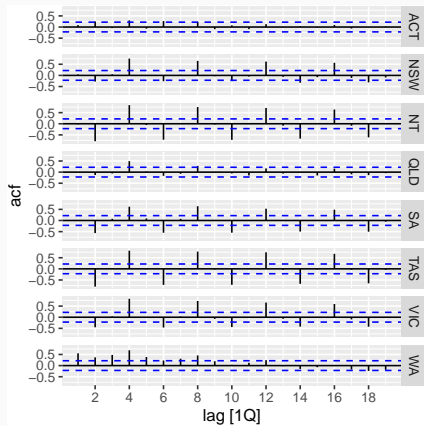
Autocorrelations

```
holidays %>% ACF(Trips)
```

```
## # A tsibble: 152 x 3 [1Q]
## # Key:      State [8]
##   State lag    acf
##   <chr> <lag>  <dbl>
## 1 ACT   1Q  0.0877
## 2 ACT   2Q  0.252
## 3 ACT   3Q -0.0496
## 4 ACT   4Q  0.300
## 5 ACT   5Q -0.0741
## 6 ACT   6Q  0.269
## 7 ACT   7Q -0.00504
## 8 ACT   8Q  0.236
## 9 ACT   9Q -0.0953
## 10 ACT  10Q  0.0750
## # ... with 142 more rows
```


ACF plots

```
holidays %>%  
  ACF(Trips) %>%  
  autoplot()
```



Google stock price

```
gafa_stock
```

```
## # A tsibble: 5,032 x 8 [!]
```

```
## # Key:      Symbol [4]
```

##	Symbol	Date	Open	High	Low	Close	Adj_Close	Volume
##	<chr>	<date>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
##	1 AAPL	2014-01-02	79.4	79.6	78.9	79.0	67.0	58671200
##	2 AAPL	2014-01-03	79.0	79.1	77.2	77.3	65.5	98116900
##	3 AAPL	2014-01-06	76.8	78.1	76.2	77.7	65.9	103152700
##	4 AAPL	2014-01-07	77.8	78.0	76.8	77.1	65.4	79302300
##	5 AAPL	2014-01-08	77.0	77.9	77.0	77.6	65.8	64632400
##	6 AAPL	2014-01-09	78.1	78.1	76.5	76.6	65.0	69787200 ₂₇
##	7 AAPL	2014-01-10	77.1	77.3	75.9	76.1	64.5	76244000

Google stock price

```
google_2015 <- gafa_stock %>%  
  filter(Symbol == "GOOG", year(Date) == 2015) %>%  
  select(Date, Close)  
google_2015
```

```
## # A tsibble: 252 x 2 [!]
```

```
##   Date      Close
```

```
##   <date>    <dbl>
```

```
## 1 2015-01-02  522.
```

```
## 2 2015-01-05  511.
```

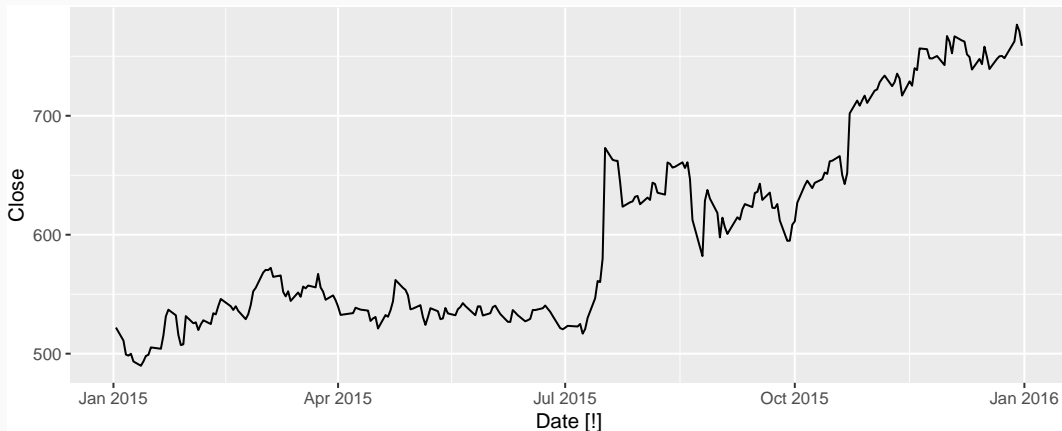
```
## 3 2015-01-06  499.
```

```
## 4 2015-01-07  498.
```

```
## 5 2015-01-08  500.
```

Google stock price

```
google_2015 %>% autoplot(Close)
```



Google stock price

```
google_2015 %>%  
  ACF(Close, lag_max = 100)
```

```
## Warning: Provided data has an irregular interval, results should be treated with  
## caution. Computing ACF by observation.
```

```
## # A tsibble: 100 x 2 [1]
```

```
##      lag    acf
```

```
##    <lag> <dbl>
```

```
##  1      1 0.982
```

```
##  2      2 0.959
```

```
##  3      3 0.937
```

```
##  4      4 0.918
```

```
##  5      5 0.901
```

```
##  6      6 0.883
```

```
##  7      7 0.865
```

```
##  8      8 0.849
```

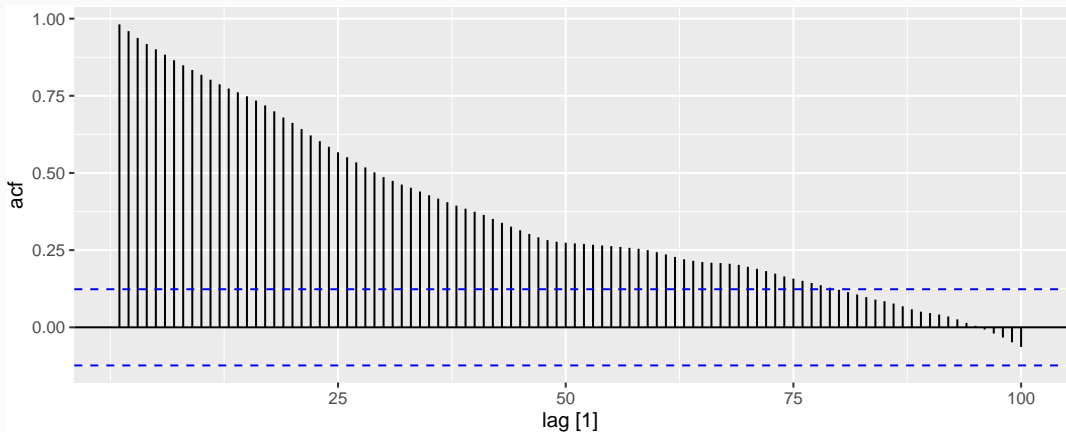
```
##  9      9 0.834
```

```
## 10     10 0.818
```

```
## # with 90 more rows
```

Google stock price

```
google_2015 %>%  
  ACF(Close, lag_max = 100) %>%  
  autoplot()
```



Outline

- 1 tsibble: Time series data
- 2 feasts: Data visualization
- 3 feasts: Time series features
- 4 fable: Forecasting
- 5 fable: Evaluating forecast accuracy
- 6 fable: Forecast reconciliation

Strength of seasonality and trend

STL decomposition

$$y_t = T_t + S_t + R_t$$

Seasonal strength

$$\max \left(0, 1 - \frac{\text{Var}(R_t)}{\text{Var}(S_t + R_t)} \right)$$

Trend strength

$$\max \left(0, 1 - \frac{\text{Var}(R_t)}{\text{Var}(T_t + R_t)} \right)$$

Feature extraction and statistics

```
tourism %>% features(Trips, feat_stl)
```

```
## # A tibble: 304 x 12
##   Region      State Purpose trend_strength seasonal_strength_~ seasonal_peak_ye~
##   <chr>      <chr> <chr>          <dbl>          <dbl>          <dbl>
## 1 Adelaide    SA     Business    0.464          0.407           3
## 2 Adelaide    SA     Holiday     0.554          0.619           1
## 3 Adelaide    SA     Other       0.746          0.202           2
## 4 Adelaide    SA     Visiting    0.435          0.452           1
## 5 Adelaide Hills SA     Business    0.464          0.179           3
## 6 Adelaide Hills SA     Holiday     0.528          0.296           2
## 7 Adelaide Hills SA     Other       0.593          0.404           2
## 8 Adelaide Hills SA     Visiting    0.488          0.254           0
## 9 Alice Springs NT     Business    0.534          0.251           0
## 10 Alice Springs NT     Holiday     0.381          0.832           3
## # ... with 294 more rows, and 6 more variables: seasonal_trough_year <dbl>,
## #   spikiness <dbl>, linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>,
## #   stl_e_acf10 <dbl>
```

Feature extraction and statistics

```
tourism %>%  
  features(Trips, feat_stl) %>%  
  ggplot(aes(x = trend_strength, y = seasonal_strength_year, col = Purpose)) +  
  geom_point() +  
  facet_wrap(vars(State))
```



Feature extraction and statistics

```
tourism %>%  
  features(Trips, feat_stl) %>%  
  ggplot(aes(x = trend_strength, y = seasonal_strength_year, col = Purpose)) +  
  geom_point() +  
  facet_wrap(vars(State))
```



- Holidays more seasonal than other travel.
- WA has strongest trends.

Feature extraction and statistics

Find the most seasonal time series:

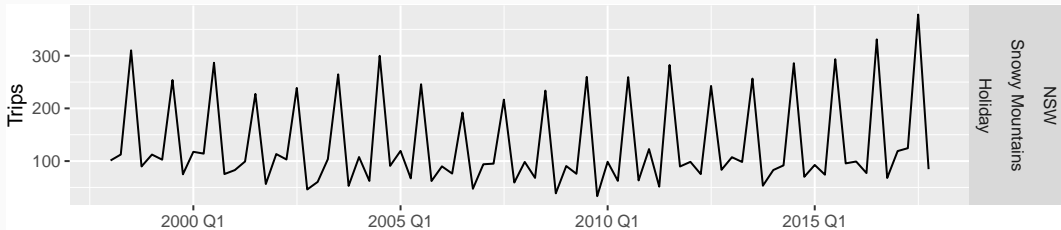
```
most_seasonal <- tourism %>%  
  features(Trips, feat_stl) %>%  
  filter(seasonal_strength_year == max(seasonal_strength_year))
```

Feature extraction and statistics

Find the most seasonal time series:

```
most_seasonal <- tourism %>%  
  features(Trips, feat_stl) %>%  
  filter(seasonal_strength_year == max(seasonal_strength_year))
```

```
tourism %>%  
  right_join(most_seasonal, by = c("State", "Region", "Purpose")) %>%  
  ggplot(aes(x = Quarter, y = Trips)) +  
  geom_line() +  
  facet_grid(vars(State, Region, Purpose))
```



Feature extraction and statistics

Find the most trended time series:

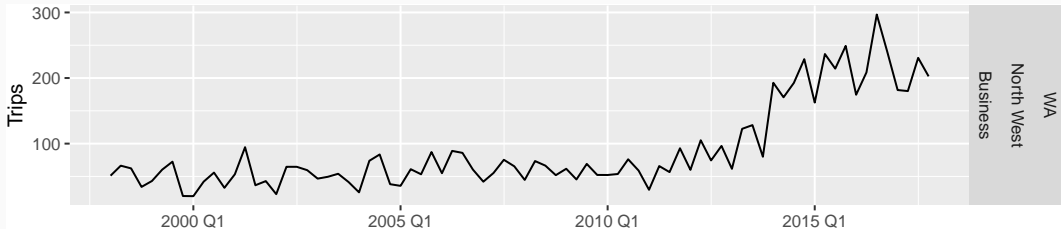
```
most_trended <- tourism %>%  
  features(Trips, feat_stl) %>%  
  filter(trend_strength == max(trend_strength))
```

Feature extraction and statistics

Find the most trended time series:

```
most_trended <- tourism %>%  
  features(Trips, feat_stl) %>%  
  filter(trend_strength == max(trend_strength))
```

```
tourism %>%  
  right_join(most_trended, by = c("State", "Region", "Purpose")) %>%  
  ggplot(aes(x = Quarter, y = Trips)) +  
  geom_line() +  
  facet_grid(vars(State, Region, Purpose))
```



Feature extraction and statistics

```
tourism_features <- tourism %>%  
  features(Trips, feature_set(pkgs = "feasts"))
```

All features from the feasts package

```
## # A tibble: 304 x 51  
##   Region      State Purpose trend_strength seasonal_strength_~ seasonal_peak_ye~  
##   <chr>      <chr> <chr>         <dbl>         <dbl>         <dbl>  
## 1 Adelaide    SA     Business    0.464         0.407         3  
## 2 Adelaide    SA     Holiday     0.554         0.619         1  
## 3 Adelaide    SA     Other       0.746         0.202         2  
## 4 Adelaide    SA     Visiting    0.435         0.452         1  
## 5 Adelaide Hills SA     Business    0.464         0.179         3  
## 6 Adelaide Hills SA     Holiday     0.528         0.296         2  
## 7 Adelaide Hills SA     Other       0.593         0.404         2  
## 8 Adelaide Hills SA     Visiting    0.488         0.254         0  
## 9 Alice Springs NT     Business    0.534         0.251         0  
## 10 Alice Springs NT     Holiday     0.381         0.832         3  
## # ... with 294 more rows, and 45 more variables: seasonal_trough_year <dbl>,  
## #   spikiness <dbl>, linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>,  
## #   stl_e_acf10 <dbl>, acf1 <dbl>, acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>,  
## #   diff2_acf1 <dbl>, diff2_acf10 <dbl>, season_acf1 <dbl>, pacf5 <dbl>
```


Feature extraction and statistics

```
pcs <- tourism_features %>%  
  select(-State, -Region, -Purpose) %>%  
  prcomp(scale = TRUE) %>%  
  broom::augment(tourism_features)
```

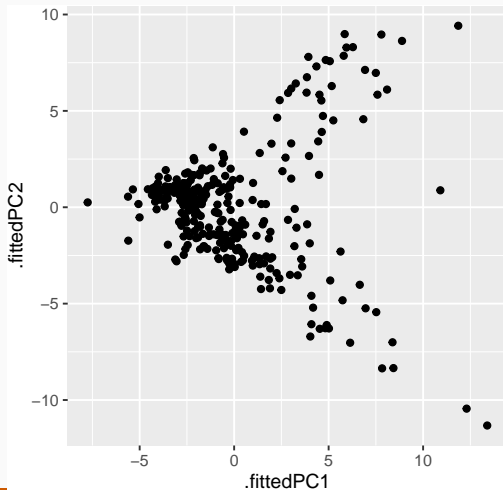
Principal components based on all
features from the feasts package

```
## # A tibble: 304 x 100  
##   .rownames Region      State Purpose trend_strength seasonal_streng~ seasonal_peak_y~  
##   <chr>      <chr>      <chr> <chr>      <dbl>          <dbl>          <dbl>  
## 1 1         Adelaide    SA     Business  0.464          0.407           3  
## 2 2         Adelaide    SA     Holiday   0.554          0.619           1  
## 3 3         Adelaide    SA     Other     0.746          0.202           2  
## 4 4         Adelaide    SA     Visiting  0.435          0.452           1  
## 5 5         Adelaide Hills SA     Business  0.464          0.179           3  
## 6 6         Adelaide Hills SA     Holiday   0.528          0.296           2  
## 7 7         Adelaide Hills SA     Other     0.593          0.404           2  
## 8 8         Adelaide Hills SA     Visiting  0.488          0.254           0  
## 9 9         Alice Springs NT     Business  0.534          0.251           0  
## 10 10        Alice Springs NT     Holiday   0.381          0.832           3  
## # ... with 294 more rows, and 93 more variables: seasonal_trough_year <dbl>,  
## #   spikiness <dbl>, linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>,  
## #   stl_e_acf10 <dbl>, acf1 <dbl>, acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>,  
## #   diff2_acf1 <dbl>, diff2_acf10 <dbl>, season_acf1 <dbl>, pacf5 <dbl>,
```

Feature extraction and statistics

```
pcs %>% ggplot(aes(x=.fittedPC1, y=.fittedPC2)) +  
  geom_point() + theme(aspect.ratio=1)
```

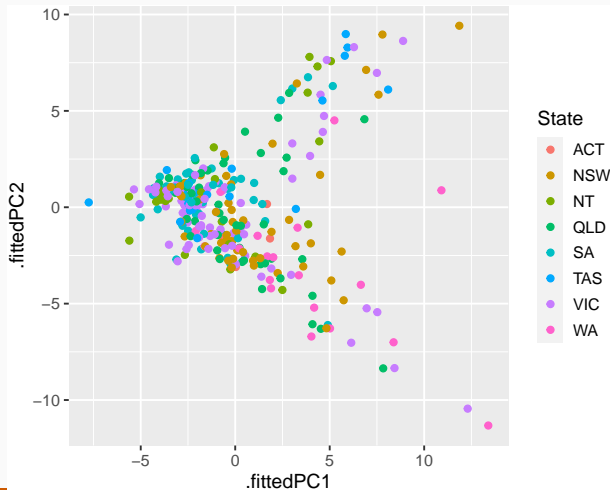
Principal components
based on all features
from the feasts
package



Feature extraction and statistics

```
pcs %>% ggplot(aes(x=.fittedPC1, y=.fittedPC2, col=State)) +  
  geom_point() + theme(aspect.ratio=1)
```

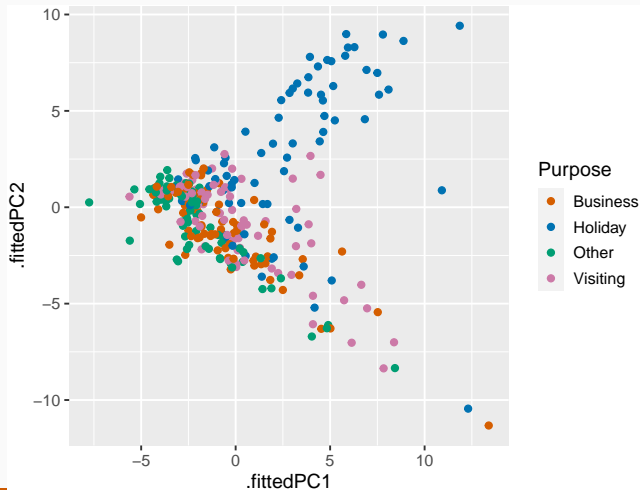
Principal components
based on all features
from the feasts
package



Feature extraction and statistics

```
pcs %>% ggplot(aes(x=.fittedPC1, y=.fittedPC2, col=Purpose)) +  
  geom_point() + theme(aspect.ratio=1)
```

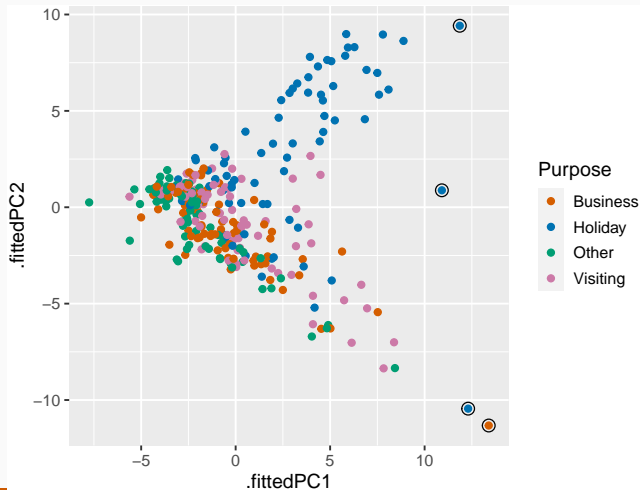
Principal components
based on all features
from the feasts
package



Feature extraction and statistics

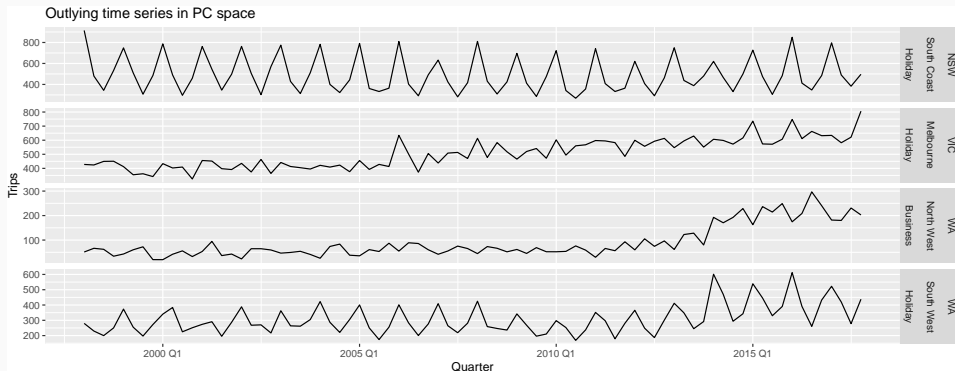
```
pcs %>% ggplot(aes(x=.fittedPC1, y=.fittedPC2, col=Purpose)) +  
  geom_point() + theme(aspect.ratio=1)
```

Principal components
based on all features
from the feasts
package



Feature extraction and statistics

```
outliers %>%  
  left_join(tourism, by = c("State", "Region", "Purpose")) %>%  
  mutate(Series = glue("{State}", "{Region}", "{Purpose}", .sep = "\n\n")) %>%  
  ggplot(aes(x = Quarter, y = Trips)) + geom_line() +  
  facet_grid(Series ~ ., scales = "free_y") +  
  ggtitle("Outlying time series in PC space")
```



Outline

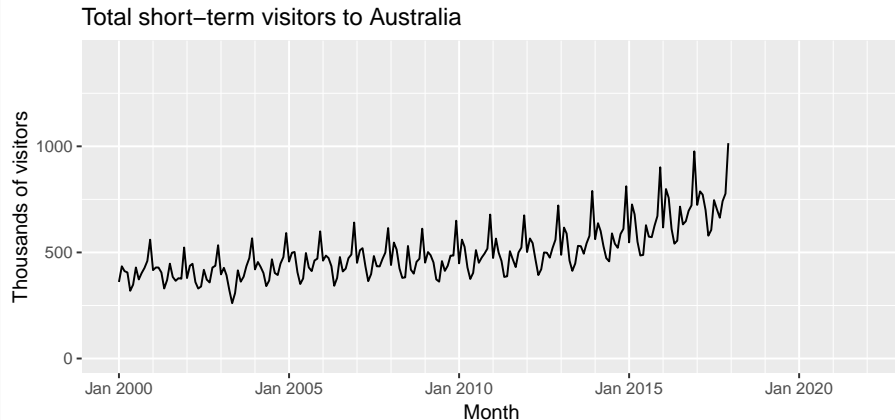
- 1 tsibble: Time series data
- 2 feasts: Data visualization
- 3 feasts: Time series features
- 4 fable: Forecasting
- 5 fable: Evaluating forecast accuracy
- 6 fable: Forecast reconciliation

Random futures

A forecast is an estimate of the probability distribution of a variable to be observed in the future.

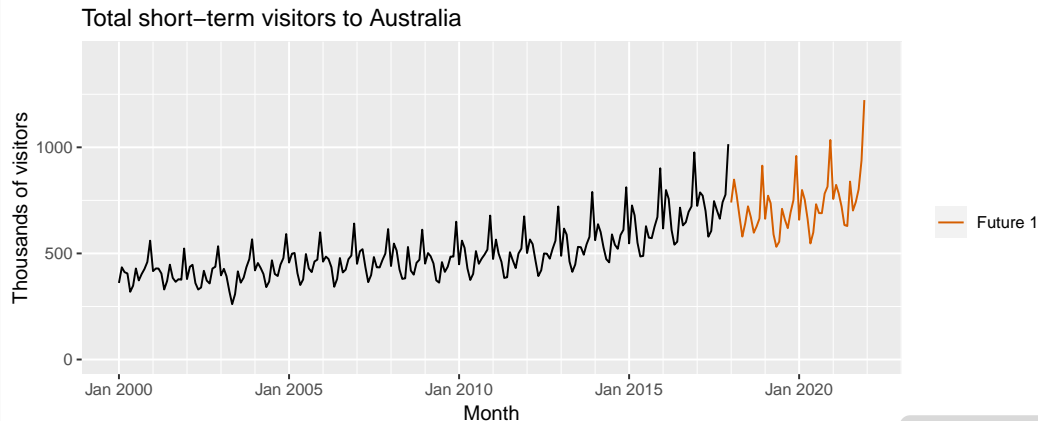
Random futures

A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Random futures

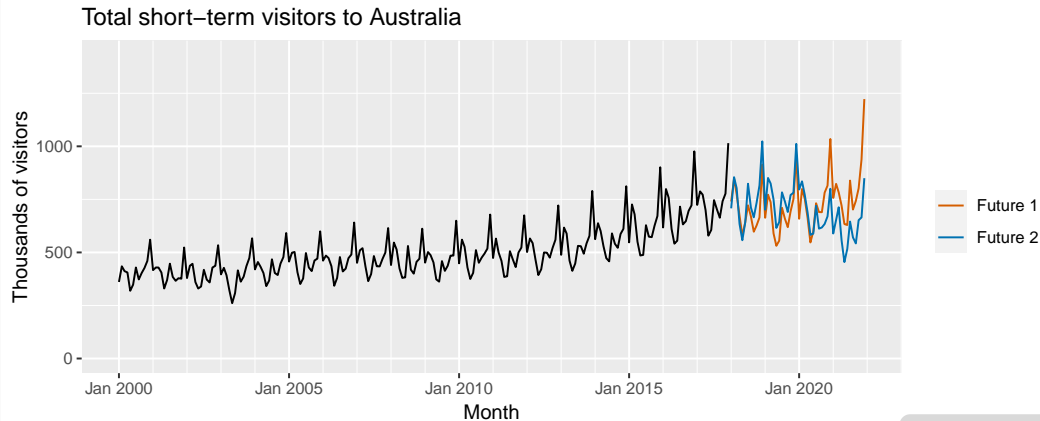
A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Random futures

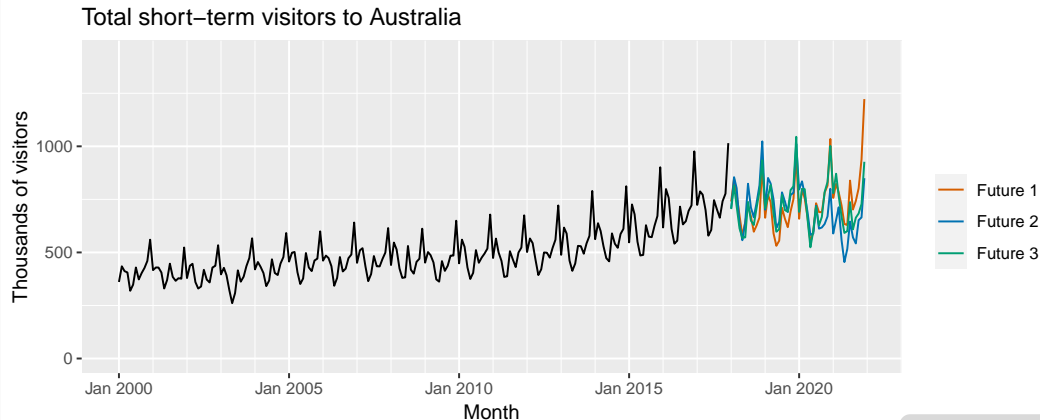
A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Random futures

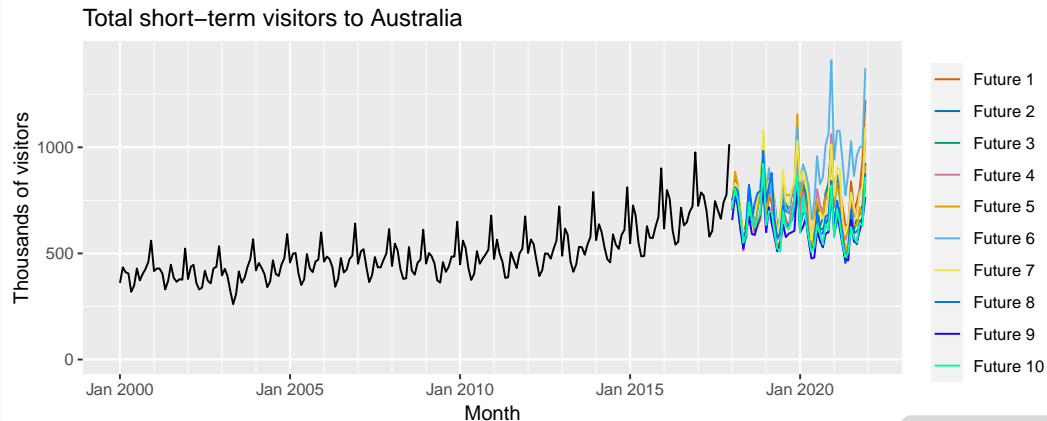
A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Random futures

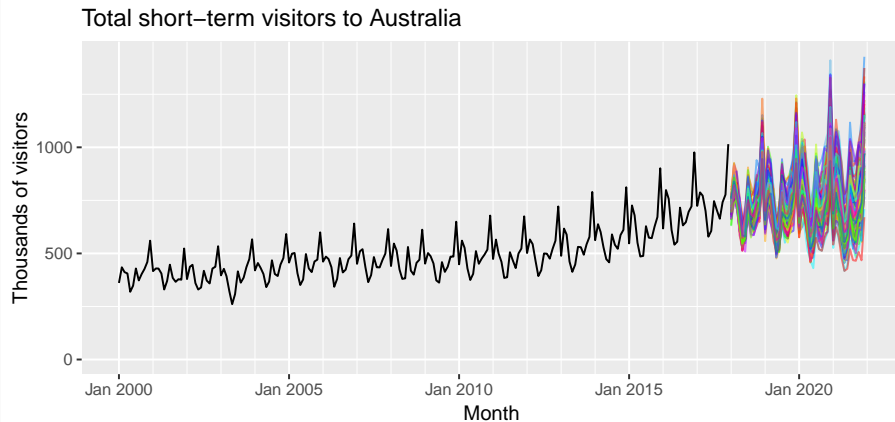
A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Random futures

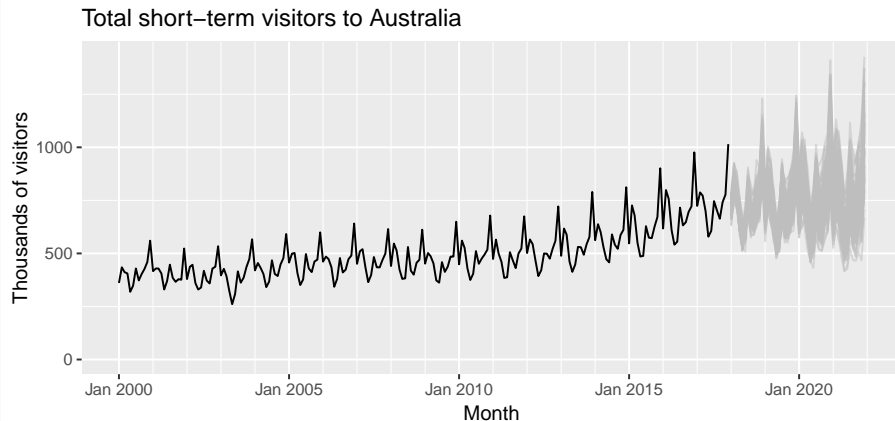
A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Random futures

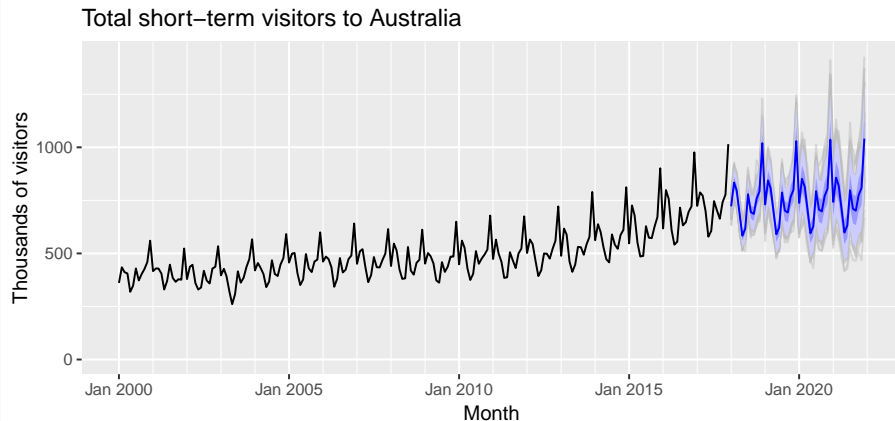
A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Random futures

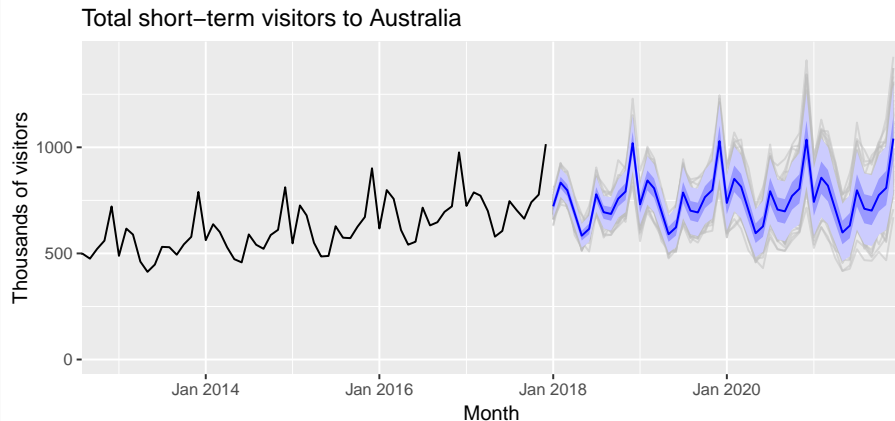
A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Random futures

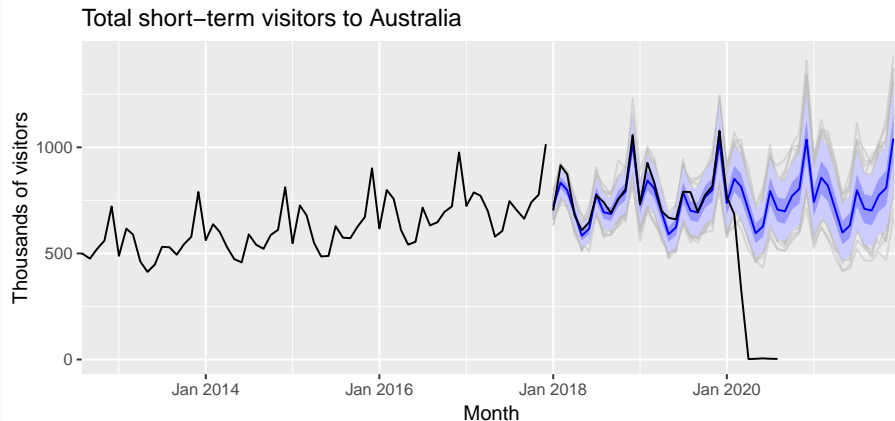
A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Random futures

A forecast is an estimate of the probability distribution of a variable to be observed in the future.



Simulated futures
from an ETS model

Model fitting

```
holiday_fit <- holidays %>%  
  model(  
    snaive = SNAIVE(Trips),  
    naive = NAIVE(Trips),  
    ets = ETS(Trips),  
    arima = ARIMA(Trips)  
  )
```

```
## # A mable: 8 x 5  
## # Key:      State [8]  
##   State   snaive   naive      ets                arima  
##   <chr>   <model> <model>    <model>                <model>  
## 1 ACT    <SNAIVE> <NAIVE> <ETS(M,N,A)>    <ARIMA(0,1,1)(1,0,1)[4]>  
## 2 NSW    <SNAIVE> <NAIVE> <ETS(M,N,A)>    <ARIMA(0,1,1)(0,1,1)[4]>  
## 3 NT     <SNAIVE> <NAIVE> <ETS(M,N,A)>    <ARIMA(0,0,0)(2,1,0)[4]>  
## 4 QLD    <SNAIVE> <NAIVE> <ETS(A,N,A)> <ARIMA(0,0,0)(1,0,0)[4] w/ mean>
```

Model fitting

```
holiday_fit %>%  
  filter(State == "VIC") %>%  
  select(arima) %>%  
  report()
```

```
## Series: Trips  
## Model: ARIMA(0,1,1)(0,1,1)[4]  
##  
## Coefficients:  
##          ma1      sma1  
##      -0.8079  -0.5758  
## s.e.   0.0802   0.1640  
##  
## sigma^2 estimated as 25711:  log likelihood=-487.82  
## AIC=981.64   AICc=981.98   BIC=988.6
```

Model fitting

```
glance(holiday_fit)
```

```
## # A tibble: 32 x 12
```

##	State	.model	sigma2	log_lik	AIC	AICc	BIC	MSE	AMSE	MAE	ar_roots
##	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<list>
##	1 ACT	snaive	2.15e+3	NA	NA	NA	NA	NA	NA	NA	<NULL>
##	2 ACT	naive	2.93e+3	NA	NA	NA	NA	NA	NA	NA	<NULL>
##	3 ACT	ets	6.80e-2	-463.	940.	941.	956.	1509.	1538.	0.189	<NULL>
##	4 ACT	arima	1.44e+3	-399.	805.	806.	815.	NA	NA	NA	<cpl [4~
##	5 NSW	snaive	4.65e+4	NA	NA	NA	NA	NA	NA	NA	<NULL>
##	6 NSW	naive	2.54e+5	NA	NA	NA	NA	NA	NA	NA	<NULL>
##	7 NSW	ets	3.47e-3	-585.	1184.	1185.	1201.	28271.	29225.	0.0458	<NULL>
##	8 NSW	arima	3.30e+4	-498.	1003.	1003.	1010.	NA	NA	NA	<cpl [0~
##	9 NT	snaive	2.45e+3	NA	NA	NA	NA	NA	NA	NA	<NULL>
##	10 NT	naive	2.11e+4	NA	NA	NA	NA	NA	NA	NA	<NULL>
##	#	... with 22 more rows, and 1 more variable: ma_roots <list>									

Model fitting

```
tidy(holiday_fit)
```

```
## # A tibble: 75 x 7
##   State .model term   estimate std.error statistic    p.value
##   <chr> <chr>  <chr>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 ACT   ets    alpha    0.0513     NA          NA         NA
## 2 ACT   ets    gamma    0.482      NA          NA         NA
## 3 ACT   ets    l[0]    148.        NA          NA         NA
## 4 ACT   ets    s[0]     7.88       NA          NA         NA
## 5 ACT   ets    s[-1]   -2.66       NA          NA         NA
## 6 ACT   ets    s[-2]  -20.5       NA          NA         NA
## 7 ACT   ets    s[-3]   15.3       NA          NA         NA
## 8 ACT   arima  ma1     -0.886     0.0668     -13.3      8.33e-22
## 9 ACT   arima  sar1     0.773     0.170       4.55      1.93e- 5
## 10 ACT  arima  sma1    -0.557     0.220      -2.52      1.36e- 2
## # ... with 65 more rows
```

Model fitting

```
augment(holiday_fit)
```

```
## # A tsibble: 2,560 x 7 [1Q]
## # Key:           State, .model [32]
##   State .model Quarter Trips .fitted .resid .innov
##   <chr> <chr>    <qtr> <dbl>   <dbl> <dbl> <dbl>
## 1 ACT   snaive 1998 Q1  196.    NA    NA    NA
## 2 ACT   snaive 1998 Q2  127.    NA    NA    NA
## 3 ACT   snaive 1998 Q3  111.    NA    NA    NA
## 4 ACT   snaive 1998 Q4  170.    NA    NA    NA
## 5 ACT   snaive 1999 Q1  108.   196. -88.4 -88.4
## 6 ACT   snaive 1999 Q2  125.   127.  -2.13 -2.13
## 7 ACT   snaive 1999 Q3  178.   111.  67.3  67.3
## 8 ACT   snaive 1999 Q4  218.   170.  47.2  47.2
## 9 ACT   snaive 2000 Q1  158.   108.  50.6  50.6
## 10 ACT  snaive 2000 Q2  155.   125.  30.2  30.2
```

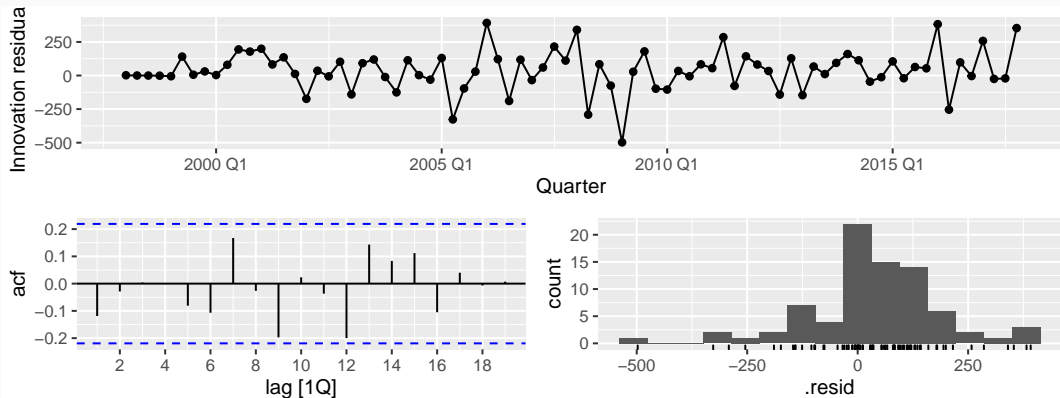
Ljung-Box test

```
augment(holiday_fit) %>%  
  filter(State == "VIC", .model == "arima") %>%  
  features(.resid, ljung_box, dof = 2, lag = 8)
```

```
## # A tibble: 1 x 4  
##   State .model lb_stat lb_pvalue  
##   <chr> <chr>    <dbl>    <dbl>  
## 1 VIC   arima      5.42     0.492
```


gg_tsresiduals() function

```
holiday_fit %>%  
  filter(State == "VIC") %>%  
  select(arima) %>%  
  gg_tsresiduals()
```



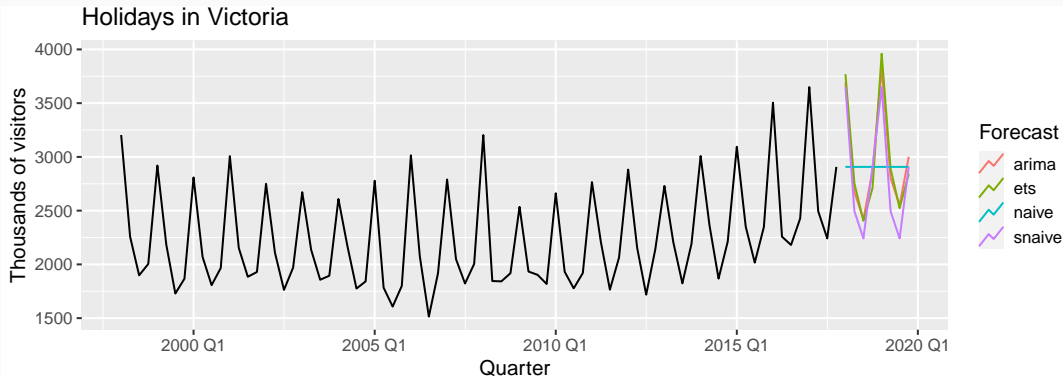
Producing forecasts

```
holiday_fc <- holiday_fit %>%  
  forecast(h = "2 years")
```

```
## # A fable: 256 x 5 [1Q]  
## # Key:      State, .model [32]  
##   State .model Quarter      Trips .mean  
##   <chr> <chr>   <qtr>      <dist> <dbl>  
## 1 ACT   snaive  2018 Q1  N(223, 2128)  223.  
## 2 ACT   snaive  2018 Q2  N(193, 2128)  193.  
## 3 ACT   snaive  2018 Q3  N(204, 2128)  204.  
## 4 ACT   snaive  2018 Q4  N(214, 2128)  214.  
## 5 ACT   snaive  2019 Q1  N(223, 4257)  223.  
## 6 ACT   snaive  2019 Q2  N(193, 4257)  193.  
## 7 ACT   snaive  2019 Q3  N(204, 4257)  204.  
## 8 ACT   snaive  2019 Q4  N(214, 4257)  214.  
## 9 ACT   naive   2018 Q1  N(214, 2894)  214.  
## 10 ACT  naive   2018 Q2  N(214, 5788)  214.  
## # ... with 246 more rows
```

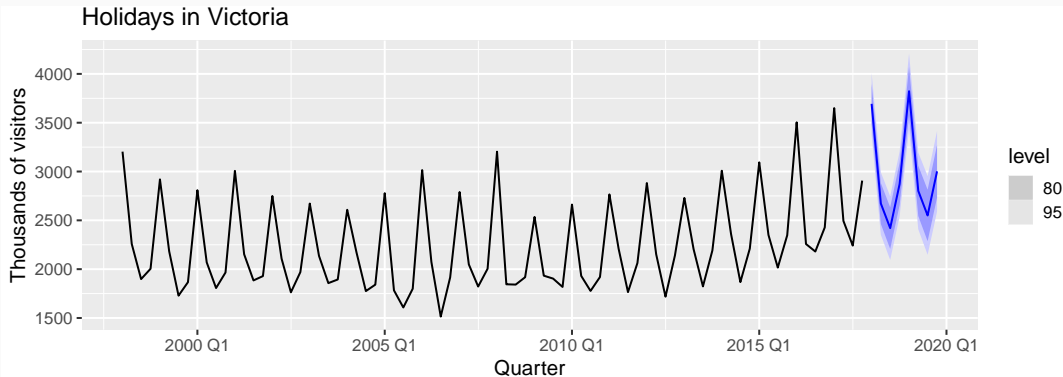
Visualising forecasts

```
holiday_fc %>%  
  filter(State == "VIC") %>%  
  autoplot(holidays, level = NULL) +  
  labs(title = "Holidays in Victoria", y = "Thousands of visitors") +  
  guides(colour = guide_legend(title = "Forecast"))
```



Visualising forecasts

```
holiday_fc %>%  
  filter(State == "VIC", .model == "arima") %>%  
  autoplot(holidays) +  
  labs(title = "Holidays in Victoria", y = "Thousands of visitors") +  
  guides(colour = guide_legend(title = "Forecast"))
```



Prediction intervals

```
holiday_fc %>% hilo(level = 95)
```

```
## # A tsibble: 256 x 6 [1Q]
## # Key:           State, .model [32]
##   State .model Quarter      Trips .mean                `95%`
##   <chr> <chr>    <qtr>      <dist> <dbl>                <hilo>
## 1 ACT    snaive  2018 Q1  N(223, 2128)  223. [132.71289, 313.5540]95
## 2 ACT    snaive  2018 Q2  N(193, 2128)  193. [103.06575, 283.9069]95
## 3 ACT    snaive  2018 Q3  N(204, 2128)  204. [113.21396, 294.0551]95
## 4 ACT    snaive  2018 Q4  N(214, 2128)  214. [124.04311, 304.8842]95
## 5 ACT    snaive  2019 Q1  N(223, 4257)  223. [ 95.25947, 351.0074]95
## 6 ACT    snaive  2019 Q2  N(193, 4257)  193. [ 65.61234, 321.3603]95
## 7 ACT    snaive  2019 Q3  N(204, 4257)  204. [ 75.76054, 331.5085]95
## 8 ACT    snaive  2019 Q4  N(214, 4257)  214. [ 86.58969, 342.3376]95
## 9 ACT    naive   2018 Q1  N(214, 2894)  214. [109.02634, 319.9010]95
```

Prediction intervals

- Point forecasts often useless without a measure of uncertainty (such as prediction intervals).
- Prediction intervals require a stochastic model (with random errors, etc).
- For most models, prediction intervals get wider as the forecast horizon increases.
- Use `level` argument to control coverage.
- Check residual assumptions before believing them.
- Usually too narrow due to unaccounted uncertainty.

Outline

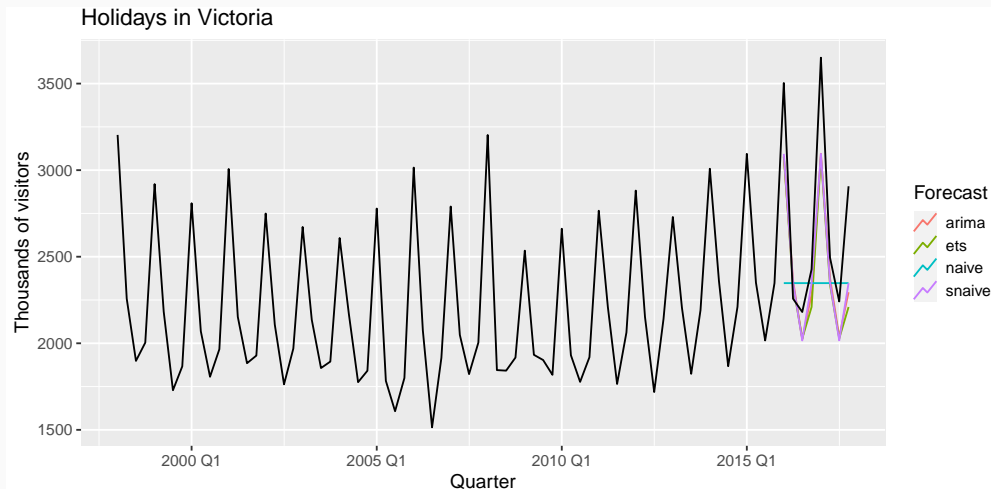
- 1 tsibble: Time series data
- 2 feasts: Data visualization
- 3 feasts: Time series features
- 4 fable: Forecasting
- 5 fable: Evaluating forecast accuracy**
- 6 fable: Forecast reconciliation

Training and test sets



- A model which fits the training data well will not necessarily forecast well.
- A perfect fit can always be obtained by using a model with enough parameters.
- Over-fitting a model to data is just as bad as failing to identify a systematic pattern in the data.
- The test set must not be used for *any* aspect of model development or calculation of forecasts.
- Forecast accuracy is based only on the test set.

Measures of forecast accuracy



Measures of forecast accuracy

```
accuracy(vic_fc, holidays)
```

```
## # A tibble: 4 x 11
##   .model State .type    ME  RMSE  MAE  MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 arima   VIC   Test  263.  349.  290.  8.83  10.0   2.28  2.07 -0.230
## 2 ets     VIC   Test  295.  378.  315. 10.0   10.9   2.48  2.25 -0.149
## 3 naive   VIC   Test  360.  654.  451. 10.1   14.2   3.54  3.88 -0.181
## 4 snaive  VIC   Test  256.  336.  279.  8.59   9.61   2.19  1.99 -0.257
```

Time series cross-validation

Traditional evaluation

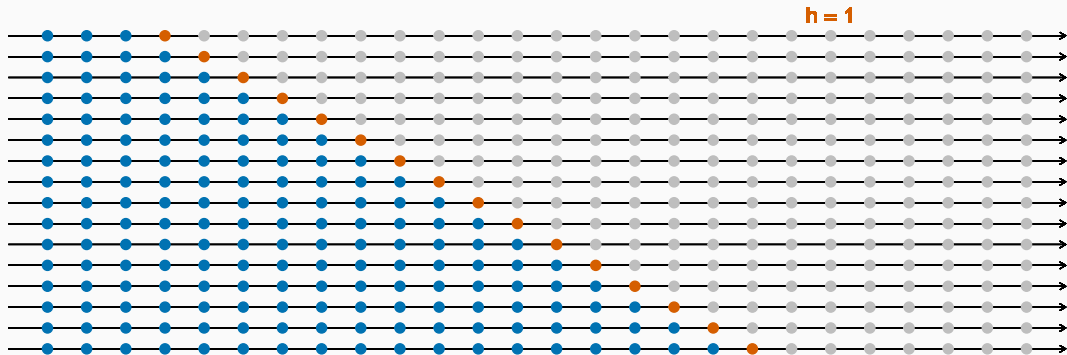


Time series cross-validation

Traditional evaluation



Time series cross-validation

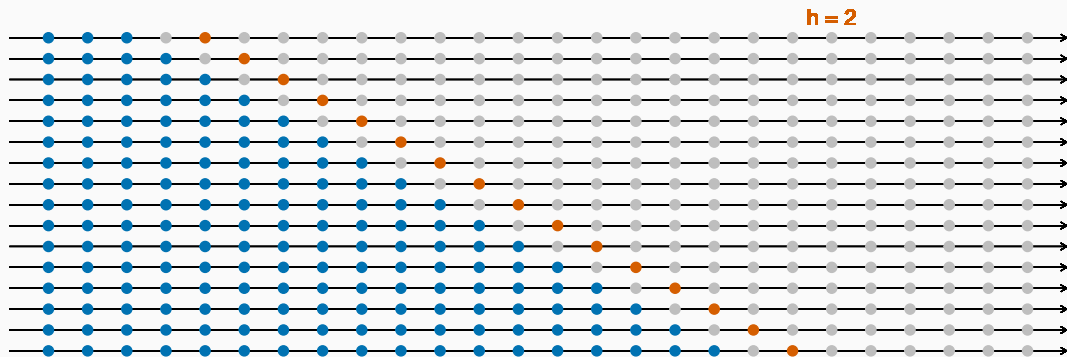


Time series cross-validation

Traditional evaluation



Time series cross-validation

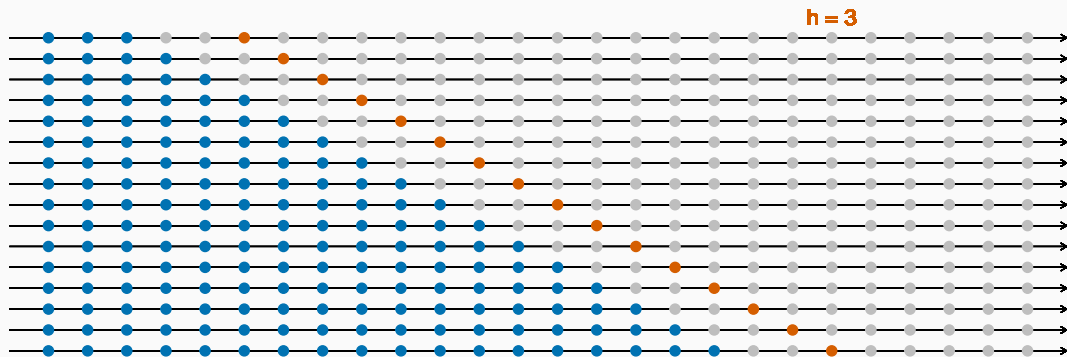


Time series cross-validation

Traditional evaluation



Time series cross-validation

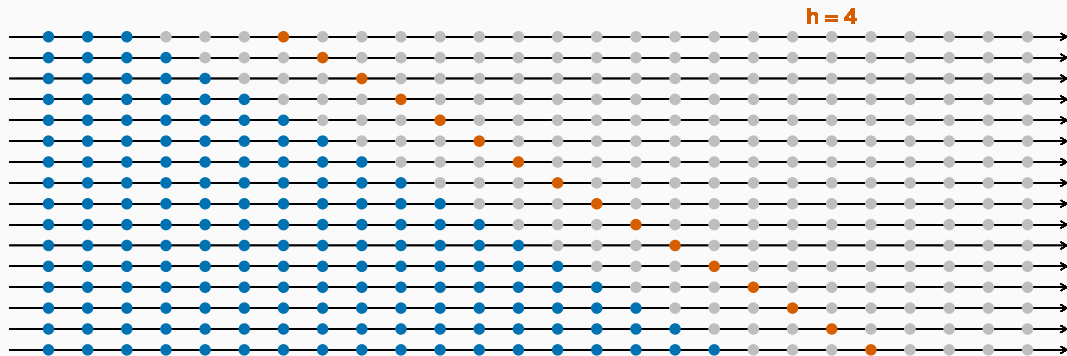


Time series cross-validation

Traditional evaluation



Time series cross-validation

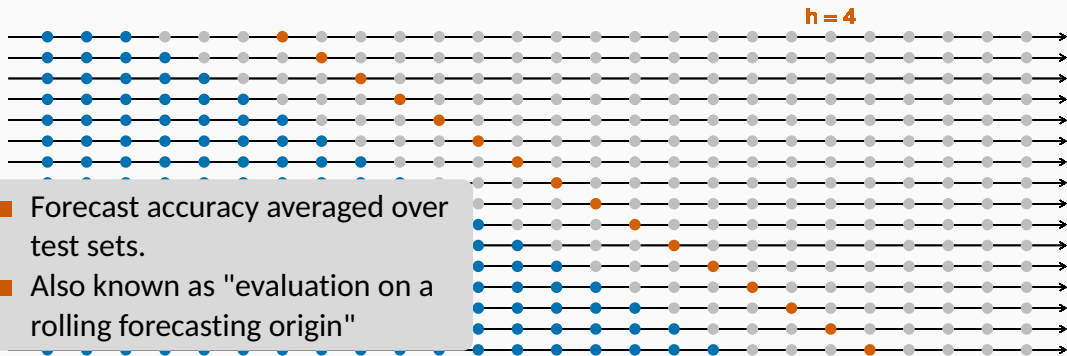


Time series cross-validation

Traditional evaluation



Time series cross-validation



Time series cross-validation

Stretch with a minimum length of 4 years, growing by 1 quarter each step.

```
vic_holiday_stretch <- holidays %>%  
  filter(State == "VIC") %>%  
  stretch_tsibble(.init = 16, .step = 1)
```

```
## # A tsibble: 3,120 x 4 [1Q]  
## # Key:           .id, State [65]  
##   State Quarter Trips   .id  
##   <chr>    <qtr> <dbl> <int>  
## 1 VIC     1998 Q1 3204.     1  
## 2 VIC     1998 Q2 2258.     1  
## 3 VIC     1998 Q3 1898.     1  
## 4 VIC     1998 Q4 2004.     1  
## 5 VIC     1999 Q1 2919.     1  
## 6 VIC     1999 Q2 2183.     1
```

Time series cross-validation

```
fit_cv <- vic_holiday_stretch %>%  
  model(  
    ets = ETS(Trips),  
    arima = ARIMA(Trips),  
    snaive = SNAIVE(Trips)  
  )
```

```
## # A mable: 65 x 5
```

```
## # Key:      .id, State [65]
```

##	.id	State	ets	arima	snaive
##	<int>	<chr>	<model>	<model>	<model>
## 1	1	VIC	<ETS(M,N,M)>	<ARIMA(1,0,0) (0,1,0) [4]>	<SNAIVE>
## 2	2	VIC	<ETS(M,N,M)>	<ARIMA(1,0,0) (0,1,0) [4]>	<SNAIVE>
## 3	3	VIC	<ETS(M,N,M)>	<ARIMA(1,0,0) (0,1,0) [4]>	<SNAIVE>
## 4	4	VIC	<ETS(M,N,M)>	<ARIMA(0,0,2) (0,1,0) [4]>	<SNAIVE>
## 5	5	VIC	<ETS(M,N,M)>	<ARIMA(0,0,2) (0,1,0) [4]>	<SNAIVE>

Time series cross-validation

Produce one step ahead forecasts from all models.

```
fc_cv <- fit_cv %>%  
  forecast(h = 1)
```

```
## # A fable: 195 x 6 [1Q]  
## # Key:      .id, State, .model [195]  
##      .id State Quarter      Trips .mean .model  
##    <int> <chr>   <qtr>      <dist> <dbl> <chr>  
## 1      1 VIC    2002 Q1 N(2965, 14682) 2965. ets  
## 2      1 VIC    2002 Q1 N(2980, 11447) 2980. arima  
## 3      1 VIC    2002 Q1 N(3007, 19035) 3007. snaive  
## 4      2 VIC    2002 Q2  N(2024, 8130) 2024. ets  
## 5      2 VIC    2002 Q2 N(1961, 14864) 1961. arima  
## 6      2 VIC    2002 Q2 N(2153, 22669) 2153. snaive  
## # ... with 189 more rows
```

Time series cross-validation

```
# Cross-validated
```

```
fc_cv %>% accuracy(holidays)
```

```
## # A tibble: 3 x 11
```

##	.model	State	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
##	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	arima	VIC	Test	46.3	183.	147.	1.57	6.41	1.08	1.02	-0.0615
## 2	ets	VIC	Test	43.3	179.	139.	1.51	6.02	1.02	0.997	-0.110
## 3	snaive	VIC	Test	36.2	186.	138.	1.14	6.01	1.02	1.04	0.0427

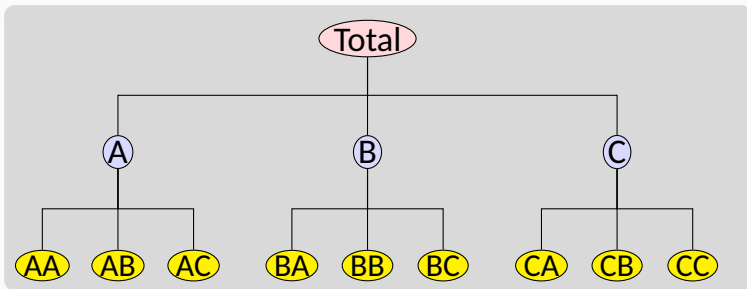
A good way to choose the best forecasting model is to find the model with the smallest RMSE computed using time series cross-validation.

Outline

- 1 tsibble: Time series data
- 2 feasts: Data visualization
- 3 feasts: Time series features
- 4 fable: Forecasting
- 5 fable: Evaluating forecast accuracy
- 6 fable: Forecast reconciliation

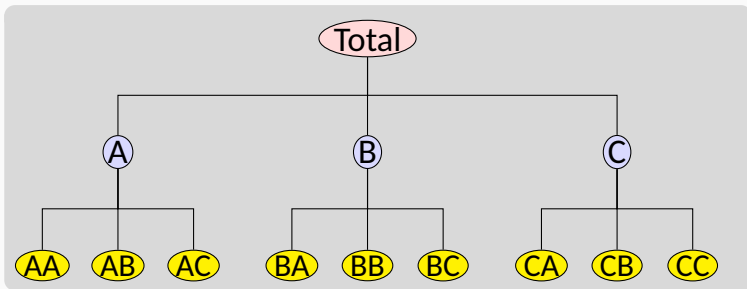
Hierarchical time series

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.

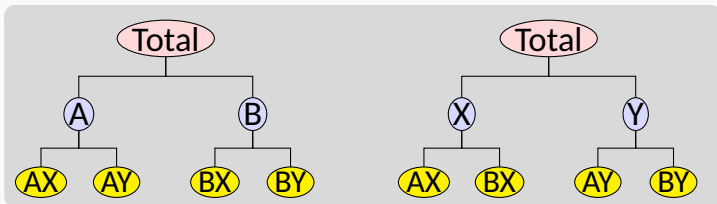


Examples

- Tourism demand by states, zones, regions

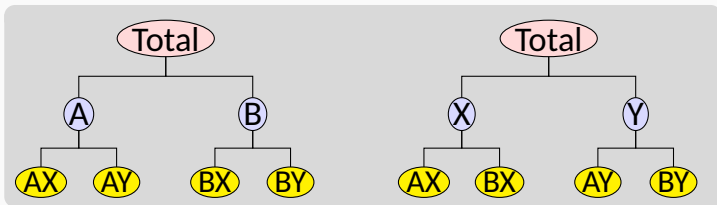
Grouped time series

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)) %>%  
  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.
```

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.

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 tibble: 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.
```

Hierarchical and grouped time series

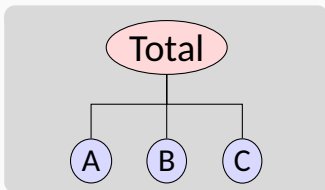
Every collection of time series with aggregation constraints can be written as

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

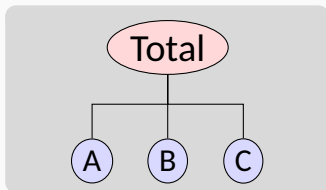
where

- \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

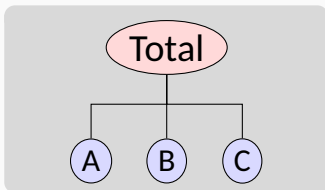


y_t : observed aggregate of all series at time t .

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

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

Hierarchical time series



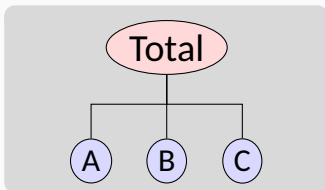
y_t : observed aggregate of all series at time t .

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

\mathbf{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 .

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

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

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 $\mathbf{G} = (\mathbf{S}'\mathbf{W}_h^{-1}\mathbf{S})^{-1}\mathbf{S}'\mathbf{W}_h^{-1}$, where \mathbf{W}_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 $\mathbf{G} = (\mathbf{S}'\mathbf{W}_h^{-1}\mathbf{S})^{-1}\mathbf{S}'\mathbf{W}_h^{-1}$, where \mathbf{W}_h is the h -step base forecast error covariance matrix.

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

Problem: \mathbf{W}_h hard to estimate, especially for $h > 1$.

Solutions:

- Ignore \mathbf{W}_h (OLS) [`min_trace(method='ols')`]
- Assume $\mathbf{W}_h = k_h\mathbf{W}_1$ is diagonal (WLS) [`min_trace(method='wls')`]
- Assume $\mathbf{W}_h = k_h\mathbf{W}_1$ and estimate it (GLS)

Features

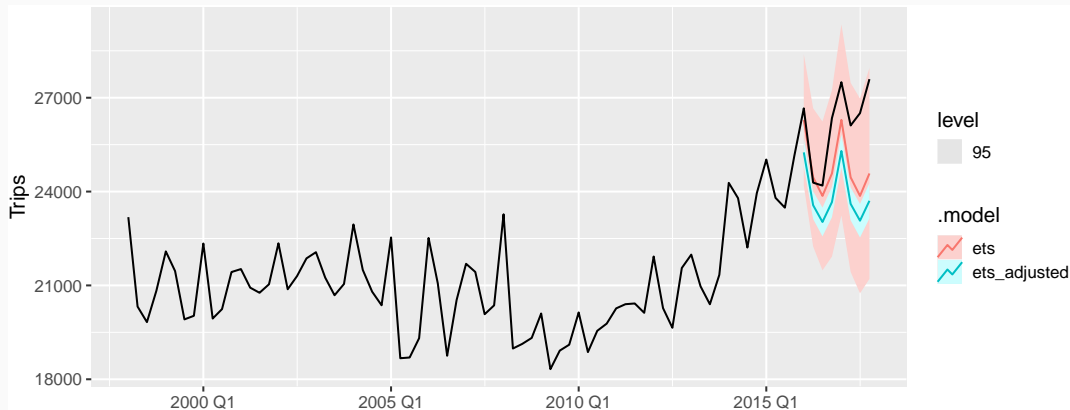
- Covariates can be included in initial forecasts.
- Adjustments can be made to initial forecasts at any level.
- Very simple and flexible method. Can work with *any* hierarchical or grouped time series.
- Conceptually easy to implement: regression of base forecasts on structure matrix.

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")
```

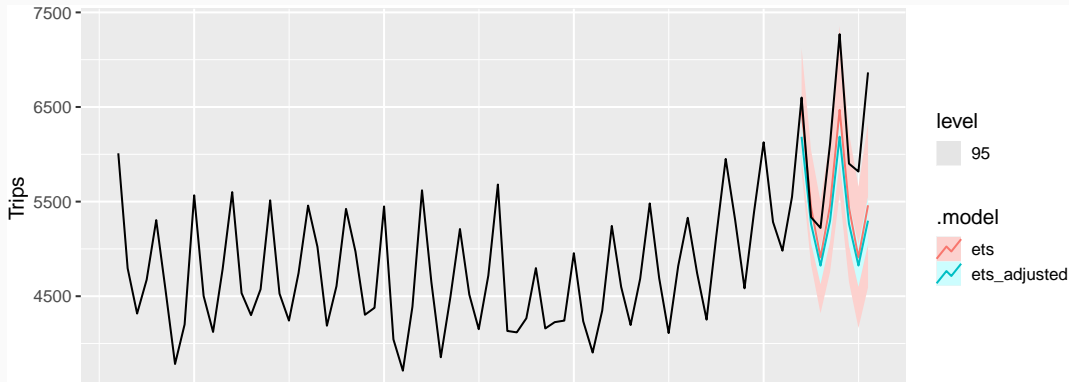
Example: Australian tourism

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



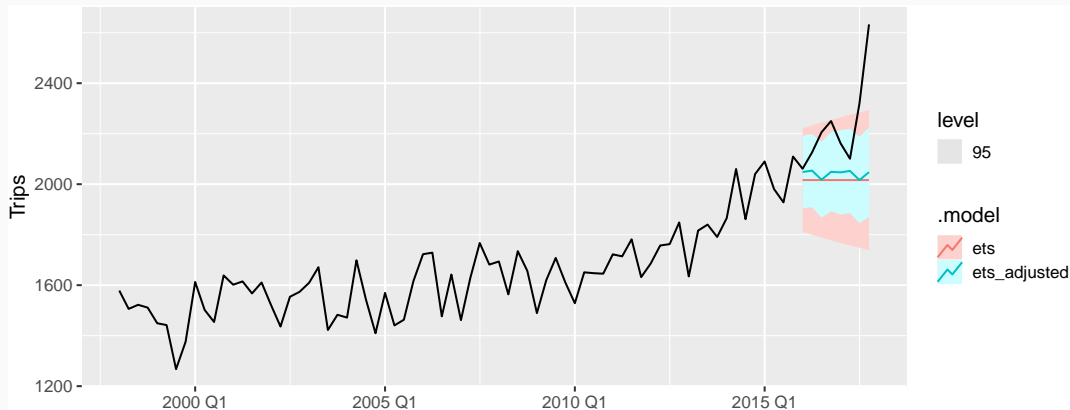
Example: Australian tourism

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



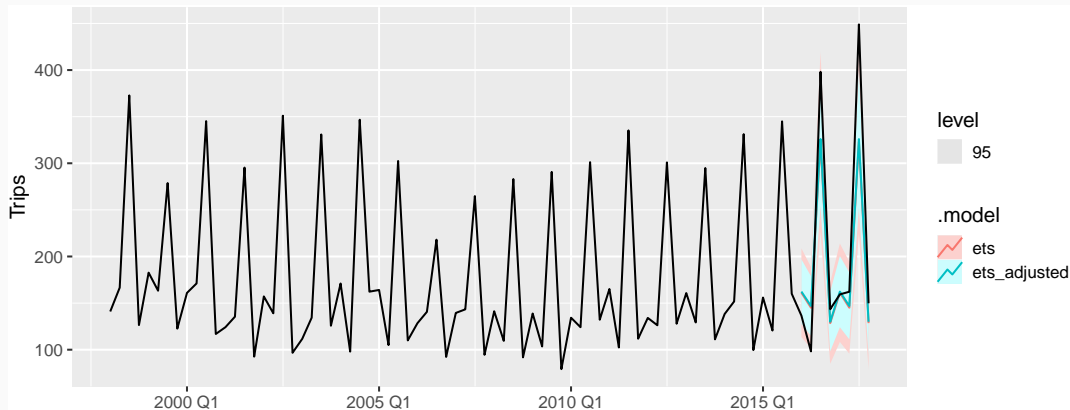
Example: Australian tourism

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



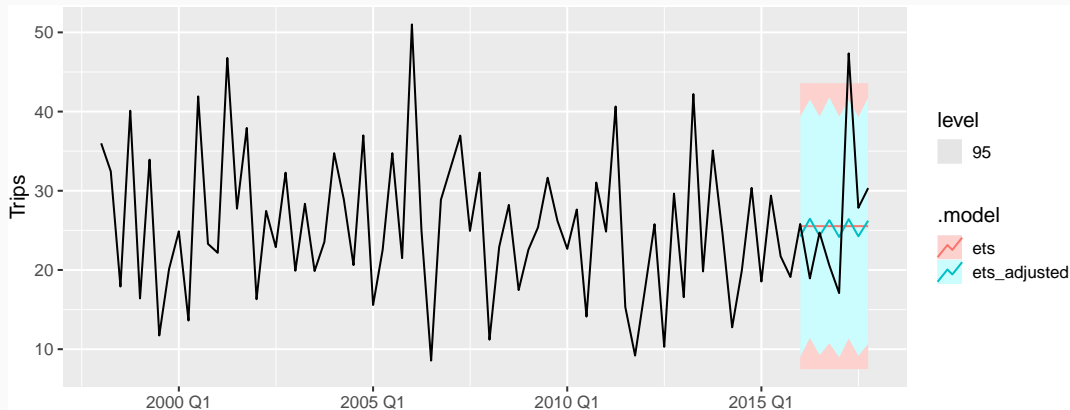
Example: Australian tourism

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



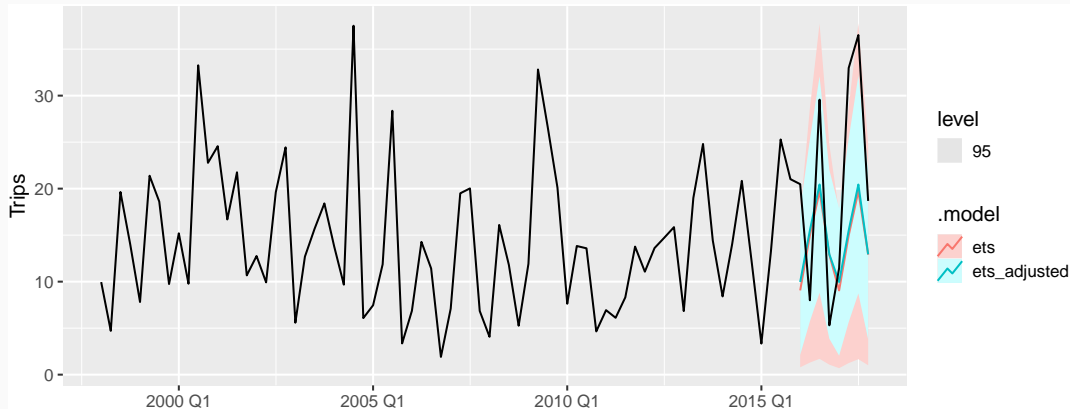
Example: Australian tourism

```
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>	
##	1	arma	Business	ACT	Canberra	~ Test	35.9	45.7	35.9	16.9	16.9
##	2	arma	Business	ACT	<aggregated>	Test	35.9	45.7	35.9	16.9	16.9
##	3	arma	Business	NSW	Blue Mountains	~ Test	1.93	10.6	8.52	-	
	18.0	48.6									
##	4	arma	Business	NSW	Capital Country	~ Test	8.08	15.6	10.4	11.8	19.0
##	5	arma	Business	NSW	Central Coast	~ Test	10.0	14.5	10.8	26.9	32.2
##	6	arma	Business	NSW	Central NSW	~ Test	17.7	31.9	28.2	12.0	24.1
##	7	arma	Business	NSW	Hunter	~ Test	35.3	43.9	35.3	24.2	24.2
##	8	arma	Business	NSW	New England North	~ Test	23.1	31.8	26.8	19.5	28.0
##	9	arma	Business	NSW	North Coast NSW	~ Test	24.8	40.1	36.8	11.5	28.5
##	10	arma	Business	NSW	Outback NSW	~ Test	6.87	11.0	7.76	13.7	16.5

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
```

Tidyverts developers

Earo Wang



Mitchell O'Hara-Wild



More information

- Slides and papers: **robjhyndman.com**
- Packages: **tidyverts.org**
- Forecasting textbook using fable package:
OTexts.com/fpp3

Find me at ...



@robjhyndman



@robjhyndman



robjhyndman.com



rob.hyndman@monash.edu