MONASH University

MONASH BUSINESS SCHOOL

Tidy forecasting in R

Rob J Hyndman

26 September 2019

Outline

Outline

Tidyverts packages

tidyverts.org

tsibble

tsibbledata

feasts

fable

Time series data

- Four-yearly Olympic winning times
- Annual Google profits
- Quarterly Australian snowy production
- Monthly rainfall
- Weekly retail sales
- Daily IBM stock prices
- Hourly electricity demand
- 5-minute freeway traffic counts
- Time-stamped stock transaction data

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
## # ... with 15,140 more rows
```

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
## # ... with 15,140 more rows
```

tsibble objects

```r
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
## # ... with 15,140 more rows
```

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  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
## # ... with 15,140 more rows
```

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

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

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

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

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

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.

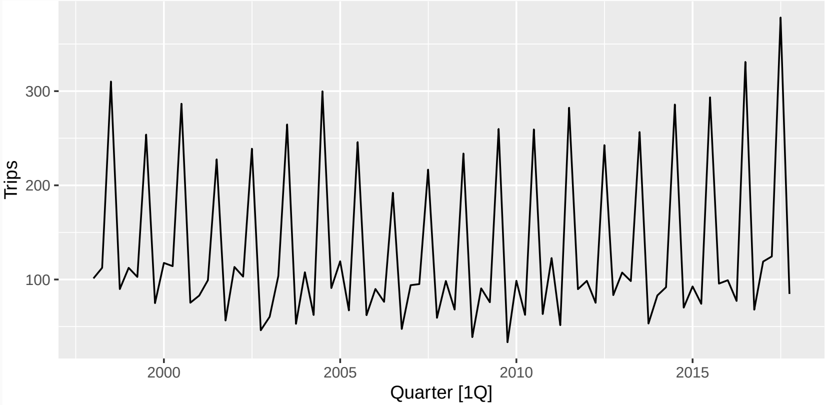
Extracting single time series

```r
snowy <- tourism %>%
  filter(
    Region=="Snowy Mountains",
    Purpose=="Holiday"
  )
snowy
```

```
## # A tsibble: 80 x 5 [1Q]
## # Key:       Region, State, Purpose [1]
##    Quarter Region          State Purpose Trips
##      <qtr> <chr>           <chr> <chr>   <dbl>
##  1 1998 Q1 Snowy Mountains NSW   Holiday 101.
##  2 1998 Q2 Snowy Mountains NSW   Holiday 112.
##  3 1998 Q3 Snowy Mountains NSW   Holiday 310.
##  4 1998 Q4 Snowy Mountains NSW   Holiday  89.8
##  5 1999 Q1 Snowy Mountains NSW   Holiday 112.
```

Extracting single time series

```r
snowy %>% autoplot(Trips)
```



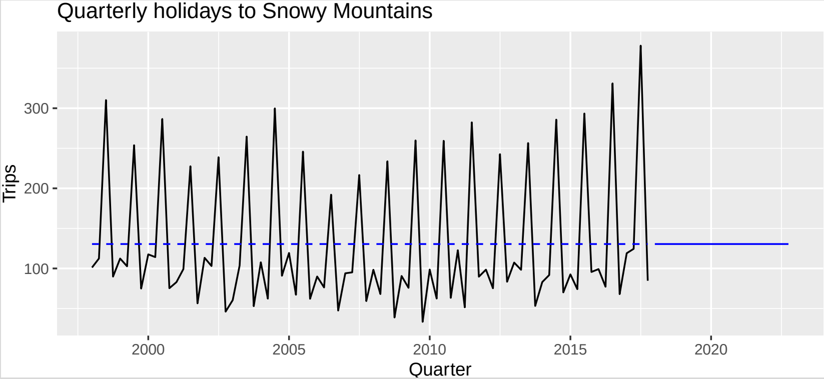
How would you forecast these series?

Outline

Benchmark forecasting methods

Mean method

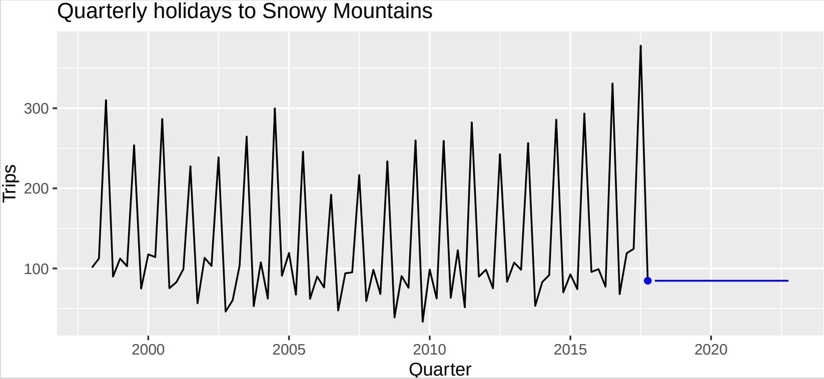
- Forecast of all future values is equal to mean of historical data $\{y_1, \ldots, y_T\}$.
- Forecasts: $\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T$



Benchmark forecasting methods

Naïve method

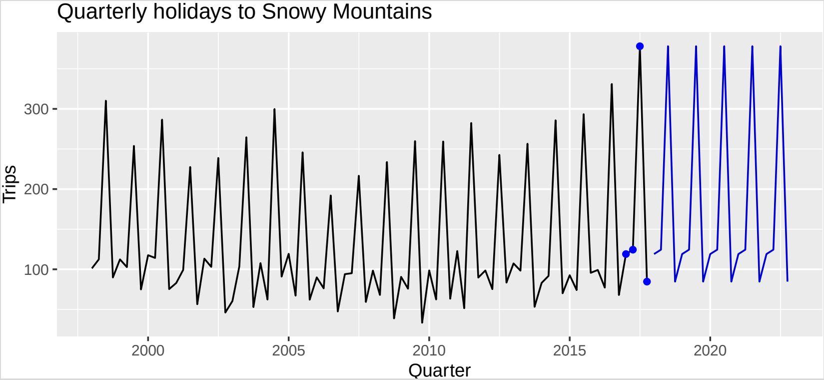
- Forecasts equal to last observed value.
- Forecasts: $\hat{y}_{T+h|T} = y_T$.
- Consequence of efficient market hypothesis.



Benchmark forecasting methods

Seasonal naïve method

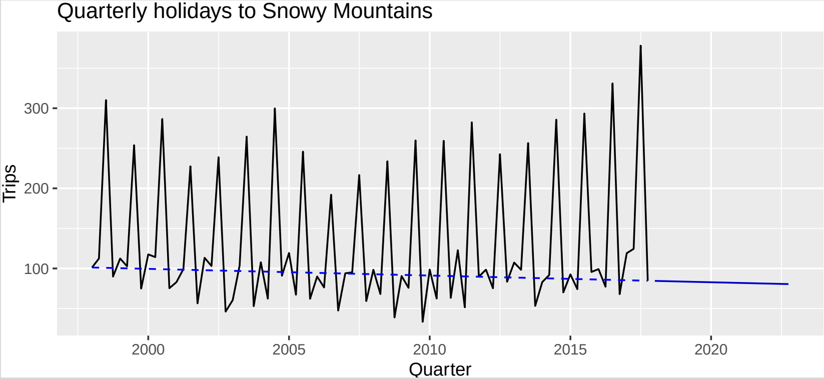
- Forecasts equal to last value from same season.
- Forecasts: $\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$, where m = seasonal period and k is the integer part of $(h-1)/m$.



Benchmark forecasting methods

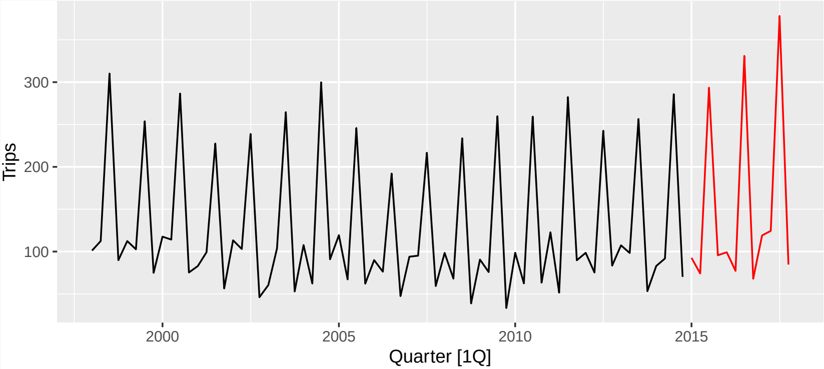
Drift method

- Forecasts equal to last value plus average change.
- Forecasts: $\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}(y_T - y_1)$.
- Equivalent to line between first and last observations.



Data preparation and visualisation

```r
# Set training data from 1970 to 2005
train <- filter(snowy, year(Quarter) <= 2014)
test <-  filter(snowy, year(Quarter) > 2014)
train %>% autoplot(Trips) + geom_line(data=test, col='red')
```



Model estimation

The `model()` function trains models to data.

```
# Fit the models
fit <- train %>%
  model(
    Mean = MEAN(Trips),
    Naïve = NAIVE(Trips),
    SeasonalNaïve = SNAIVE(Trips),
    Drift = RW(Trips ~ drift())
  )
```

```
fit
```

```
## # A mable: 1 x 7
## # Key:     Region, State, Purpose [1]
##   Region    State Purpose Mean    Naïve SeasonalNaïve Drift
##   <chr>     <chr> <chr>   <mode> <mod> <model>       <model>
## 1 Snowy Mo~ NSW   Holiday <MEAN> <NAI~ <SNAIVE>      <RW w/~17
```

Producing forecasts

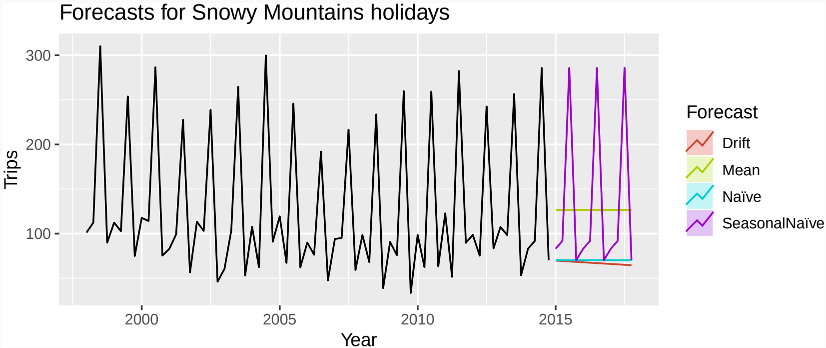
```r
fc <- fit %>%
  forecast(h = 12)
```

```
## # A fable: 48 x 7 [1Q]
## # Key:     Region, State, Purpose, .model [4]
##   Region State Purpose .model    Quarter Trips .distribution
##   <chr>  <chr> <chr>   <chr>       <qtr> <dbl> <dist>
## 1 Snowy~ NSW   Holiday Mean      2015 Q1  126. N(126, 6408)
## 2 Snowy~ NSW   Holiday Mean      2015 Q2  126. N(126, 6408)
## 3 Snowy~ NSW   Holiday Mean      2015 Q3  126. N(126, 6408)
## 4 Snowy~ NSW   Holiday Mean      2015 Q4  126. N(126, 6408)
## # ... with 44 more rows
```

A `fable` is a forecast table with point forecasts and distributions.

Visualising forecasts

```
fc %>%
  autoplot(train, level = NULL) +
  ggtitle("Forecasts for Snowy Mountains holidays") +
  xlab("Year") +
  guides(colour=guide_legend(title="Forecast"))
```



Forecasting many series

```
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.
```

Forecasting many series

```r
tourism %>%
  model(
    mean = MEAN(Trips),
    snaive = SNAIVE(Trips)
  )
```

```
## # A mable: 304 x 5
## # Key:     Region, State, Purpose [304]
##    Region          State Purpose  mean    snaive
##    <chr>           <chr> <chr>    <model> <model>
##  1 Adelaide        SA    Business <MEAN>  <SNAIVE>
##  2 Adelaide        SA    Holiday  <MEAN>  <SNAIVE>
##  3 Adelaide        SA    Other    <MEAN>  <SNAIVE>
##  4 Adelaide        SA    Visiting <MEAN>  <SNAIVE>
##  5 Adelaide Hills  SA    Business <MEAN>  <SNAIVE>
##  6 Adelaide Hills  SA    Holiday  <MEAN>  <SNAIVE>
```

Forecasting many series

```r
tourism %>%
  model(
    mean = MEAN(Trips),
    snaive = SNAIVE(Trips)
  ) %>%
  forecast(h= "3 years")
```

```
## # A fable: 7,296 x 7 [1Q]
## # Key:     Region, State, Purpose, .model [608]
##    Region State Purpose .model    Quarter Trips
##    <chr>  <chr> <chr>   <chr>       <qtr> <dbl>
##  1 Adela~ SA    Busine~ mean      2018 Q1  156.
##  2 Adela~ SA    Busine~ mean      2018 Q2  156.
##  3 Adela~ SA    Busine~ mean      2018 Q3  156.
##  4 Adela~ SA    Busine~ mean      2018 Q4  156.
##  5 Adela~ SA    Busine~ mean      2019 Q1  156.
```

Outline

Historical perspective

- Developed in the 1950s and 1960s as methods (algorithms) to produce point forecasts.
- Combine a “level”, “trend” (slope) and “seasonal” component to describe a time series.
- The rate of change of the components are controlled by “smoothing parameters”: α, β and γ respectively.
- Need to choose best values for the smoothing parameters (and initial states).
- Equivalent ETS state space models developed in the 1990s and 2000s.

ETS models

General notation $\quad \underset{\nearrow\quad\uparrow\quad\nwarrow}{\text{E T S}}$: **E**xponen**T**ial **S**moothing

Error **T**rend **S**eason

Error: Additive (`"A"`) or multiplicative (`"M"`)

ETS models

General notation $\quad$ E T S : **E**xponen**T**ial **S**moothing

$$\begin{array}{ccc} & \text{E} \;\; \text{T} \;\; \text{S} & \\ \nearrow & \uparrow & \nwarrow \\ \textbf{Error} & \textbf{Trend} & \textbf{Season} \end{array}$$

Error: Additive (`"A"`) or multiplicative (`"M"`)

Trend: None (`"N"`), additive (`"A"`), multiplicative (`"M"`), or damped (`"Ad"` or `"Md"`).

ETS models

General notation E T S : **E**xponen**T**ial **S**moothing

↗ ↑ ↖

Error **T**rend **S**eason

Error: Additive (`"A"`) or multiplicative (`"M"`)

Trend: None (`"N"`), additive (`"A"`), multiplicative (`"M"`), or damped (`"Ad"` or `"Md"`).

Seasonality: None (`"N"`), additive (`"A"`) or multiplicative (`"M"`)

Exponential smoothing models

Additive Error

Trend Component		Seasonal Component: N (None)	A (Additive)	M (Multiplicative)
N	(None)	A,N,N	A,N,A	~~A,N,M~~
A	(Additive)	A,A,N	A,A,A	~~A,A,M~~
A_d	(Additive damped)	A,A_d,N	A,A_d,A	~~A,A_d,M~~

Multiplicative Error

Trend Component		Seasonal Component: N (None)	A (Additive)	M (Multiplicative)
N	(None)	M,N,N	M,N,A	M,N,M
A	(Additive)	M,A,N	M,A,A	M,A,M
A_d	(Additive damped)	M,A_d,N	M,A_d,A	M,A_d,M

Automatic forecasting

From Hyndman et al. (IJF, 2002):

1. Apply each model that is appropriate to the data. Optimize parameters and initial values using MLE.
2. Select best method using AICc.
3. Produce forecasts using best method.
4. Obtain forecast intervals using underlying state space model.

- Method performed very well in M3 competition.
- Used as a benchmark in the M4 competition.

Example: Australian population

```r
aus_economy <- global_economy %>% filter(Country == "Australia") %>%
  mutate(Pop = Population/1e6)
fit <- aus_economy %>% model(best = ETS(Pop))
report(fit)
```

```
## Series: Pop
## Model: ETS(A,A,N)
##   Smoothing parameters:
##     alpha = 1
##     beta  = 0.327
##
##   Initial states:
##     l     b
##  10.1 0.222
##
##   sigma^2:  0.0041
##
##   AIC  AICc   BIC
## -77.0 -75.8 -66.7
```

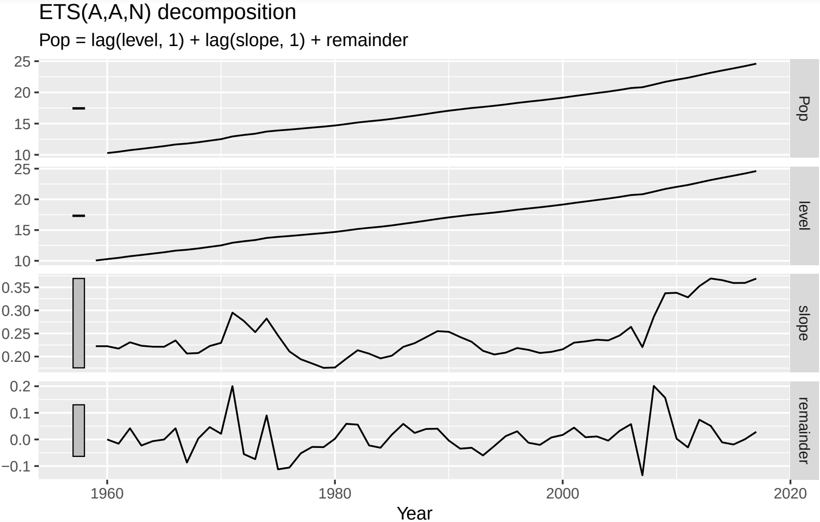
Example: Australian population

```r
components(fit)
```

```
## # A dable:                       59 x 7 [1Y]
## # Key:                           Country, .model [1]
## # ETS(A,A,N) Decomposition: Pop = lag(level, 1) +
## #   lag(slope, 1) + remainder
##    Country   .model  Year   Pop level slope remainder
##    <fct>     <chr>  <dbl> <dbl> <dbl> <dbl>     <dbl>
##  1 Australia best    1959  NA    10.1 0.222 NA
##  2 Australia best    1960  10.3  10.3 0.222 -0.000145
##  3 Australia best    1961  10.5  10.5 0.217 -0.0159
##  4 Australia best    1962  10.7  10.7 0.231  0.0418
##  5 Australia best    1963  11.0  11.0 0.223 -0.0229
##  6 Australia best    1964  11.2  11.2 0.221 -0.00641
##  7 Australia best    1965  11.4  11.4 0.221 -0.000314
##  8 Australia best    1966  11.7  11.7 0.235  0.0418
##  9 Australia best    1967  11.8  11.8 0.206 -0.0869
## 10 Australia best    1968  12.0  12.0 0.208  0.00350
```

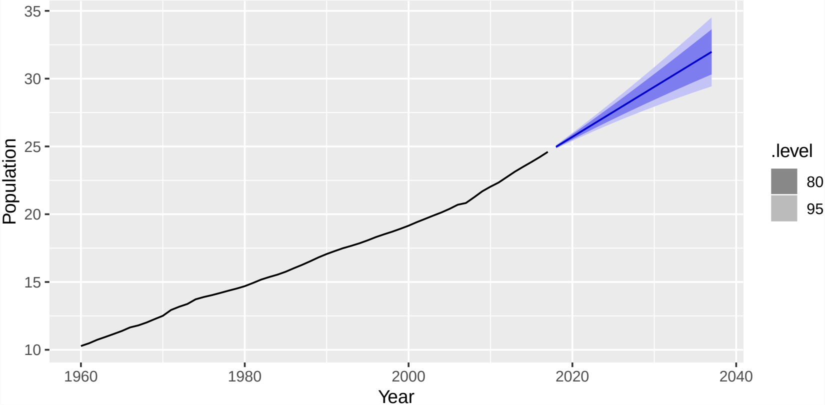
Example: Australian population

```r
components(fit) %>% autoplot()
```



Example: Australian population

```
fit %>%
  forecast(h = 20) %>%
  autoplot(aus_economy) +
  ylab("Population") + xlab("Year")
```



Example: National populations

```r
fit <- global_economy %>%
  mutate(Pop = Population/1e6) %>%
  model(ets = ETS(Pop))
fit
```

```
## # A mable: 263 x 2
## # Key:     Country [263]
##    Country             ets
##    <fct>               <model>
##  1 Afghanistan         <ETS(A,A,N)>
##  2 Albania             <ETS(M,A,N)>
##  3 Algeria             <ETS(M,A,N)>
##  4 American Samoa      <ETS(M,A,N)>
##  5 Andorra             <ETS(M,A,N)>
##  6 Angola              <ETS(M,A,N)>
##  7 Antigua and Barbuda <ETS(M,A,N)>
##  8 Arab World          <ETS(M,A,N)>
##  9 Argentina           <ETS(A,A,N)>
## 10 Armenia             <ETS(M,A,N)>
## # ... with 253 more rows
```

Example: National populations

```r
fit %>%
  forecast(h = 5)
```

```
## # A fable: 1,315 x 5 [1Y]
## # Key:     Country, .model [263]
##    Country     .model  Year   Pop .distribution
##    <fct>       <chr>  <dbl> <dbl> <dist>
##  1 Afghanistan ets     2018 36.4  N(36, 0.012)
##  2 Afghanistan ets     2019 37.3  N(37, 0.059)
##  3 Afghanistan ets     2020 38.2  N(38, 0.164)
##  4 Afghanistan ets     2021 39.0  N(39, 0.351)
##  5 Afghanistan ets     2022 39.9  N(40, 0.644)
##  6 Albania     ets     2018  2.87 N(2.9, 0.00012)
##  7 Albania     ets     2019  2.87 N(2.9, 0.00060)
##  8 Albania     ets     2020  2.87 N(2.9, 0.00169)
##  9 Albania     ets     2021  2.86 N(2.9, 0.00362)
```

Example: Australian holiday tourism

```r
holidays <- tourism %>%
  filter(Purpose == "Holiday")
fit <- holidays %>% model(ets = ETS(Trips))
fit
```

```
## # A mable: 76 x 4
## # Key:     Region, State, Purpose [76]
##    Region                     State Purpose ets
##    <chr>                      <chr> <chr>   <model>
##  1 Adelaide                   SA    Holiday <ETS(A,N,A)>
##  2 Adelaide Hills             SA    Holiday <ETS(A,A,N)>
##  3 Alice Springs              NT    Holiday <ETS(M,N,A)>
##  4 Australia's Coral Coast    WA    Holiday <ETS(M,N,A)>
##  5 Australia's Golden Outback WA    Holiday <ETS(M,N,M)>
##  6 Australia's North West     WA    Holiday <ETS(A,N,A)>
##  7 Australia's South West     WA    Holiday <ETS(M,N,M)>
##  8 Ballarat                   VIC   Holiday <ETS(M,N,A)>
##  9 Barkly                     NT    Holiday <ETS(A,N,A)>
## 10 Barossa                    SA    Holiday <ETS(A,N,N)>
## # … with 66 more rows
```

Example: Australian holiday tourism

```
fit %>% filter(Region=="Snowy Mountains") %>% report()
```

```
## Series: Trips
## Model: ETS(M,N,A)
##   Smoothing parameters:
##     alpha = 0.157
##     gamma = 1e-04
##
##   Initial states:
##     l   s1  s2    s3    s4
##    142 -61 131 -42.2 -27.7
##
##   sigma^2:  0.0388
##
##    AIC AICc  BIC
##    852  854  869
```

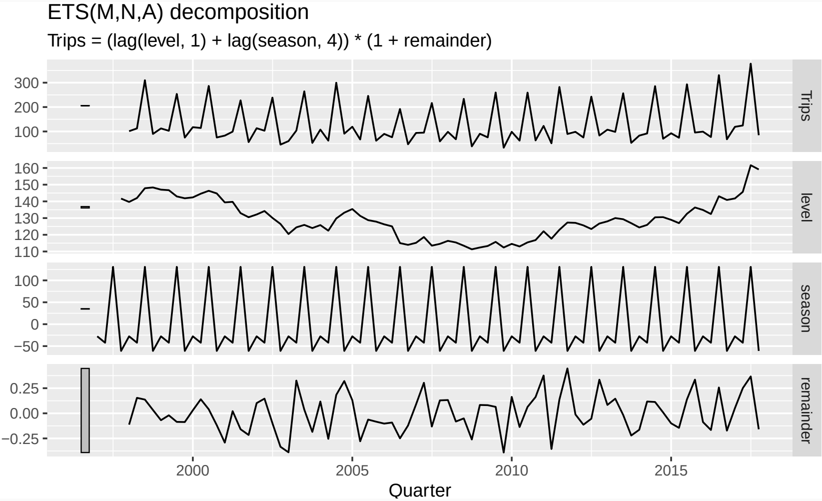
Example: Australian holiday tourism

```
fit %>% filter(Region=="Snowy Mountains") %>% components(fit)
```

```
## # A dable:                  84 x 9 [1Q]
## # Key:                      Region, State, Purpose, .model
## #   [1]
## # ETS(M,N,A) Decomposition: Trips = (lag(level, 1) +
## #   lag(season, 4)) * (1 + remainder)
##    Region State Purpose .model    Quarter Trips level season
##    <chr>  <chr> <chr>   <chr>       <qtr> <dbl> <dbl>  <dbl>
##  1 Snowy~ NSW   Holiday ets       1997 Q1  NA      NA  -27.7
##  2 Snowy~ NSW   Holiday ets       1997 Q2  NA      NA  -42.2
##  3 Snowy~ NSW   Holiday ets       1997 Q3  NA      NA  131.
##  4 Snowy~ NSW   Holiday ets       1997 Q4  NA     142. -61.0
##  5 Snowy~ NSW   Holiday ets       1998 Q1 101.    140. -27.7
##  6 Snowy~ NSW   Holiday ets       1998 Q2 112.    142. -42.2
##  7 Snowy~ NSW   Holiday ets       1998 Q3 310.    148. 131.
##  8 Snowy~ NSW   Holiday ets       1998 Q4  89.8   148. -61.0
##  9 Snowy~ NSW   Holiday ets       1999 Q1 112.    147. -27.7
## 10 Snowy~ NSW   Holiday ets       1999 Q2 103.    147. -42.2
```

Example: Australian holiday tourism

```r
fit %>% filter(Region=="Snowy Mountains") %>%
  components(fit) %>% autoplot()
```



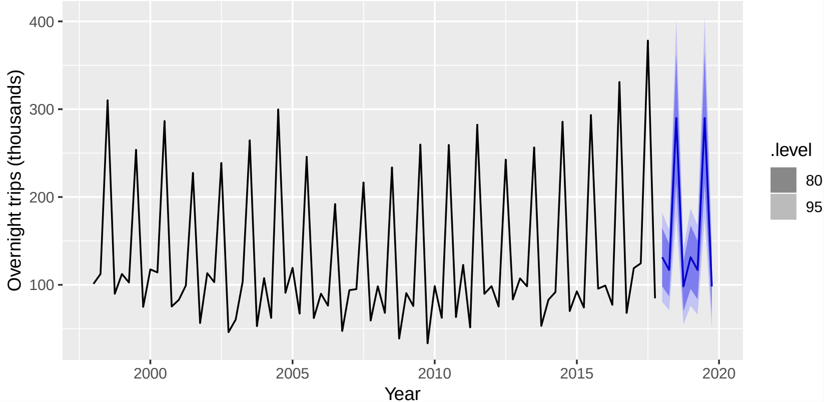
Example: Australian holiday tourism

```r
fit %>% forecast()
```

```
## # A fable: 608 x 7 [1Q]
## # Key:     Region, State, Purpose, .model [76]
##    Region State Purpose .model    Quarter Trips
##    <chr>  <chr> <chr>   <chr>       <qtr> <dbl>
##  1 Adela~ SA    Holiday ets       2018 Q1 210.
##  2 Adela~ SA    Holiday ets       2018 Q2 173.
##  3 Adela~ SA    Holiday ets       2018 Q3 169.
##  4 Adela~ SA    Holiday ets       2018 Q4 186.
##  5 Adela~ SA    Holiday ets       2019 Q1 210.
##  6 Adela~ SA    Holiday ets       2019 Q2 173.
##  7 Adela~ SA    Holiday ets       2019 Q3 169.
##  8 Adela~ SA    Holiday ets       2019 Q4 186.
##  9 Adela~ SA    Holiday ets       2018 Q1  19.4
## 10 Adela~ SA    Holiday ets       2018 Q2  19.6
## # ... with 598 more rows, and 1 more variable:
## #   .distribution <dist>
```

Example: Australian holiday tourism

```
fit %>% forecast() %>%
  filter(Region=="Snowy Mountains") %>%
  autoplot(holidays) +
    xlab("Year") + ylab("Overnight trips (thousands)")
```



Outline

ARIMA models

AR: autoregressive (lagged observations as inputs)
I: integrated (differencing to make series stationary)
MA: moving average (lagged errors as inputs)

ARIMA models

AR:	autoregressive (lagged observations as inputs)
I:	integrated (differencing to make series stationary)
MA:	moving average (lagged errors as inputs)

An ARIMA model is rarely interpretable in terms of visible data structures like trend and seasonality. But it can capture a huge range of time series patterns.

ARIMA models

Autoregressive Moving Average models:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} \\ + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t.$$

ARIMA models

Autoregressive Moving Average models:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t.$$

- Predictors include both **lagged values of y_t and lagged errors.**

ARIMA models

Autoregressive Moving Average models:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t.$$

- Predictors include both **lagged values of y_t and lagged errors.**

Autoregressive Integrated Moving Average models

- Combine ARMA model with **differencing**.
- d-differenced series follows an ARMA model.
- Need to choose p, d, q and whether or not to include c.

Seasonal ARIMA models

ARIMA	$\underbrace{(p, d, q)}$	$\underbrace{(P, D, Q)_m}$
	$\uparrow$	$\uparrow$
	Non-seasonal part of the model	Seasonal part of of the model

- m = number of observations per year.
- d first differences, D seasonal differences
- p AR lags, q MA lags
- P seasonal AR lags, Q seasonal MA lags

Seasonal and non-seasonal terms combine multiplicatively

How does ARIMA() work?

Hyndman and Khandakar (JSS, 2008) algorithm:

- Select no. differences d via KPSS test.
- Select p, q and inclusion of c by minimising AICc.
- Use stepwise search to traverse model space.

How does ARIMA() work?

Hyndman and Khandakar (JSS, 2008) algorithm:

- Select no. differences d via KPSS test.
- Select p, q and inclusion of c by minimising AICc.
- Use stepwise search to traverse model space.

$$\text{AICc} = -2\log(L) + 2(p+q+k+1)\left[1 + \frac{(p+q+k+2)}{T-p-q-k-2}\right].$$

where L is the maximised likelihood fitted to the *differenced* data, $k = 1$ if $c \neq 0$ and $k = 0$ otherwise.

How does ARIMA() work?

Hyndman and Khandakar (JSS, 2008) algorithm:

- Select no. differences d via KPSS test.
- Select p, q and inclusion of c by minimising AICc.
- Use stepwise search to traverse model space.

$$\text{AICc} = -2\log(L) + 2(p+q+k+1)\left[1 + \frac{(p+q+k+2)}{T-p-q-k-2}\right].$$

where L is the maximised likelihood fitted to the *differenced* data, $k = 1$ if $c \neq 0$ and $k = 0$ otherwise.

Note: Can't compare AICc for different values of d.

How does ARIMA() work?

Step1: Select current model (with smallest AICc) from:

ARIMA$(2, d, 2)$
ARIMA$(0, d, 0)$
ARIMA$(1, d, 0)$
ARIMA$(0, d, 1)$

How does ARIMA() work?

Step1: Select current model (with smallest AICc) from:

ARIMA$(2, d, 2)$

ARIMA$(0, d, 0)$

ARIMA$(1, d, 0)$

ARIMA$(0, d, 1)$

Step 2: Consider variations of current model:

- vary one of $p, q,$ from current model by ± 1;
- p, q both vary from current model by ± 1;
- Include/exclude c from current model.

Model with lowest AICc becomes current model.

How does ARIMA() work?

Step1: Select current model (with smallest AICc) from:

ARIMA$(2, d, 2)$

ARIMA$(0, d, 0)$

ARIMA$(1, d, 0)$

ARIMA$(0, d, 1)$

Step 2: Consider variations of current model:

- vary one of p, q, from current model by ± 1;
- p, q both vary from current model by ± 1;
- Include/exclude c from current model.

Model with lowest AICc becomes current model.

Repeat Step 2 until no lower AICc can be found.

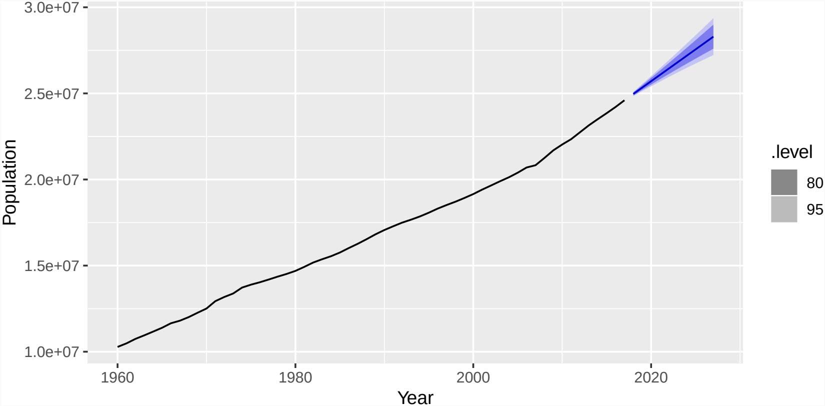
Example: National populations

```r
fit <- global_economy %>%
  model(arima = ARIMA(Population))
fit
```

```
## # A mable: 263 x 2
## # Key:     Country [263]
##    Country             arima
##    <fct>               <model>
##  1 Afghanistan         <ARIMA(4,2,1)>
##  2 Albania             <ARIMA(0,2,2)>
##  3 Algeria             <ARIMA(2,2,2)>
##  4 American Samoa      <ARIMA(2,2,2)>
##  5 Andorra             <ARIMA(2,1,2) w/ drift>
##  6 Angola              <ARIMA(4,2,1)>
##  7 Antigua and Barbuda <ARIMA(2,1,2) w/ drift>
##  8 Arab World          <ARIMA(0,2,1)>
```

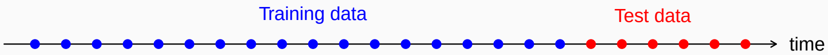
Example: National populations

```
fit %>% forecast(h=10) %>%
  filter(Country=="Australia") %>%
  autoplot(global_economy)
```



Outline

Training and test sets



- A model which fits the training data well will not necessarily forecast well.
- Forecast accuracy is based only on the test set.

Forecast errors

Forecast “error”: the difference between an observed value and its forecast.

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T},$$

where the training data is given by $\{y_1, \dots, y_T\}$

Measures of forecast accuracy

Measures of forecast accuracy

$$
\begin{aligned}
y_{T+h} &= \text{$(T+h)$th observation, } h = 1, \dots, H \\
\hat{y}_{T+h|T} &= \text{its forecast based on data up to time } T. \\
e_{T+h} &= y_{T+h} - \hat{y}_{T+h|T}
\end{aligned}
$$

$$
\begin{aligned}
\text{MAE} &= \text{mean}(|e_{T+h}|) \\
\text{MSE} &= \text{mean}(e_{T+h}^2) \qquad \qquad \text{RMSE} = \sqrt{\text{mean}(e_{T+h}^2)} \\
\text{MAPE} &= 100\text{mean}(|e_{T+h}|/|y_{T+h}|)
\end{aligned}
$$

Measures of forecast accuracy

$$
\begin{aligned}
y_{T+h} &= (T+h)\text{th observation}, h = 1, \dots, H \\
\hat{y}_{T+h|T} &= \text{its forecast based on data up to time } T. \\
e_{T+h} &= y_{T+h} - \hat{y}_{T+h|T}
\end{aligned}
$$

$$
\begin{aligned}
\text{MAE} &= \text{mean}(|e_{T+h}|) \\
\text{MSE} &= \text{mean}(e_{T+h}^2) \qquad \text{RMSE} = \sqrt{\text{mean}(e_{T+h}^2)} \\
\text{MAPE} &= 100\text{mean}(|e_{T+h}|/|y_{T+h}|)
\end{aligned}
$$

- MAE, MSE, RMSE are all scale dependent.
- MAPE is scale independent but is only sensible if $y_t \gg 0$ for all t, and y has a natural zero.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = \text{mean}(|e_{T+h}|/Q)$$

where Q is a stable measure of the scale of the time series $\{y_t\}$.

Proposed by Hyndman and Koehler (IJF, 2006).

For non-seasonal time series,

$$Q = (T-1)^{-1}\sum_{t=2}^{T}|y_t - y_{t-1}|$$

works well. Then MASE is equivalent to MAE relative to a naïve method.

Measures of forecast accuracy

Mean Absolute Scaled Error

$$\text{MASE} = \text{mean}(|e_{T+h}|/Q)$$

where Q is a stable measure of the scale of the time series $\{y_t\}$.

Proposed by Hyndman and Koehler (IJF, 2006).

For seasonal time series,

$$Q = (T-m)^{-1} \sum_{t=m+1}^{T} |y_t - y_{t-m}|$$

works well. Then MASE is equivalent to MAE relative to a seasonal naïve method.

Measures of forecast accuracy

```r
recent_production <- aus_production %>%
  filter(year(Quarter) >= 1992)
train <- recent_production %>% filter(year(Quarter) <= 2007)
snowy_fit <- train %>%
  model(
    ets = ETS(Trips),
    arima = ARIMA(Trips)
  )
snowy_fc <- forecast(snowy_fit, h="4 years")
accuracy(snowy_fc, aus_production)
```