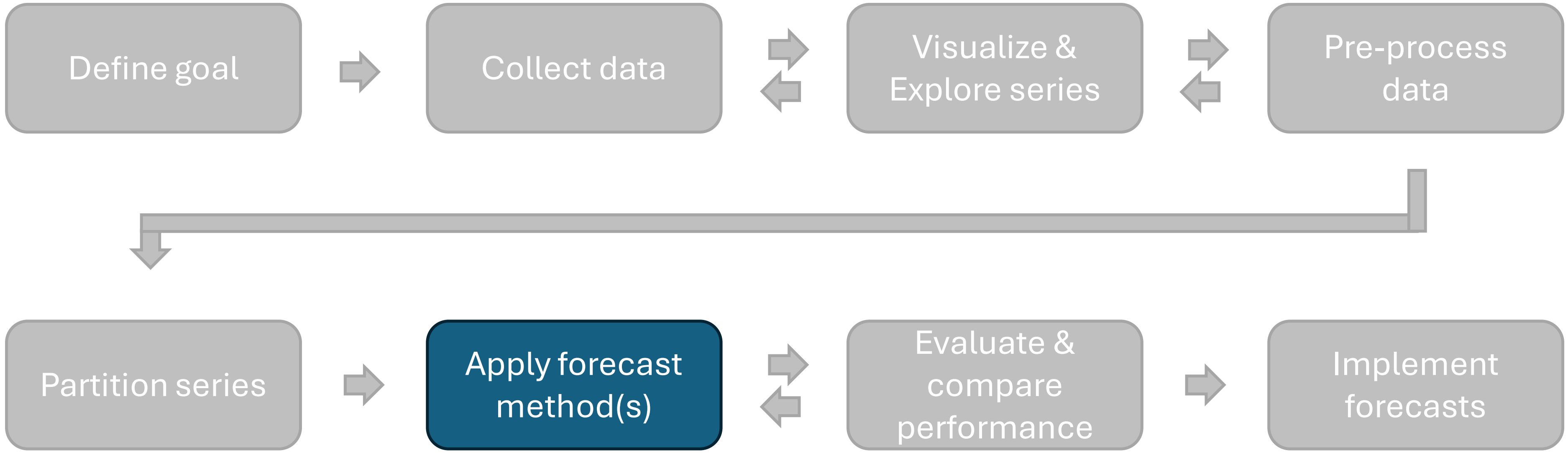
Time series Forecasting

Part 2: Smoothing Methods

The forecasting process



Components of a Time Series - Recap

1. **Level** (always present)
2. **Trend**: steady increase/decrease over time.
3. **Seasonality**: pattern that repeats itself every season
4. Random **noise** (always present)

Reminder: Modeling Principles

Time series Analysis

- Reasonableness and parsimony
- Goodness of fit (residual analysis)

Time series Forecasting

- Forecast accuracy
- Parsimony and reasonableness

Why so many different methods?

Model-based methods	Data-driven methods
Linear regression	Naïve forecast
Autoregressive models	Smoothing methods
ARIMA	Neural nets
Logistic regression	
Econometrics models	

Not accrue over time ⟷	Accrue over time	**Structural changes**
Unlikely to be violated ⟷	Likely to be violated	**Assumptions**
Short time series ⟷	Long time series	**Amount of data**
Substantial ⟷	Minimal	**User inputs required**
Global patterns ⟷	Local patterns	**Global/local patterns**

Smoothing methods

Moving average

Exponential smoothing

- Smoothing methods are useful for
  - Data visualization
  - Removing seasonality and computing seasonal indexes
  - Forecasting

The Moving Average Method

The Idea:
Forecast future points by using an average of several past points

Advantages:
Simple, popular

Disadvantages:
Forecast only in series that lack seasonality and trend

Uses:
Time series visualisation
Computing seasonal indexes
Forecasting

Key concept:
Width of window

Notation

$t = 1,2,3, \ldots$	An index denoting the time period of interest. $t = 1$ is the first period in a series.
$y_1, y_2, y_3, \cdots, y_n$	A series of n values measured over n time periods, where y_t denotes the series value at time t.
F_t	The forecasted value for time period t.
F_{t+k}	The k-step-ahead forecast.
e_t	The forecast error for time period t, which is the difference between the actual value and the forecast at time t, and equal to $y_t - F_t$.

Two types of windows

Centered moving average:

based on a window <u>centered</u> around time *t*

Centered window (w=5)

y_{t-2} y_{t-1} y_t y_{t+1} y_{t+2}

Past Future

$$F_t = \frac{y_{t-2}+y_{t-1}+y_t+y_{t+1}+y_{t+2}}{5}$$

$$F_{t+3} = \frac{y_{t+1}+y_{t+2}+y_{t+3}+y_{t+4}+y_{t+5}}{5}$$

Trailing moving average:

based on a window from time *t* and backwards

Trailing window (w=5)

y_{t-4} y_{t-3} y_{t-2} y_{t-1} y_t

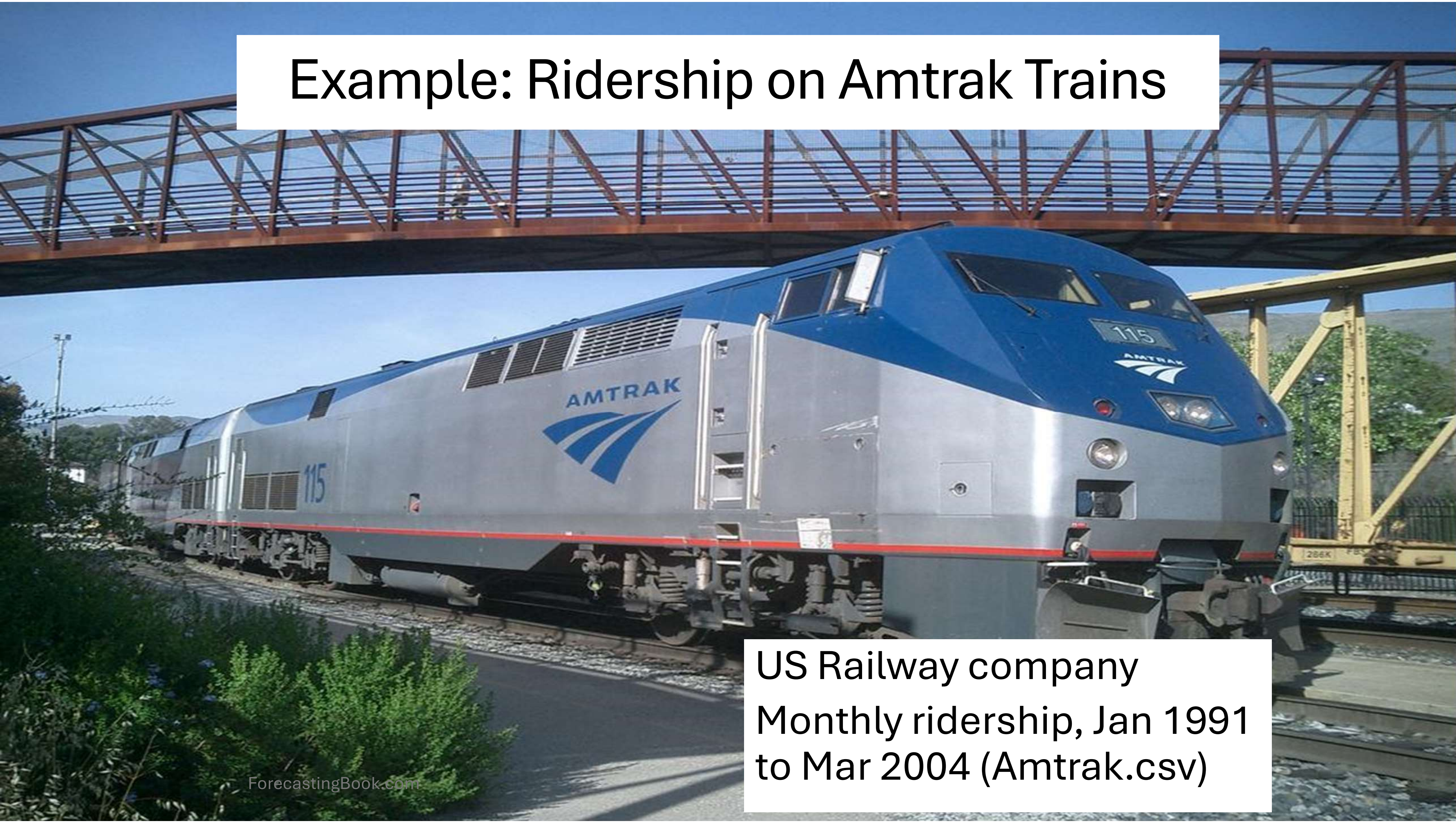
$$F_t = \frac{y_{t-1}+y_{t-2}+y_{t-3}+y_{t-4}+y_{t-5}}{5}$$

$$F_{t+1} = \frac{y_t+y_{t-1}+y_{t-2}+y_{t-3}+y_{t-4}}{5}$$

Moving averages for visualizing time series

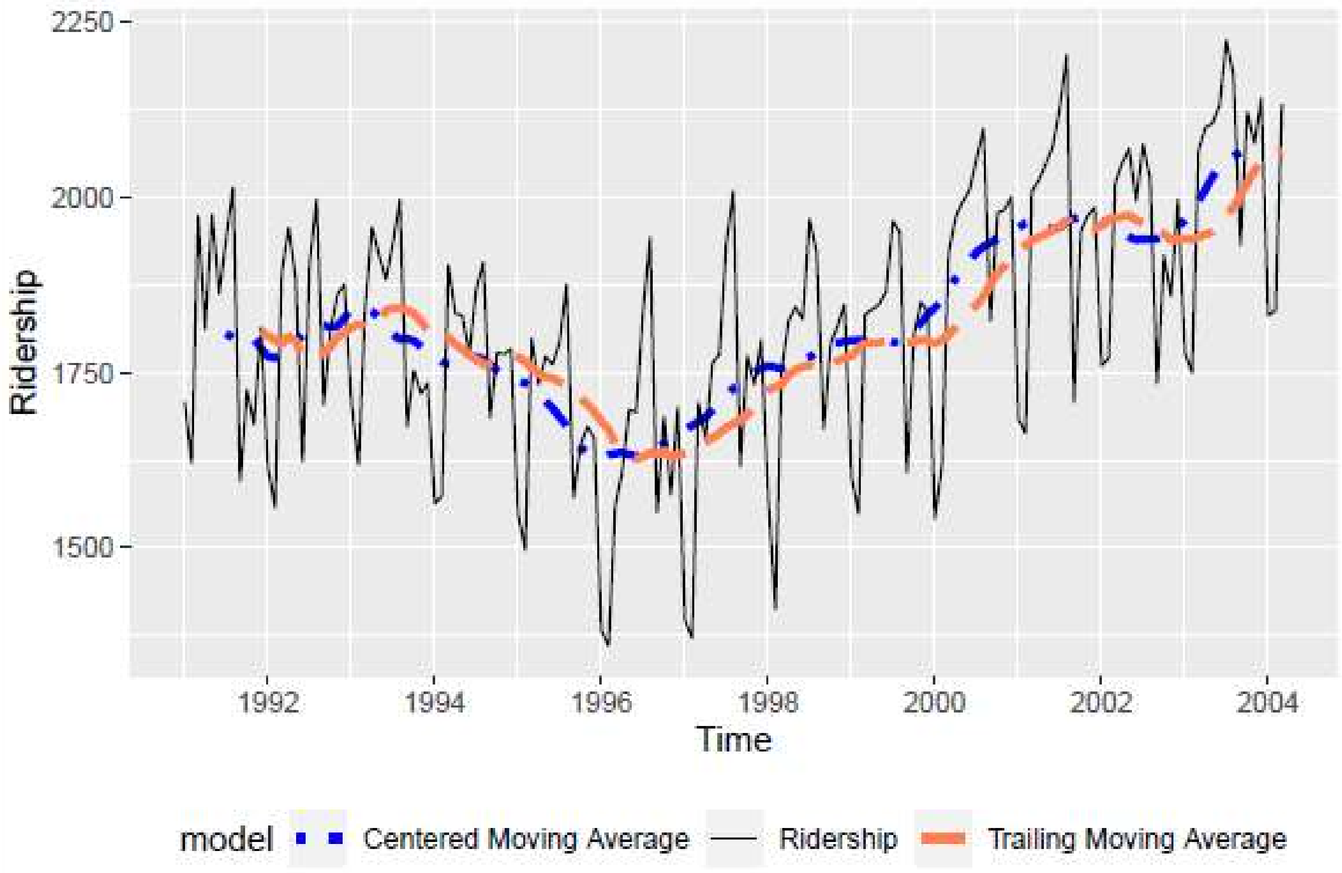
A time-plot of the **moving averages** can help reveal the LEVEL and TREND of a series, by filtering out the seasonal and random components

Example: Ridership on Amtrak Trains



US Railway company

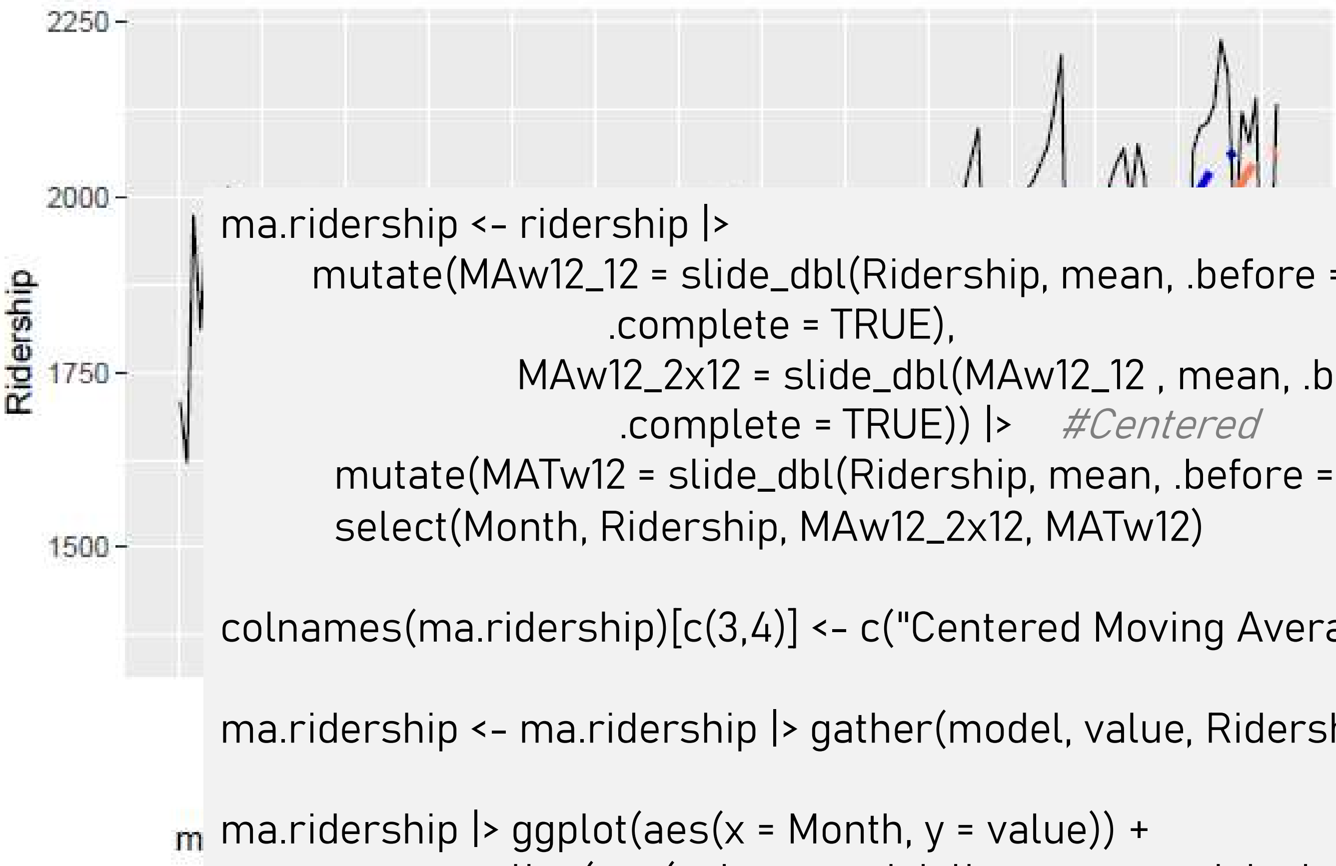
Monthly ridership, Jan 1991 to Mar 2004 (Amtrak.csv)



Moving average

Centered moving average and trailing moving average with window width w=12, overlaid on Amtrak ridership series

Moving average



Centered moving average and trailing moving average with window width w=12, overlaid on Amtrak ridership series

```
ma.ridership <- ridership |>
        mutate(MAw12_12 = slide_dbl(Ridership, mean, .before = 5, .after = 6,
                                .complete = TRUE),
                          MAw12_2x12 = slide_dbl(MAw12_12 , mean, .before = 1, .after = 0,
                                .complete = TRUE)) |>     #Centered
          mutate(MATw12 = slide_dbl(Ridership, mean, .before = 11, .after = 0, .complete = TRUE)) |> #Trailing
          select(Month, Ridership, MAw12_2x12, MATw12)

colnames(ma.ridership)[c(3,4)] <- c("Centered Moving Average", "Trailing Moving Average")

ma.ridership <- ma.ridership |> gather(model, value, Ridership:`Trailing Moving Average`)

ma.ridership |> ggplot(aes(x = Month, y = value)) +
              geom_line(aes(color = model, linetype = model, size = model) ) +
              scale_size_manual(values = c(1.2 , 0.4, 1.2)) +   s
              cale_linetype_manual(values = c("dotdash", "solid", "longdash")) +
              theme(legend.position = "bottom") +
              scale_x_yearmonth(date_breaks = "2 years", date_labels = "%Y") +
              labs(x = "Time", y = "Ridership")
```

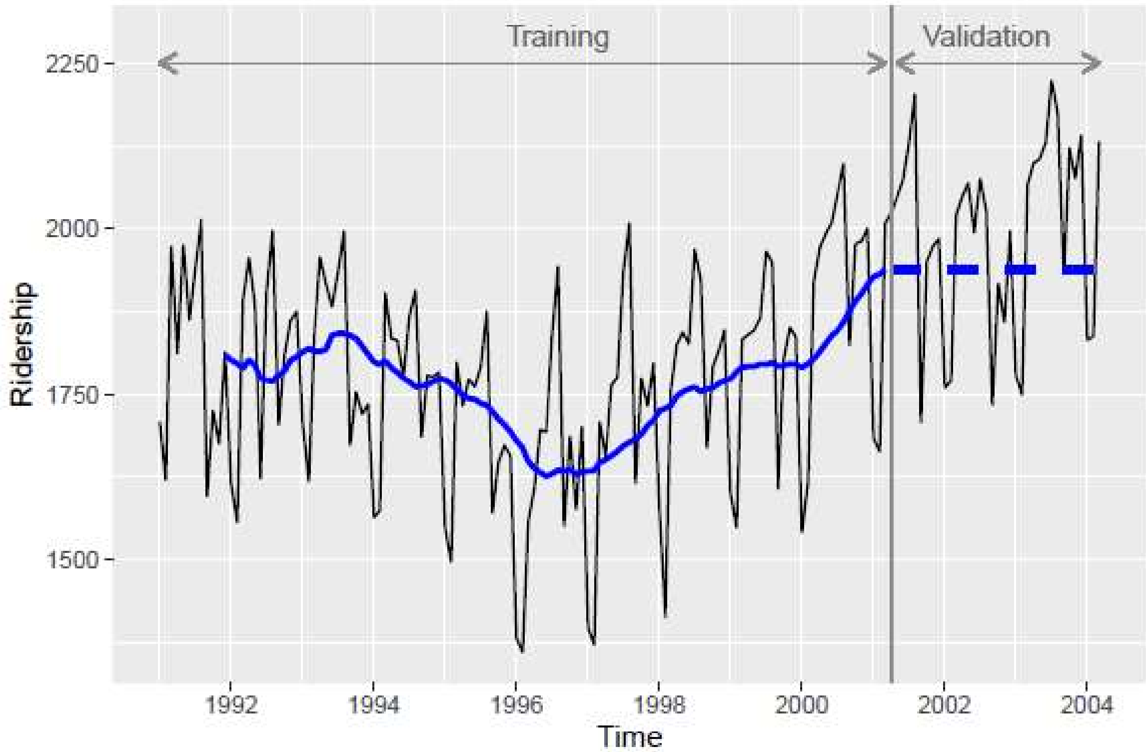
Centered moving average with an even window

```
ma.ridership <- ridership |>
      mutate(MAw12_12 = slide_dbl(Ridership, mean, .before = 5, .after = 6,
                          .complete = TRUE),
                   MAw12_2x12 = slide_dbl(MAw12_12 , mean, .before = 1, .after = 0,
                          .complete = TRUE))     #Centered
```

$$MAw12\_2x12 = \frac{1}{2}\left[\frac{1}{12}(y_{t-5} + y_{t-4} + y_{t-3} + y_{t-2} + y_{t-1} + y_t + y_{t+1} + y_{t+2} + y_{t+3} + y_{t+4} + y_{t+5} + y_{t+6}) + \frac{1}{12}(y_{t-6} + y_{t-5} + y_{t-4} + y_{t-3} + y_{t-2} + y_{t-1} + y_t + y_{t+1} + y_{t+2} + y_{t+3} + y_{t+4} + y_{t+5})\right]$$

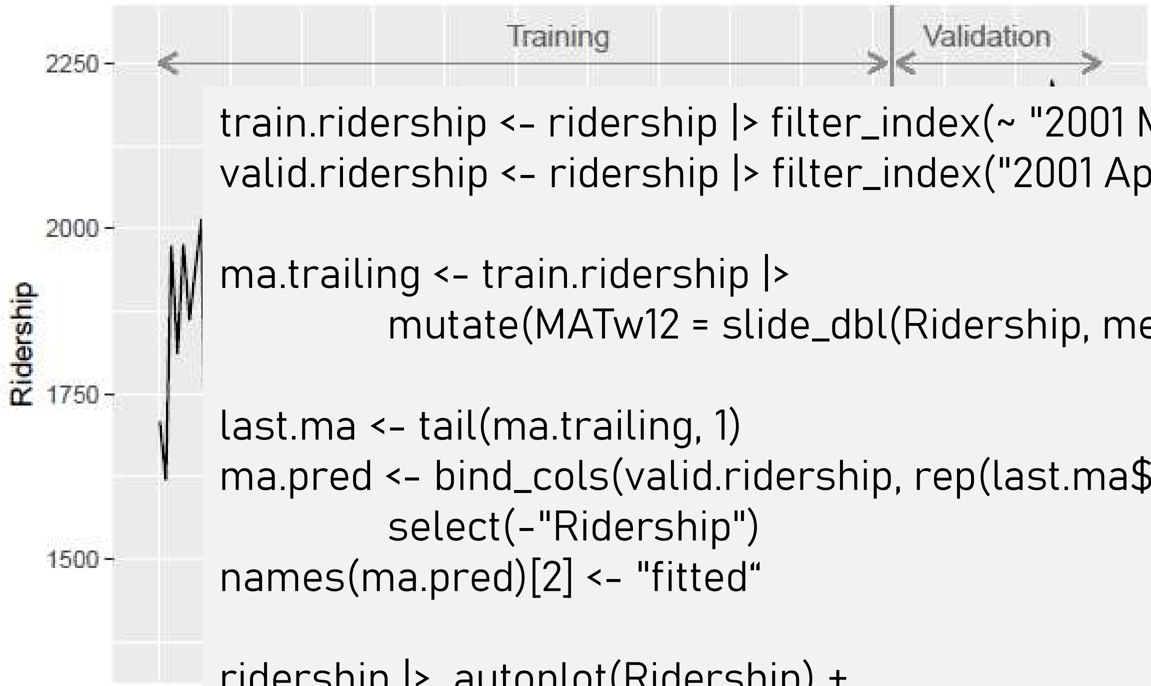
$$= \frac{1}{24}y_{t-6} + \frac{1}{12}y_{t-5} + + \frac{1}{12}y_{t-4} + \frac{1}{12}y_{t-3} + \frac{1}{12}y_{t-2} + \frac{1}{12}y_{t-1} + \frac{1}{12}y_t + \frac{1}{12}y_{t+1} + \frac{1}{12}y_{t+2} + \frac{1}{12}y_{t+3} + \frac{1}{12}y_{t+4} + \frac{1}{12}y_{t+5} + \frac{1}{24}y_{t+6}$$

The Moving Average Method



Trailing moving average forecaster with w=12 applied to Amtrak ridership series.

The Moving Average Method



Trailing moving average forecaster with

```
train.ridership <- ridership |> filter_index(~ "2001 Mar")
valid.ridership <- ridership |> filter_index("2001 Apr" ~ .)

ma.trailing <- train.ridership |>
        mutate(MATw12 = slide_dbl(Ridership, mean, .before = 11, .complete = TRUE))

last.ma <- tail(ma.trailing, 1)
ma.pred <- bind_cols(valid.ridership, rep(last.ma$MATw12, nrow(valid.ridership))) |>
        select(-"Ridership")
names(ma.pred)[2] <- "fitted“

ridership |>  autoplot(Ridership) +
        geom_line(aes(y = MATw12), data = ma.trailing, size = 1.0, colour = "blue1") +
        autolayer(ma.pred, alpha = 0.5, level = NULL, linetype = "dashed", size = 1.25, colour = "blue1") +
         xlab("Time") + ylab("Ridership")  +
        theme(legend.position = "none")
```

Choosing window width (w)

- Balance over- and under-smoothing
- Wide window – global trend, narrow window – reveal local trend
- If no seasonality, use a narrow window (under-smoothing)

Test yourself

For a seasonal series, what window width should you use?

1. Smaller than the # seasons
2. Larger than the # seasons
3. Equal to the # seasons

Hands-On # 2.1

- Fortified wine has the largest market share of the six types of wine. You are asked to focus on fortified wine sales alone
- Fit moving average model with a window of size 5 and size 12
- What patterns do these models suggest?
- (Hint: don’t forget to split the data into training and validation sets)

Removing trend and/or seasonality

<u>Differencing:</u>

Taking the difference between two consecutive observations.

Usage

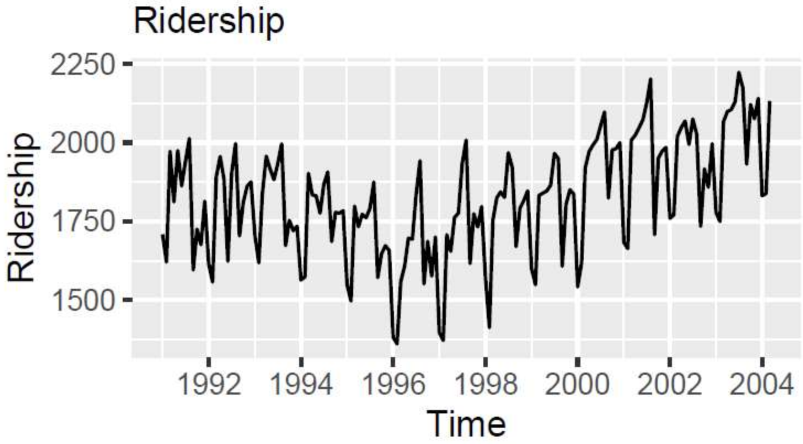
Removing a trend and/or seasonality from a time series

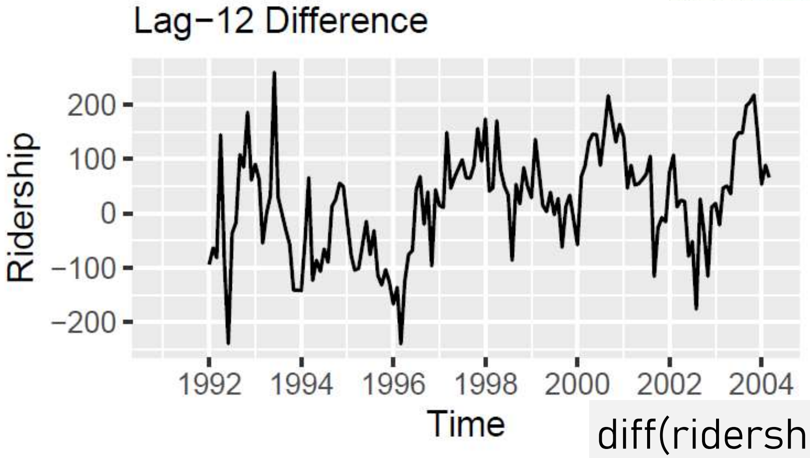
$Lag - 1$ difference: $y_t - y_{t-1}$	Removing trend
$Lag - m$ difference: $y_t - y_{t-m}$	Removing seasonality with m seasons

E.g., monthly seasonality m=12, daily seasonality m=7

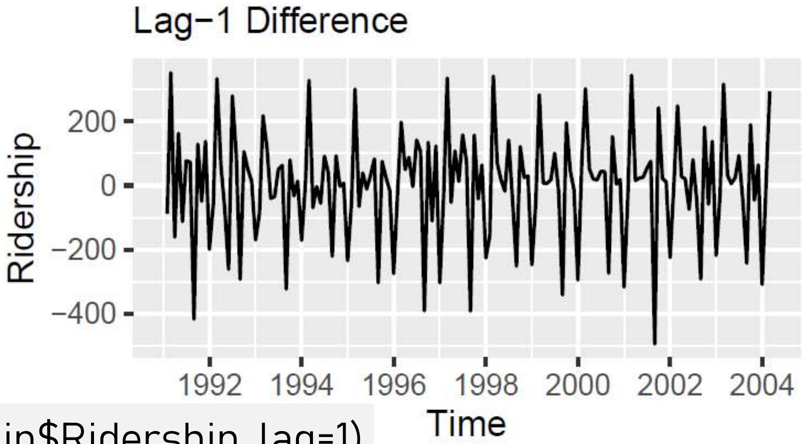
Double-differencing: difference the differenced series

Example: Ridership on Amtrak Trains

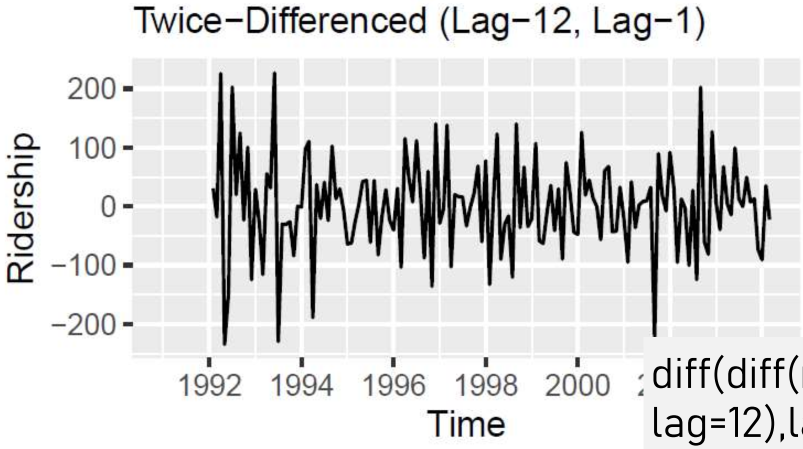




diff(ridership$Ridership, lag=12)



diff(ridership$Ridership, lag=1)



diff(diff(ridership$Ridership, lag=12),lag=1)

Methods require no trend and/or seasonality

- Moving average
- (Simple) Exponential smoothing

The simple exponential smoothing Method

The Idea:

Forecast future points by using an exponential weighted average of several past points

Uses:

Forecasting
Automated forecasting
Capture well for local patterns
Visualization
Creating seasonal indexes

Advantages:

Simple to understand
Gives more influence on recent information
Storage/computation efficient (we only need to store the last forecast and most recent observation)

Disadvantages:

Forecast only in series that lack seasonality and trend*
Forecast into the future & one-step-ahead forecast the same

Key concept:

Smoothing constant α

Exponential smoothing

Types of exponential smoothing

- Simple exponential smoothing (no trend or seasonality)
- Advance exponential smoothing (trend and/or seasonality)
  - Holt's method (with trend but no seasonality)
  - Winter's method (with trend and seasonality)

Simple exponential smoothing

- Assume that the series has only level (L_t) and noise (unpredictable)

$F_{t+1} = L_t$

Forecasts are estimated as level at the most recent time

$L_t = \alpha y_t + (1 - \alpha)L_{t-1}$

Exp smoothing is an adaptive algorithm.

It adjust the most resent forecast (level) based on actual data

$0 < \alpha \leq 1$

α = the smoothing constant

Initialization: $F_t = L_1 = y_1$

Why 'exponential smoothing'?

- Let's examine $F_{t+1} = L_t$ where $L_t = \alpha y_t + (1-\alpha)L_{t-1}$

$$
\begin{aligned}
F_{t+1} &= L_t \\
&= \alpha y_t + (1-\alpha)[\alpha y_{t-1} + (1-\alpha)L_{t-2}] = \\
&= \alpha y_t + \alpha(1-\alpha)y_{t-1} + (1-\alpha)^2 L_{t-2} = \\
&= \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \cdots
\end{aligned}
$$

Required data storage and computational time!

Weights decrease exponentially into the past!

$$= F_t + \alpha e_t$$

Active learner!

See page 94 1 para

Update **previous forecast**

α controls the **degree of 'leaning'**

By an amount that depends on the **error** in the previous forecast

The smoothing constant α

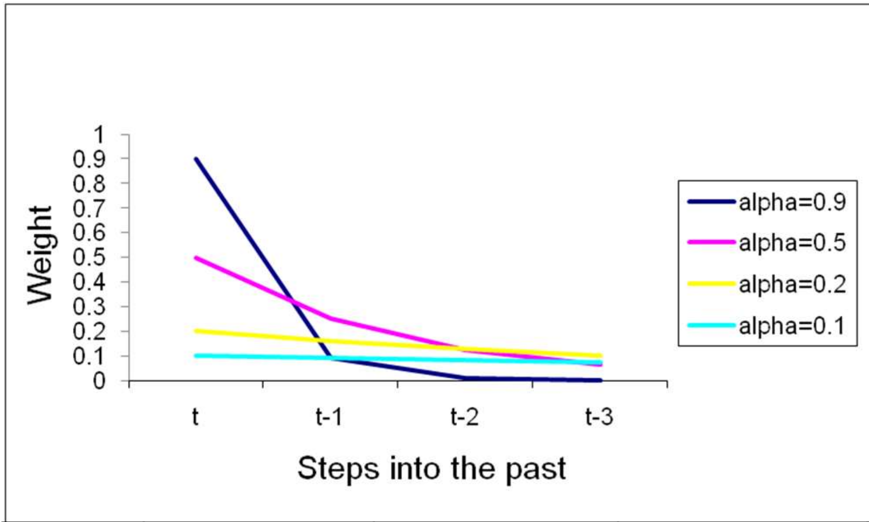
- Controls the degree of 'leaning'

$$0 \leftarrow \quad \alpha \quad \rightarrow 1$$

	Slow learner	Fast learner
Past obs.	Have a large influence on the forecast	Have little to no influence forecasts
	Over-smoothing	Under-smoothing

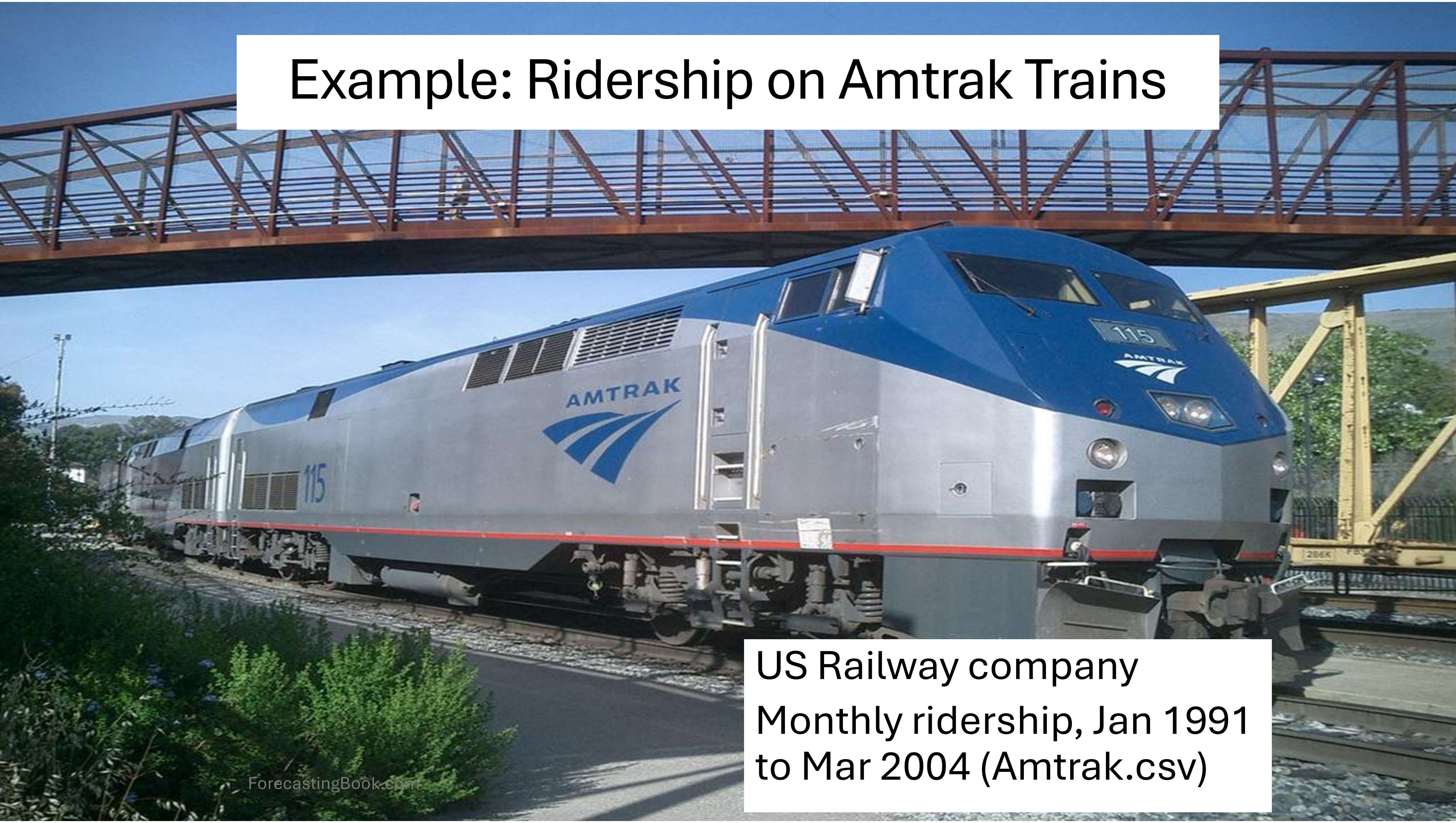
- Selecting α

Common values: 0.1 or 0.2
Trail & error: effect on visualisation
Minimise RMSE or MAPE of training data



α	α (1- α)	α(1- α)²	α(1- α)³
0.9	0.09	0.009	0.0009
0.5	0.25	0.125	0.0625
0.2	0.16	0.128	0.1024
0.1	0.09	0.081	0.0729

Example: Ridership on Amtrak Trains

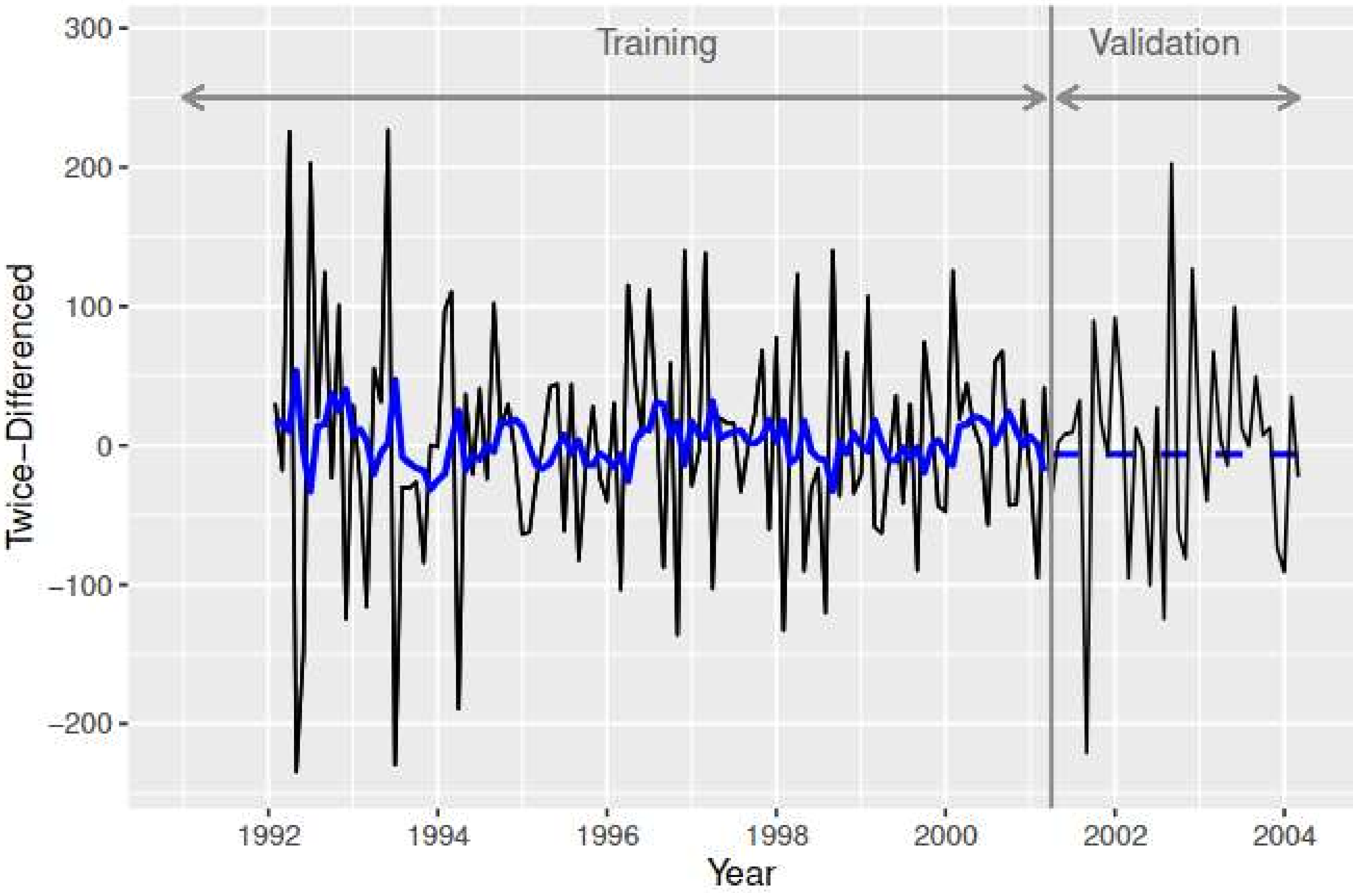


US Railway company

Monthly ridership, Jan 1991 to Mar 2004 (Amtrak.csv)

Exp smoothing

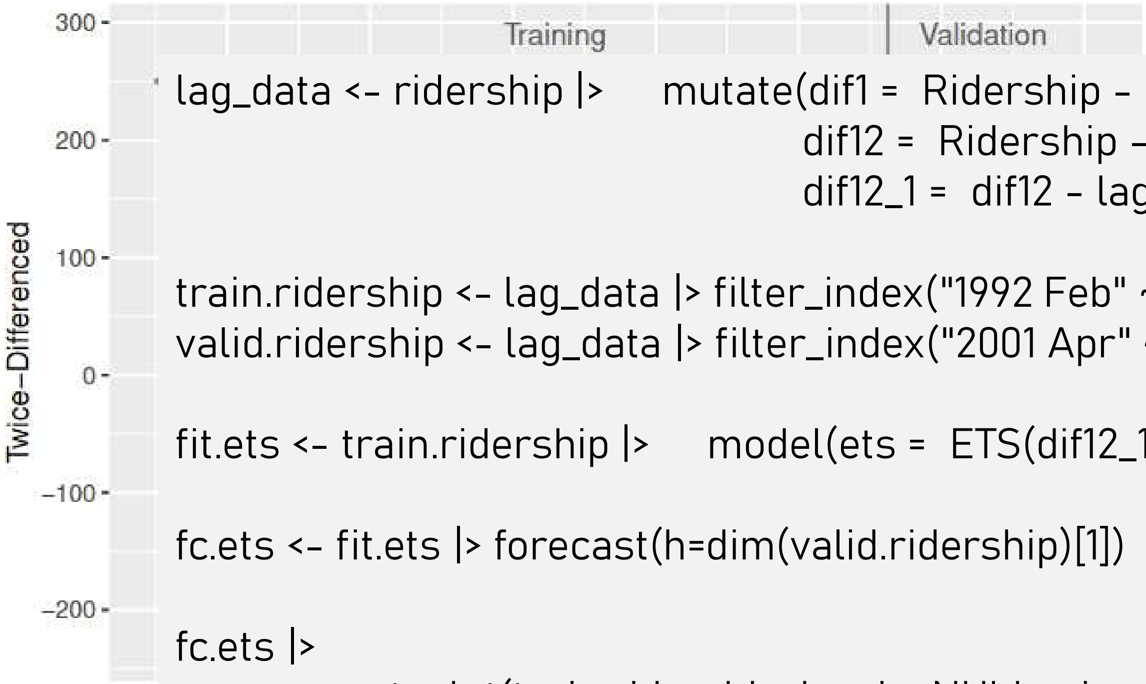
Exponential smoothing



Exp Smoothing (α=0.2) applied to twice-differenced Amtrak series

Exponential smoothing

Hands on activity – generate auto ETS




```
lag_data <- ridership |>     mutate(dif1 =  Ridership - lag(Ridership, 1),
                                    dif12 =  Ridership - lag(Ridership, 12),
                                    dif12_1 =  dif12 - lag(dif12, 1))

train.ridership <- lag_data |> filter_index("1992 Feb" ~ "2001 Mar")
valid.ridership <- lag_data |> filter_index("2001 Apr" ~ .)

fit.ets <- train.ridership |>     model(ets =  ETS(dif12_1 ~ error("A") + trend("N", alpha = 0.2) + season("N")))

fc.ets <- fit.ets |> forecast(h=dim(valid.ridership)[1])

fc.ets |>
        autoplot(train.ridership, level = NULL, size = 1, linetype = "dashed") +
                geom_line(aes(y = dif12_1), data = lag_data, size = 0.5) +
                geom_line(aes(y = .fitted), data = fitted.values(fit.ets), size = 1, colour = "blue1") +
                xlab("Year") + ylab("Twice-Differenced")
```

Some useful commands

- *report*(fit.ets)

```
> report(fit.ets)
Series: dif12_1
Model: ETS(A,N,N)
  Smoothing parameters:
    alpha = 0.2

  Initial states:
     l[0]
 14.35123

  sigma^2:  8401.487

     AIC     AICc      BIC
1513.012 1513.125 1518.413
```

Some useful commands

- Forecasted values for the training data
- pred.values.ets <- fitted.values(fit.ets)
- View(pred.values.ets)

Filter

	.model	Month	.fitted
1	ets	1992 Feb	14.35123000
2	ets	1992 Mar	17.59938400
3	ets	1992 Apr	10.48070720
4	ets	1992 May	53.54616576
5	ets	1992 Jun	-4.07626739
6	ets	1992 Jul	-33.07381391
7	ets	1992 Aug	14.09354887
8	ets	1992 Sep	15.27463910
9	ets	1992 Oct	37.17811128
10	ets	1992 Nov	25.10968902
11	ets	1992 Dec	40.25935122
12	ets	1993 Jan	7.27248097
13	ets	1993 Feb	11.65258478
14	ets	1993 Mar	3.52506782
15	ets	1993 Apr	-20.37214574
16	ets	1993 May	-5.18251659
17	ets	1993 Jun	2.13578673
18	ets	1993 Jul	47.08562938
19	ets	1993 Aug	-8.26269650

Some useful commands

- Forecasted values for the validation data
- fc.ets

```
> fc.ets
# A fable: 36 x 4 [1M]
# Key:     .model [1]
   .model    Month         dif12_1 .mean
   <chr>     <mth>          <dist> <dbl>
 1 ets    2001 Apr  N(-6.3, 8401) -6.35
 2 ets    2001 May  N(-6.3, 8738) -6.35
 3 ets    2001 Jun  N(-6.3, 9074) -6.35
 4 ets    2001 Jul  N(-6.3, 9410) -6.35
 5 ets    2001 Aug  N(-6.3, 9746) -6.35
 6 ets    2001 Sep N(-6.3, 10082) -6.35
 7 ets    2001 Oct N(-6.3, 10418) -6.35
 8 ets    2001 Nov N(-6.3, 10754) -6.35
 9 ets    2001 Dec N(-6.3, 11090) -6.35
10 ets    2002 Jan N(-6.3, 11426) -6.35
# ℹ 26 more rows
# ℹ Use `print(n = ...)` to see more rows
>
```

Exponential smoothing – link to moving average

Moving average with $w = \frac{2}{\alpha} - 1$ will provide (approximately) equal result to the result of exponential smoothing with smoothing constant α.

Advanced exponential smoothing

Double exponential smoothing, Holt's method. **Additive trend**

$$F_{t+k} = L_t + kT_t$$

Assume a trend that can change over time (local trend). The level changes from one period to next by a fix amount

$$L_t = \alpha y_t + (1 - \alpha)(L_{t-1}+T_{t-1})$$

Level estimate at time t, a weighted average of observation at time t and the level in the previous period adjusted for trend

$$T_t = \beta(L_{t-} - L_{t-1}) + (1 - \beta)T_{t-1}$$

Trend estimate at time t, a weighted average of the most recent info on the change in level and the trend in the previous period

$$0 < \alpha \leq 1$$

$$0 < \beta \leq 1$$

Smoothing constants determine the rate of learning. As closer both to 1 as faster the learning (more weight to more recent obs.)

Advanced exponential smoothing

Two types of errors (additive trend)

Additive error

$$y_t = L_t + T_t + e_t$$

The error has a similar magnitude irrespective of the current level and/or trend of the series

Multiplicative error

$$y_t = (L_t + T_t)(1 + e_t)$$

The error proportionally increases (decrease) with level and/or trend changes.

Advanced exponential smoothing

Multiplicative trend

$$F_{t+1} = L_t \times T_t^k$$

Assume a trend that can change over time (local trend). The level changes from one period to the next by a ~~fix amount~~ factor

$$L_t = \alpha y_t + (1 - \alpha)(L_{t-1} \times T_{t-1})$$

Updating equation for level at time t

$$T_t = \beta(L_t / L_{t-1}) + (1 - \beta)T_{t-1}$$

Updating equation for trend at time t

$$0 < \alpha \leq 1$$

$$0 < \beta \leq 1$$

Smoothing constants determine the rate of learning. As closer both to 1 as faster the learning (more weight to more recent obs.)

Advanced exponential smoothing

Two types of errors (Multiplicative trend)

Additive error

$$y_t = L_t \times T_t + e_t$$

The error has a similar magnitude irrespective of the current level and/or trend of the series

Multiplicative error

$$y_t = (L_t \times T_t)(1 + e_t)$$

The error proportionally increases (decrease) with level and/or trend changes.

Exp smoothing

Advanced exponential smoothing

Holt-Winter's exponential smoothing. **Additive seasonality**

$$F_{t+k} = L_t + kT_t + S_{t+k-m}$$

Assume a trend that can change over time (local trend). The level changes from one period to the next by a fixed amount. The values of different seasons differ by a fix amount

$$L_t = \alpha(\underbrace{y_t - S_{t-m}}_{\text{Seasonal adjusted value}}) + (1-\alpha)(L_{t-1} + T_{t-1})$$

Updating equation for level at time t

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}$$

Updating equation for trend at time t

$$S_t = \gamma(\underbrace{y_t - L_t}_{\text{Level adjusted value}}) + (1-\gamma)S_{t-m}$$

Updating equation for seasonality at time t

$$0 < \alpha, \beta, \gamma \leq 1$$

Smoothing constants determine the rate of learning.

Advanced exponential smoothing

Holt-Winter's exponential smoothing. **Multiplicative seasonality**

$$F_{t+k} = (L_t + kT_t) \times S_{t+k-m}$$

Assume a trend that can change over time (local trend). The level changes from one period to the next by a fixed amount. The values of different seasons differ by percentage amount

$$L_t = \alpha\underbrace{(y_t / S_{t-m})}_{\text{Seasonal adjusted value}} + (1-\alpha)(L_{t-1} + T_{t-1})$$

Updating equation for level at time t

$$T_t = \beta(L_{t-} - L_{t-1}) + (1-\beta)T_{t-1}$$

Updating equation for trend at time t

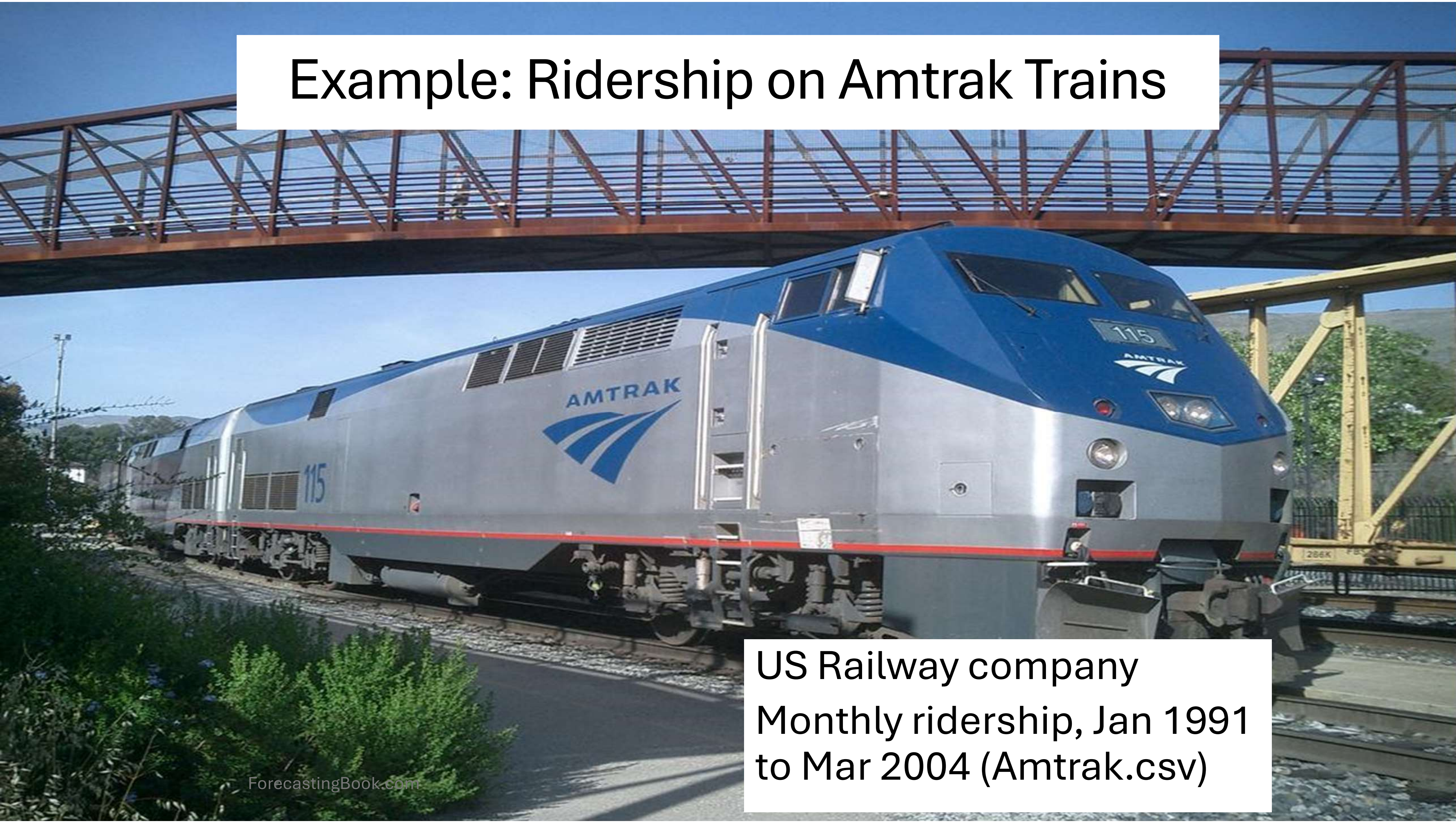
$$S_t = \gamma\underbrace{(y_t / L_t)}_{\text{Level adjusted value}} + (1-\gamma)S_{t-m}$$

Updating equation for seasonality at time t

$$0 < \alpha, \beta, \gamma \leq 1$$

Smoothing constants determine the rate of learning.

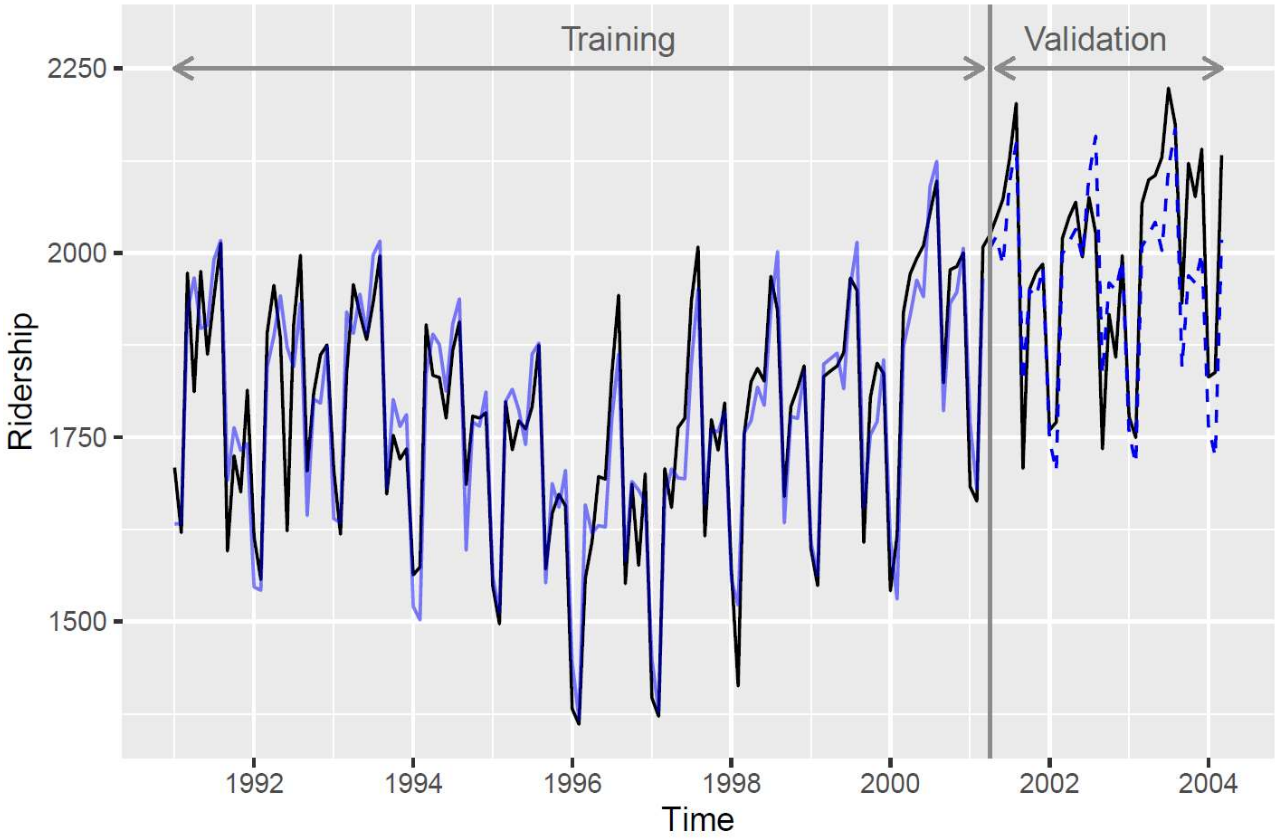
Example: Ridership on Amtrak Trains



US Railway company

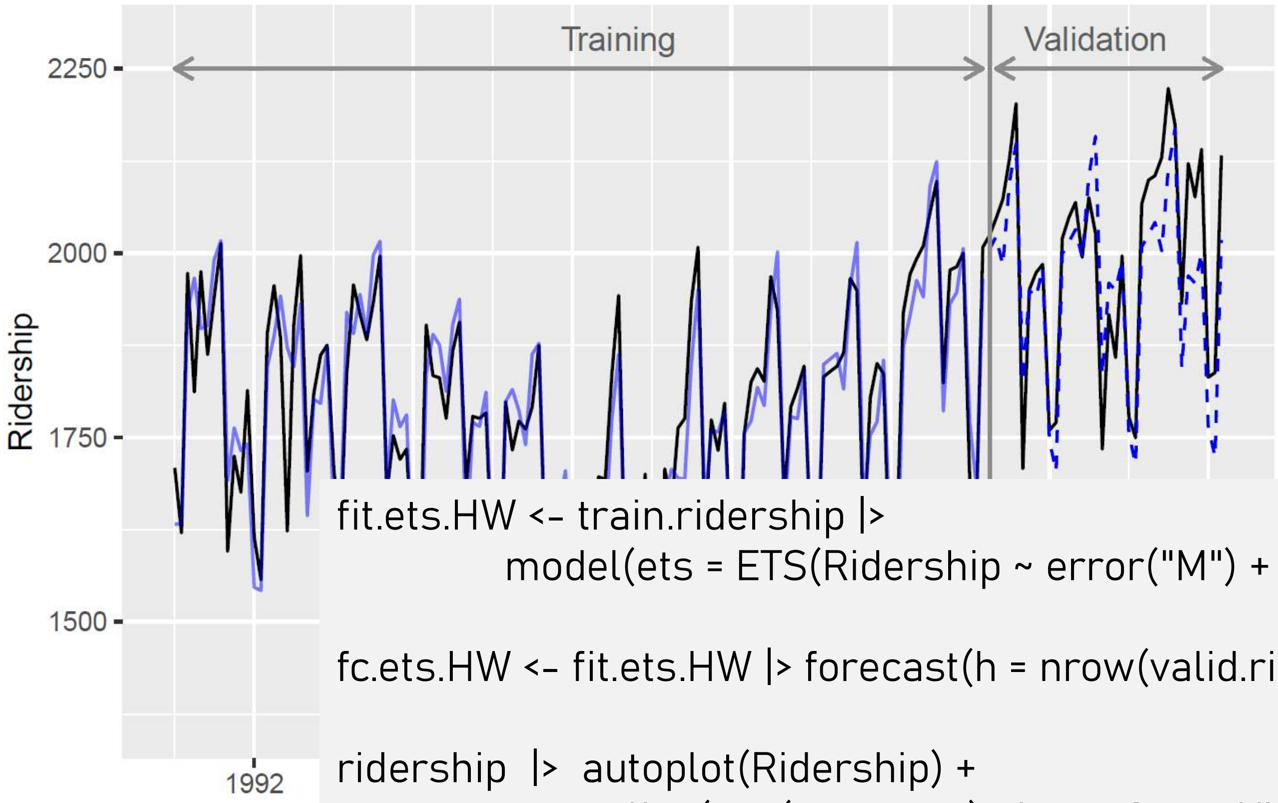
Monthly ridership, Jan 1991 to Mar 2004 (Amtrak.csv)

Advanced exponential smoothing



Forecasts from the Holt-Winter's exponential smoothing applied to the Amtrak ridership series.

Advanced exponential smoothing



Forecasts from the Holt-Winter's exponential smoothing applied to the Amtrak ridership series.

```
fit.ets.HW <- train.ridership |>
        model(ets = ETS(Ridership ~ error("M") + trend("A") + season("A")))

fc.ets.HW <- fit.ets.HW |> forecast(h = nrow(valid.ridership))

ridership  |>  autoplot(Ridership) +
        geom_line(aes(y = .mean), data = fc.ets.HW, colour = "blue1", linetype = "dashed") +
        autolayer(fitted.values(fit.ets.HW), alpha = 0.5, level = NULL, colour = "blue1") +
        xlab("Time") + ylab("Ridership")
```

Advanced exponential smoothing

```
> report(fit.ets.HW)
Series: Ridership
Model: ETS(M,A,A)
        Smoothing parameters:
          alpha = 0.5517889
          beta  = 0.0001119929
          gamma = 0.0001186167

        Initial states:
          l[0]      b[0]      s[0]      s[-1]     s[-2]     s[-3]     s[-4]     s[-5]     s[-6]
          1838.577  0.8081064 28.44536  -11.21868 -0.533687 -124.2766 200.1969  146.8103  36.59075
          s[-7]     s[-8]     s[-9]     s[-10]    s[-11]
          76.04396  60.15303  44.17906  -249.4682 -206.9221

         sigma^2:  0.0012

          AIC       AICc      BIC
          1617.596  1623.424  1665.403
```

error("M") + trend("A") + season("A")))

Estimate far from zero => level adopts locally

Estimate close zero => trend & seasonality global

Advanced exponential smoothing

```
# Final states:

> n <- nrow(components(fit.ets.HW))

> components(fit.ets.HW)[n, c("level", "slope")] # level and trend

#  A tibble: 1 x 2
             level     slope
             <dbl>     <dbl>1
              1945.     0.810

> t(components(fit.ets.HW)[(n-11):n, c("season")])  # s1 to s12
             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]         [,8]       [,9]
season   60.13771   76.05659   36.58921   146.8106   200.1936   -124.2748   -0.5324645   -11.21618   28.44364
              [,10]        [,11]       [,12]
season   -206.9249   -249.471   44.1899
```

Advanced exponential smoothing

The ETS() function

- Provide 18 model choices, combination of:

	Seasonality		
Trend	None	Additive	Multiplicative
None	ZNN	ZNA	ZNM
Additive	ZAN	ZAA	ZAM
Multiplicative	ZMN	ZMA	ZMM

where Z can be set to A or M corresponding to additive or multiplicative error, respectively.

```
train.ridership |>  model(ets = ETS(Ridership ~ error("M") + trend("A") + season("A")))
```

Automated exponential smoothing

- If you do not know which model to choose, you can use the ETS function to do automated model selection.
- It will fit all the models and select the one that minimised *Akaike's Information Criterion* (AIC),
- AIC combines fit to the training data with a penalty for the number of smoothing parameters.
- The AMM, AAM, AMA, and MMA models will not be considered as they can lead to numerical difficulties.

	Seasonality		
Trend	None	Additive	Multiplicative
None	ZNN	ZNA	ZNM
Additive	ZAN	ZAA	ZAM
Multiplicative	ZMN	ZMA	ZMM

A M A

Automated exponential smoothing

- To include the restricted models when searching for a model, include the options *restrict = FALSE* in the ETS function.
- You can automatically select a model from a subset of pre-specified models.
- For example,

  train.ridership |> model(ets= ETS(Ridership ~ error("M"))

  will consider all the models with multiplicative error except the MMA model (unless you include *restrict = FALSE*).

Automated exponential smoothing - example

```
fit.ets.auto <- train.ridership |> model(ets =  ETS(Ridership))

pred.values.ets.auto <- fitted.values(fit.ets.auto)

fc.ets.auto <- fit.ets.auto |> forecast(h=dim(valid.ridership)[1])
```

error("M"),

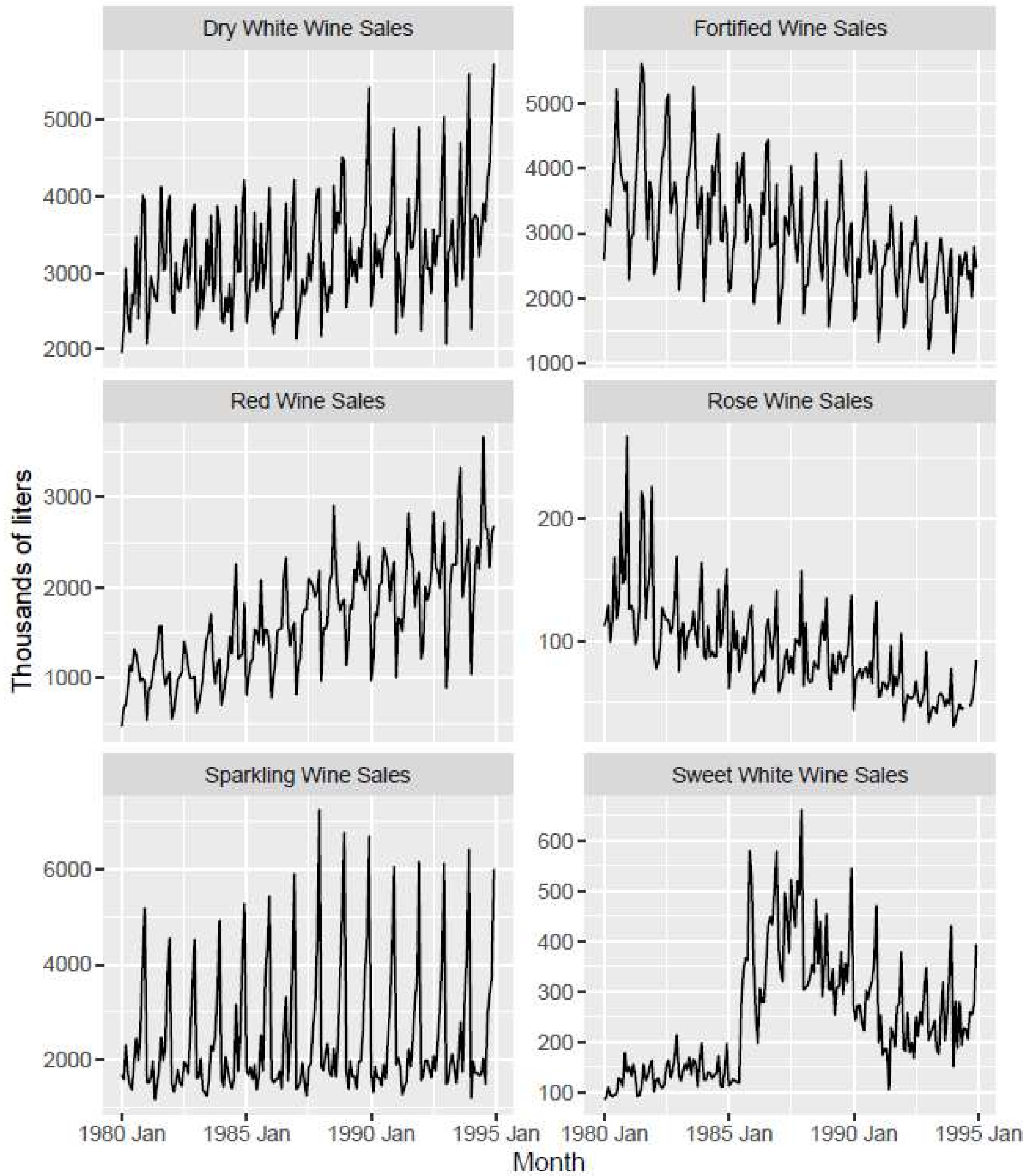
trend("N"),

season("A")

```
> report(fit.ets.auto)
Series: Ridership
Model: ETS(M,N,A)
  Smoothing parameters:
    alpha = 0.5550234
    gamma = 0.0001000004

  Initial states:
     l[0]     s[0]     s[-1]      s[-2]     s[-3]   s[-4]    s[-5]    s[-6]    s[-7]
 1807.921 27.01543 -11.25932 0.01014284 -121.9422 199.759 149.5166 37.72519 75.45785
    s[-8]    s[-9]    s[-10]   s[-11]
 59.73486 47.18978 -252.1883 -211.019

  sigma^2:  0.0012

     AIC     AICc      BIC
1615.746 1620.232 1657.929
```

Hands-on

Which smoothing method would you choose if you had to choose the same method for all series? Why?



Hands-on (#2.2)

- Apply Holt-Winter’s exponential smoothing (with multiplicative seasonality) to sales of fortified wine.
  Use the training data to fit your model.

Measuring forecasting accuracy

Forecast accuracy (or predictive accuracy) describes how closely forecasts are to the actual values. Hence, we compare the actual and forecast values and look directly at forecast errors.

Forecast error: $e_t = y_t - F_t$

- It is common to examine the predictive accuracy on the validation period data
- Several measures of predictive accuracy are commonly used:

Average error	$\frac{1}{v}\sum_{i=1}^{v} e_t$	MAE or MAD (mean absolute error/deviation)	$\frac{1}{v}\sum_{i=1}^{v} \left\lvert\frac{e_t}{y_t}\right\rvert \times 100$
MAPE (mean absolute percentage error)	$\frac{1}{v}\sum_{i=1}^{v} \lvert e_t \rvert$	RMSE (root mean squared error)	$\sqrt{\frac{1}{v}\sum_{i=1}^{v} e_t^2}$

Measuring forecasting accuracy

accuracy(model estimate object) – *Gives many accuracy measurements for* ***training*** *dataset.*

accuracy(forecast object, data set object) – *Gives many accuracy measurements for the* ***forecasting*** *dataset.*

```
fit.ets.HW <- train.ridership |>
                    model(ets = ETS(Ridership ~ error("M") + trend("A") + season("A")))
fc.ets.HW <- fit.ets.HW |> forecast(h = nrow(valid.ridership))

accuracy(fit.ets.HW)

accuracy( fc.ets.HW, ridership)
```

```
> accuracy(fit.ets.HW)
# A tibble: 1 × 10
  .model .type        ME  RMSE   MAE     MPE  MAPE  MASE RMSSE   ACF1
  <chr>  <chr>     <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>  <dbl>
1 ets    Training 0.138  56.5  45.2 -0.0792  2.58 0.548 0.569 0.0606
> accuracy( fc.ets.HW, ridership)
# A tibble: 1 × 10
  .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
  <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 ets    Test   33.7  76.7  62.2  1.58  3.12 0.754 0.772 0.617
>
```