MONASH University

REVOLUTION ANALYTICS

Rob J Hyndman

Forecasting using

7. Transformations and adjustments

OTexts.com/fpp/2/4/

Outline

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class `ets`.

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class `ets`.

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class `ets`.

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class `ets`.

Exponential smoothing

`ets()` function

- Automatically chooses a model by default using the AIC, AICc or BIC.
- Can handle any combination of trend, seasonality and damping
- Produces prediction intervals for every model
- Ensures the parameters are admissible (equivalent to invertible)
- Produces an object of class `ets`.

Exponential smoothing

`ets` objects

- **Methods:** `coef()`, `plot()`, `summary()`, `residuals()`, `fitted()`, `simulate()` and `forecast()`
- `plot()` function shows time plots of the original time series along with the extracted components (level, growth and seasonal).

Exponential smoothing

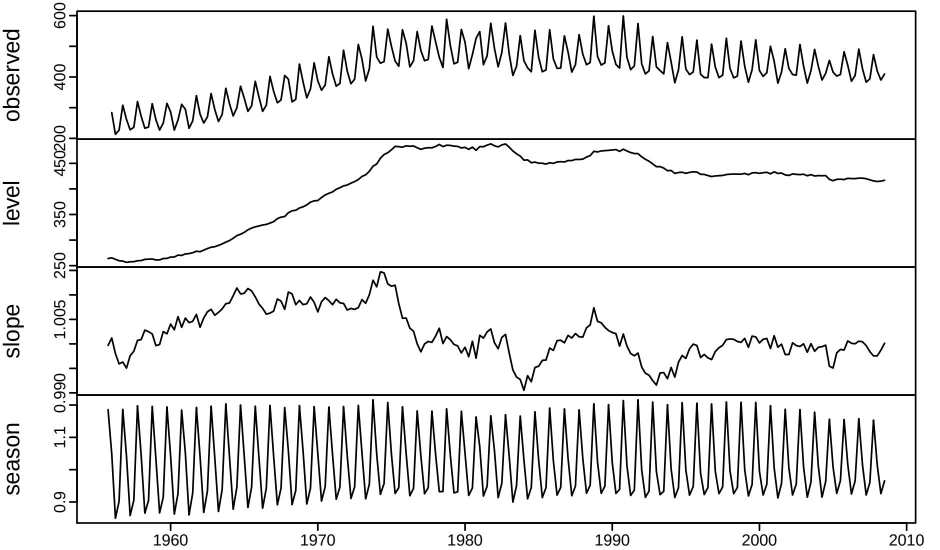
`ets` objects

- **Methods:** `coef()`, `plot()`, `summary()`, `residuals()`, `fitted()`, `simulate()` and `forecast()`
- `plot()` function shows time plots of the original time series along with the extracted components (level, growth and seasonal).

Exponential smoothing

`plot(fit)`

Decomposition by ETS(M,Md,M) method

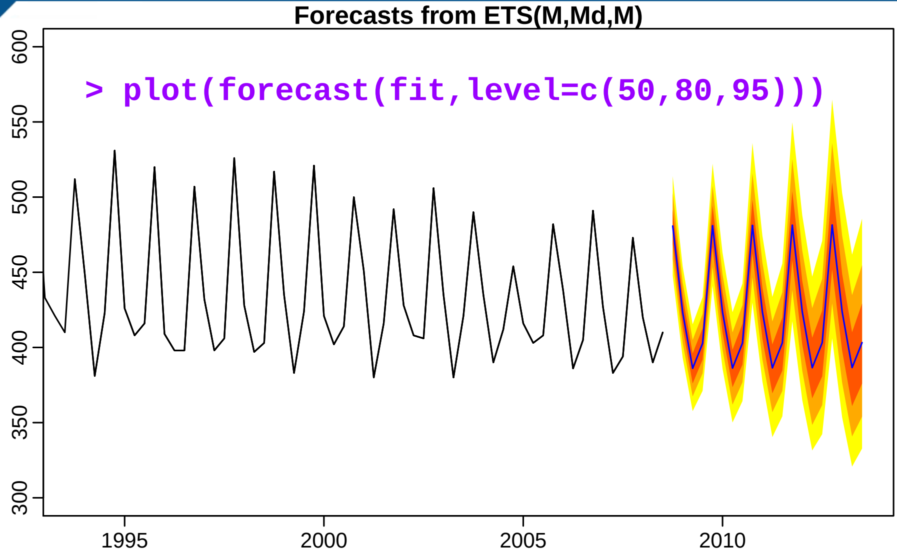


Goodness-of-fit

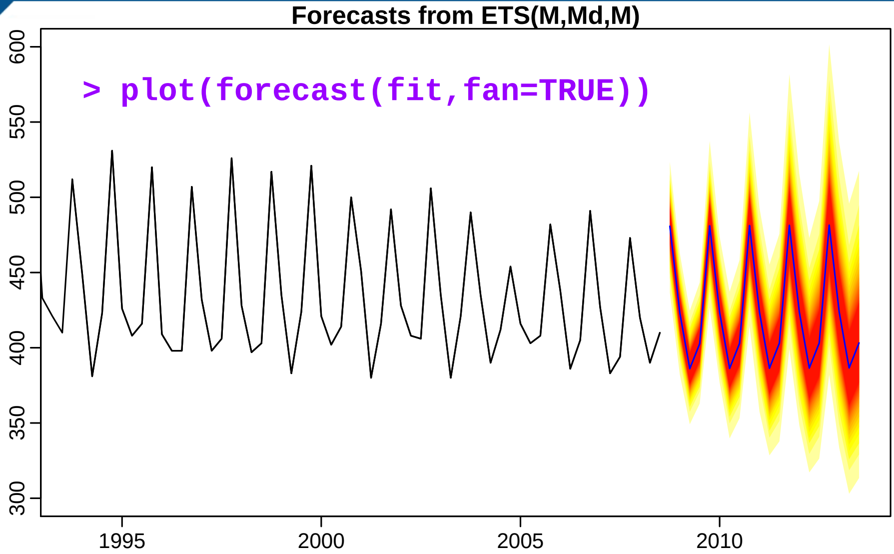
```
> accuracy(fit)
      ME     RMSE      MAE      MPE     MAPE     MASE
 0.17847 15.48781 11.77800  0.07204  2.81921  0.20705

> accuracy(fit2)
      ME     RMSE      MAE      MPE     MAPE     MASE
-0.11711 15.90526 12.18930 -0.03765  2.91255  0.21428
```

Forecast intervals



Forecast intervals



Exponential smoothing

`ets()` function also allows refitting model to new data set.

```
> usfit <- ets(usnetelec[1:45])
> test <- ets(usnetelec[46:55], model = usfit)

> accuracy(test)
      ME      RMSE       MAE       MPE      MAPE      MASE
-3.35419 58.02763 43.85545 -0.07624  1.18483  0.52452

> accuracy(forecast(usfit,10), usnetelec[46:55])
        ME        RMSE         MAE         MPE        MAPE         MASE
  40.7034   61.2075   46.3246    1.0980    1.2620    0.6776
```

Unstable models

- ETS(M,M,A)
- ETS(M,M_d,A)
- ETS(A,N,M)
- ETS(A,A,M)
- ETS(A,A_d,M)
- ETS(A,M,N)
- ETS(A,M,A)
- ETS(A,M,M)
- ETS(A,M_d,N)
- ETS(A,M_d,A)
- ETS(A,M_d,M)

Unstable models

- ETS(M,M,A)
- ETS(M,M_d,A)
- ETS(A,N,M)
- ETS(A,A,M)
- ETS(A,A_d,M)
- ETS(A,M,N)
- ETS(A,M,A)
- ETS(A,M,M)
- ETS(A,M_d,N)
- ETS(A,M_d,A)
- ETS(A,M_d,M)

In practice, the models work fine for short- to medium-term forecasts provided the data are strictly positive.

Forecastability conditions

```
ets(y, model="ZZZ", damped=NULL, alpha=NULL,
    beta=NULL, gamma=NULL, phi=NULL,
    additive.only=FALSE,
    lower=c(rep(0.0001,3),0.80),
    upper=c(rep(0.9999,3),0.98),
    opt.crit=c("lik","amse","mse","sigma"),
    nmse=3,
    bounds=c("both","usual","admissible"),
    ic=c("aic","aicc","bic"), restrict=TRUE)
```

The magic `forecast()` function

- `forecast` returns forecasts when applied to an `ets` object (or the output from many other time series models).
- If you use `forecast` directly on data, it will select an ETS model automatically and then return forecasts.

The magic `forecast()` function

- `forecast` returns forecasts when applied to an `ets` object (or the output from many other time series models).
- If you use `forecast` directly on data, it will select an ETS model automatically and then return forecasts.

Outline

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.

Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

Mathematical transformations for stabilizing variation

Square root	$w_t = \sqrt{y_t}$	$\downarrow$
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

Logarithms, in particular, are useful because they are more interpretable: changes in a log value are **relative (percent) changes on the original scale.**

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.
Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

Mathematical transformations for stabilizing variation

Square root	$w_t = \sqrt{y_t}$	$\downarrow$
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

Logarithms, in particular, are useful because they are more interpretable: changes in a log value are **relative (percent) changes on the original scale.**

Transformations to stabilize the variance

If the data show different variation at different levels of the series, then a transformation can be useful.
Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

Mathematical transformations for stabilizing variation

Square root	$w_t = \sqrt{y_t}$	$\downarrow$
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

Logarithms, in particular, are useful because they are more interpretable: changes in a log value are **relative (percent) changes on the original scale.**

Transformations to stabilize the variance

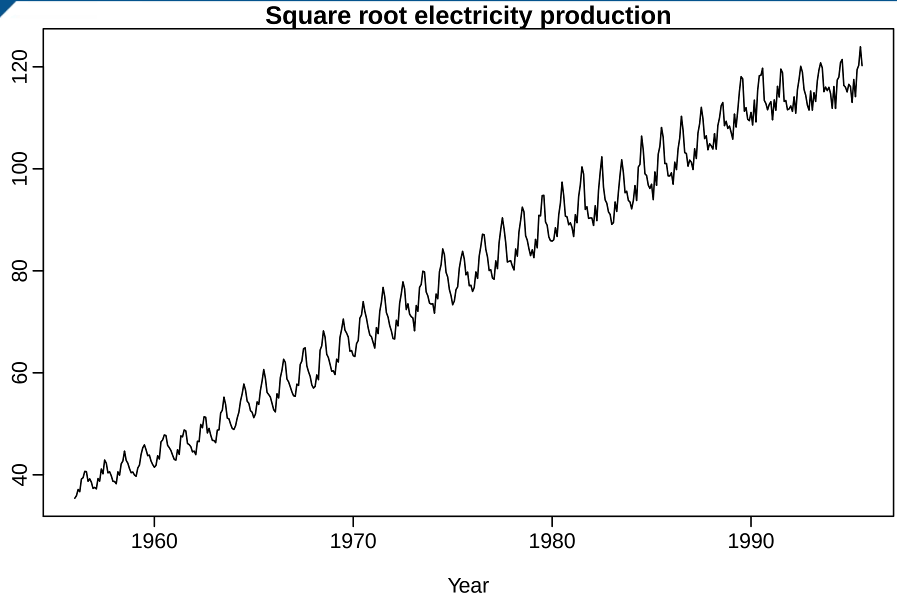
If the data show different variation at different levels of the series, then a transformation can be useful.
Denote original observations as $y_1, \dots, y_n$ and transformed observations as $w_1, \dots, w_n$.

Mathematical transformations for stabilizing variation

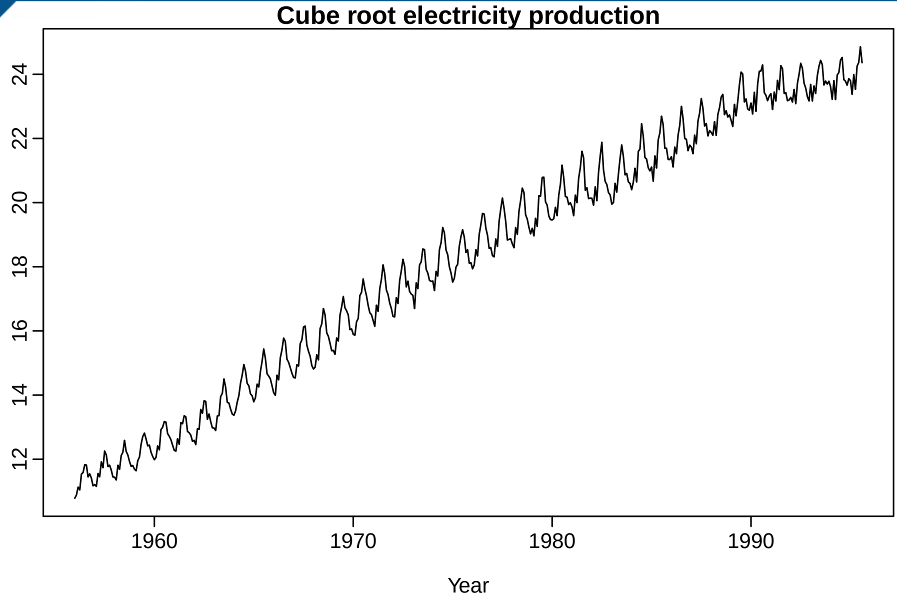
Square root	$w_t = \sqrt{y_t}$	$\downarrow$
Cube root	$w_t = \sqrt[3]{y_t}$	Increasing
Logarithm	$w_t = \log(y_t)$	strength

Logarithms, in particular, are useful because they are more interpretable: changes in a log value are **relative (percent) changes on the original scale.**

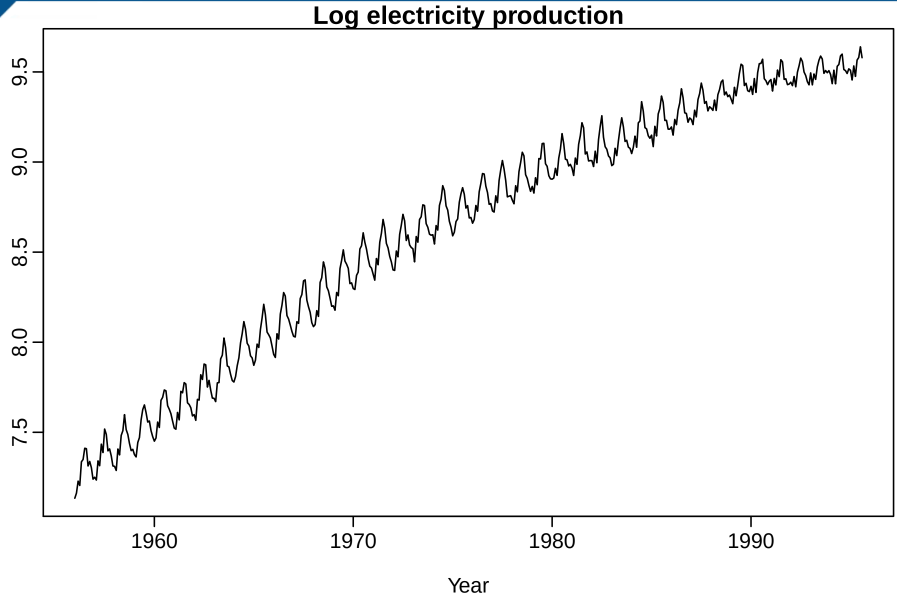
Transformations



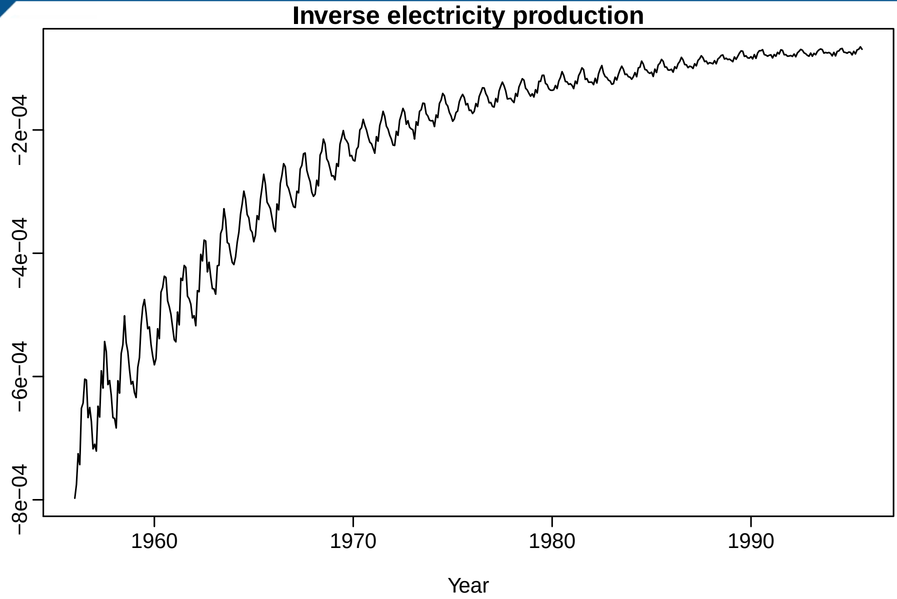
Transformations



Transformations



Transformations



Box-Cox transformations

Each of these transformations is close to a member of the family of **Box-Cox transformations**:

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

Box-Cox transformations

Each of these transformations is close to a member of the family of **Box-Cox transformations**:

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

Box-Cox transformations

Each of these transformations is close to a member of the family of **Box-Cox transformations**:

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

Box-Cox transformations

Each of these transformations is close to a member of the family of **Box-Cox transformations**:

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

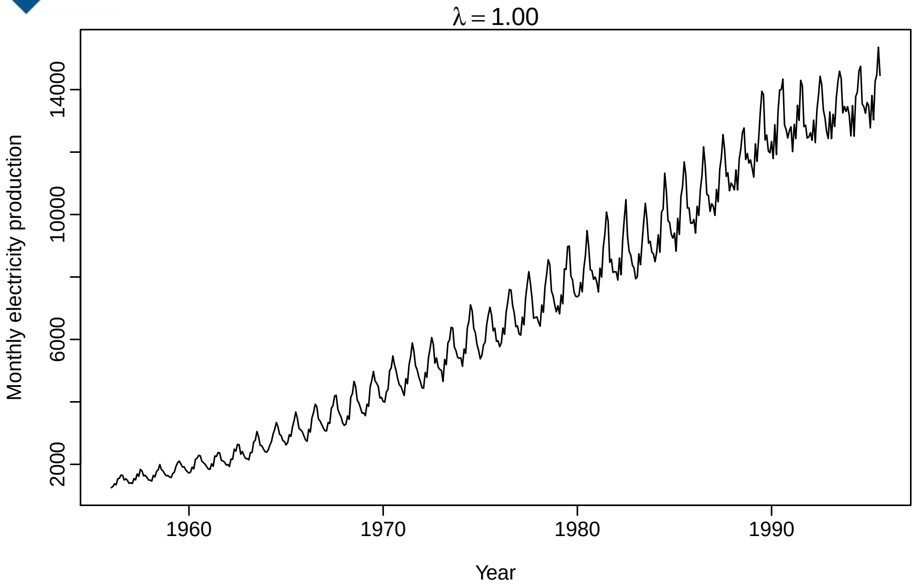
Box-Cox transformations

Each of these transformations is close to a member of the family of **Box-Cox transformations**:

$$w_t = \begin{cases} \log(y_t), & \lambda = 0; \\ (y_t^\lambda - 1)/\lambda, & \lambda \neq 0. \end{cases}$$

- $\lambda = 1$: (No substantive transformation)
- $\lambda = \frac{1}{2}$: (Square root plus linear transformation)
- $\lambda = 0$: (Natural logarithm)
- $\lambda = -1$: (Inverse plus 1)

Box-Cox transformations



Box-Cox transformations

- y_t^{λ} for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

Box-Cox transformations

- y_t^λ for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

Box-Cox transformations

- y_t^λ for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

Box-Cox transformations

- y_t^λ for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

Box-Cox transformations

- y_t^λ for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

Box-Cox transformations

- y_t^λ for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

Box-Cox transformations

- y_t^λ for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

Box-Cox transformations

- y_t^λ for λ close to zero behaves like logs.
- If some $y_t = 0$, then must have $\lambda > 0$
- if some $y_t < 0$, no power transformation is possible unless all y_t adjusted by **adding a constant to all values**.
- Choose a simple value of λ. It makes explanation easier.
- Results are relatively insensitive to value of λ
- Often no transformation ($\lambda = 1$) needed.
- Transformation often makes little difference to forecasts but has large effect on PI.
- Choosing $\lambda = 0$ is a simple way to force forecasts to be positive

Back-transformation

We must reverse the transformation (or *back-transform*) to obtain forecasts on the original scale. The reverse Box-Cox transformations are given by

$$y_t = \begin{cases} \exp(w_t), & \lambda = 0; \\ (\lambda W_t + 1)^{1/\lambda}, & \lambda \neq 0. \end{cases}$$

```
plot(BoxCox(elec,lambda=1/3))
fit <- snaive(elec, lambda=1/3)
plot(fit)
plot(fit, include=120)
```

Back-transformation

We must reverse the transformation (or *back-transform*) to obtain forecasts on the original scale. The reverse Box-Cox transformations are given by

$$y_t = \begin{cases} \exp(w_t), & \lambda = 0; \\ (\lambda W_t + 1)^{1/\lambda}, & \lambda \neq 0. \end{cases}$$

```
plot(BoxCox(elec,lambda=1/3))
fit <- snaive(elec, lambda=1/3)
plot(fit)
plot(fit, include=120)
```

Automated Box-Cox transformations

```
BoxCox.lambda(elec)
```

- This attempts to balance the seasonal fluctuations and random variation across the series.
- Always check the results.
- A low value of λ can give extremely large prediction intervals.

Automated Box-Cox transformations

```
BoxCox.lambda(elec)
```

- This attempts to balance the seasonal fluctuations and random variation across the series.
- Always check the results.
- A low value of λ can give extremely large prediction intervals.

Automated Box-Cox transformations

```
BoxCox.lambda(elec)
```

- This attempts to balance the seasonal fluctuations and random variation across the series.
- Always check the results.
- A low value of λ can give extremely large prediction intervals.

Automated Box-Cox transformations

```
BoxCox.lambda(elec)
```

- This attempts to balance the seasonal fluctuations and random variation across the series.
- Always check the results.
- A low value of λ can give extremely large prediction intervals.

ETS and transformations

- A Box-Cox transformation followed by an additive ETS model is often better than an ETS model without transformation.
- A Box-Cox transformation followed by STL + ETS is often better than an ETS model without transformation.
- It makes no sense to use a Box-Cox transformation and a *non-additive* ETS model.

ETS and transformations

- A Box-Cox transformation followed by an additive ETS model is often better than an ETS model without transformation.
- A Box-Cox transformation followed by STL + ETS is often better than an ETS model without transformation.
- It makes no sense to use a Box-Cox transformation and a *non-additive* ETS model.

ETS and transformations

- A Box-Cox transformation followed by an additive ETS model is often better than an ETS model without transformation.
- A Box-Cox transformation followed by STL + ETS is often better than an ETS model without transformation.
- It makes no sense to use a Box-Cox transformation and a *non-additive* ETS model.

Outline

Calendar adjustments

Some of the variation in a time series may be due to the variation in the number of trading days each month. It is a good idea to adjust for this known source of variation to allow study of other interesting features.

- Month length
- Trading day

Calendar adjustments

Some of the variation in a time series may be due to the variation in the number of trading days each month. It is a good idea to adjust for this known source of variation to allow study of other interesting features.

- Month length
- Trading day

Month length adjustment

If this is not removed, it shows up as a seasonal effect, which may not cause problems though it does make any seasonal pattern hard to interpret. It is easily adjusted for:

$$
\begin{aligned}
y_t^* &= y_t \times \frac{\text{no. of days in an average month}}{\text{no. of days in month } t} \\
&= y_t \times \frac{365.25/12}{\text{no. of days in month } t}
\end{aligned}
$$

where y_t has already been transformed if necessary.

`monthdays` gives the number of days in each month or quarter.

Month length adjustment

If this is not removed, it shows up as a seasonal effect, which may not cause problems though it does make any seasonal pattern hard to interpret. It is easily adjusted for:

$$y_t^* = y_t \times \frac{\text{no. of days in an average month}}{\text{no. of days in month } t} = y_t \times \frac{365.25/12}{\text{no. of days in month } t}$$

where y_t has already been transformed if necessary.

`monthdays` gives the number of days in each month or quarter.

Trading day adjustment

- occurs in monthly data when there is also a weekly cycle, since proportions of various days in given month vary from year to year.
- number of trading days is predictable, but effects of various days are unknown.
- **Simplest case:** All trading days assumed to have same effect.

$$y_t^* = y_t \times \frac{\text{no. of trading days in average month}}{\text{no. of trading days in month } t}.$$

  where y_t has already been adjusted for month length and transformed if necessary.
- If weekly cycle more complex, these effects must be estimated from data.

Trading day adjustment

- occurs in monthly data when there is also a weekly cycle, since proportions of various days in given month vary from year to year.
- number of trading days is predictable, but effects of various days are unknown.
- **Simplest case:** All trading days assumed to have same effect.

$$y_t^* = y_t \times \frac{\text{no. of trading days in average month}}{\text{no. of trading days in month } t}.$$

  where y_t has already been adjusted for month length and transformed if necessary.
- If weekly cycle more complex, these effects must be estimated from data.

Trading day adjustment

- occurs in monthly data when there is also a weekly cycle, since proportions of various days in given month vary from year to year.
- number of trading days is predictable, but effects of various days are unknown.
- **Simplest case:** All trading days assumed to have same effect.

$$y_t^* = y_t \times \frac{\text{no. of trading days in average month}}{\text{no. of trading days in month } t}.$$

  where y_t has already been adjusted for month length and transformed if necessary.
- If weekly cycle more complex, these effects must be estimated from data.

Trading day adjustment

- occurs in monthly data when there is also a weekly cycle, since proportions of various days in given month vary from year to year.
- number of trading days is predictable, but effects of various days are unknown.
- **Simplest case:** All trading days assumed to have same effect.

$$y_t^* = y_t \times \frac{\text{no. of trading days in average month}}{\text{no. of trading days in month } t}.$$

  where y_t has already been adjusted for month length and transformed if necessary.
- If weekly cycle more complex, these effects must be estimated from data.

Explainable variation

Examples:

- Calendar variation
- Increasing population
- Inflation
- Strikes
- Changes in government
- Changes in law

Try and understand all possible sources of variation before modelling the time series.

Explainable variation

Examples:

- Calendar variation
- Increasing population
- Inflation
- Strikes
- Changes in government
- Changes in law

Try and understand all possible sources of variation before modelling the time series.

Explainable variation

Examples:

- Calendar variation
- Increasing population
- Inflation
- Strikes
- Changes in government
- Changes in law

Try and understand all possible sources of variation before modelling the time series.

Explainable variation

Examples:

- Calendar variation
- Increasing population
- Inflation
- Strikes
- Changes in government
- Changes in law

Try and understand all possible sources of variation before modelling the time series.

Explainable variation

Examples:

- Calendar variation
- Increasing population
- Inflation
- Strikes
- Changes in government
- Changes in law

Try and understand all possible sources of variation before modelling the time series.

Explainable variation

Examples:

- Calendar variation
- Increasing population
- Inflation
- Strikes
- Changes in government
- Changes in law

Try and understand all possible sources of variation before modelling the time series.

Explainable variation

Examples:

- Calendar variation
- Increasing population
- Inflation
- Strikes
- Changes in government
- Changes in law

Try and understand all possible sources of variation before modelling the time series.

Explainable variation

Examples:

- Calendar variation
- Increasing population
- Inflation
- Strikes
- Changes in government
- Changes in law

Try and understand all possible sources of variation before modelling the time series.