Transformations for variance stabilization

FORECASTING IN R

Rob J. Hyndman
Professor of Statistics at Monash University

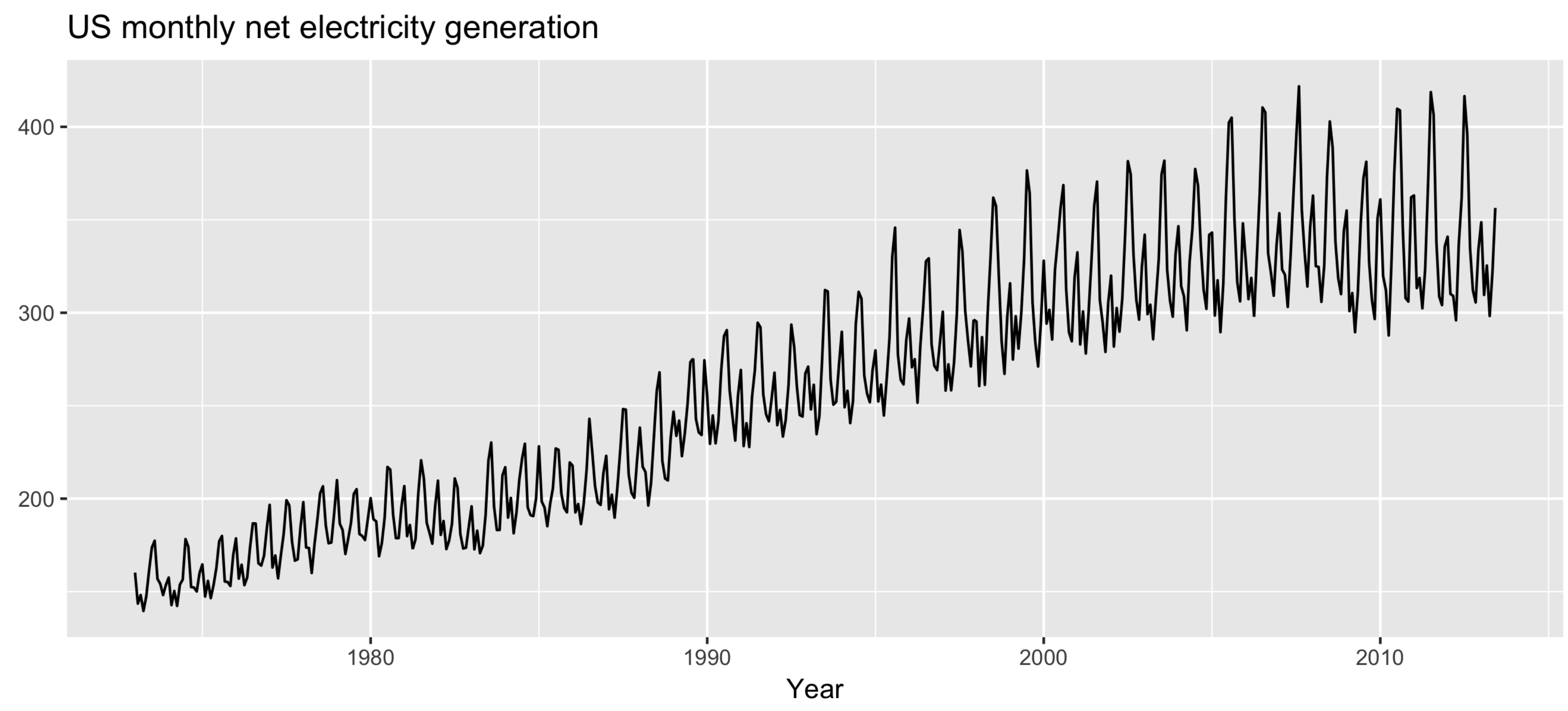
Variance stabilization

- If the data show increasing variation as the level of the series increases, then a **transformation** can be useful
- $y_1, \ldots, y_n$: original observations, $w_1, \ldots, w_n$: transformed observations

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
Inverse	$w_t = -1/y_t$	$\downarrow$

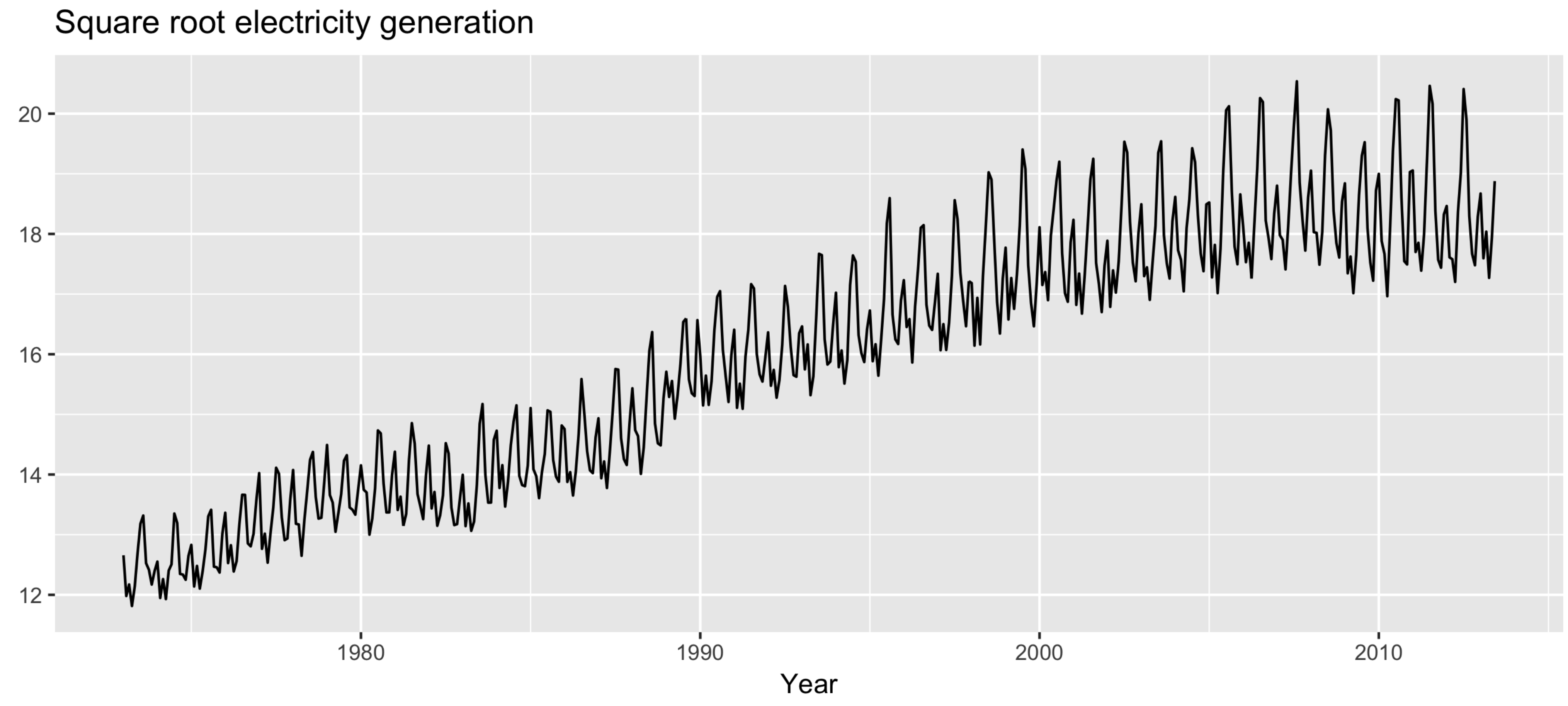
Variance stabilization

```
autoplot(usmelec) +
  xlab("Year") + ylab("") +
  ggtitle("US monthly net electricity generation")
```



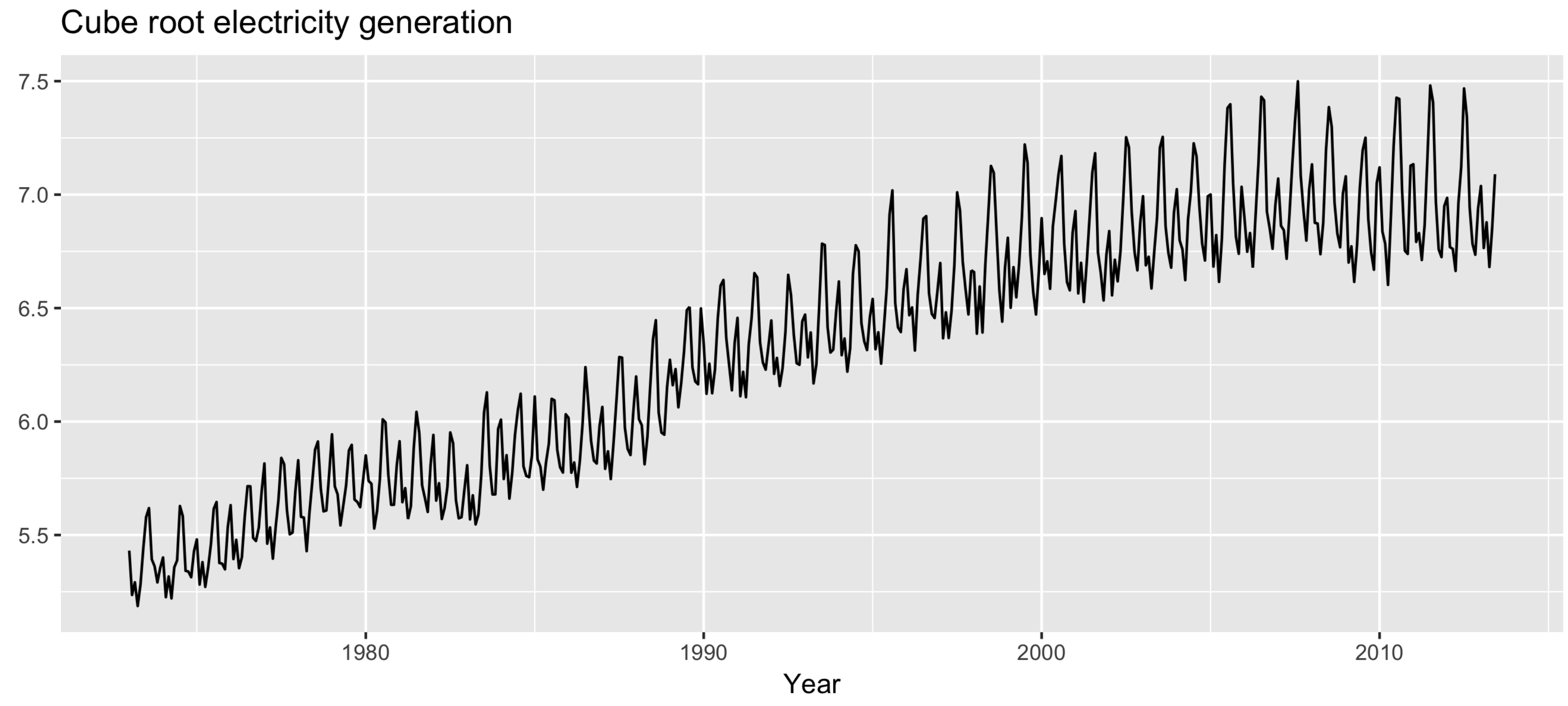
Variance stabilization

```
autoplot(usmelec^0.5) +
  xlab("Year") + ylab("") +
  ggtitle("Square root electricity generation")
```



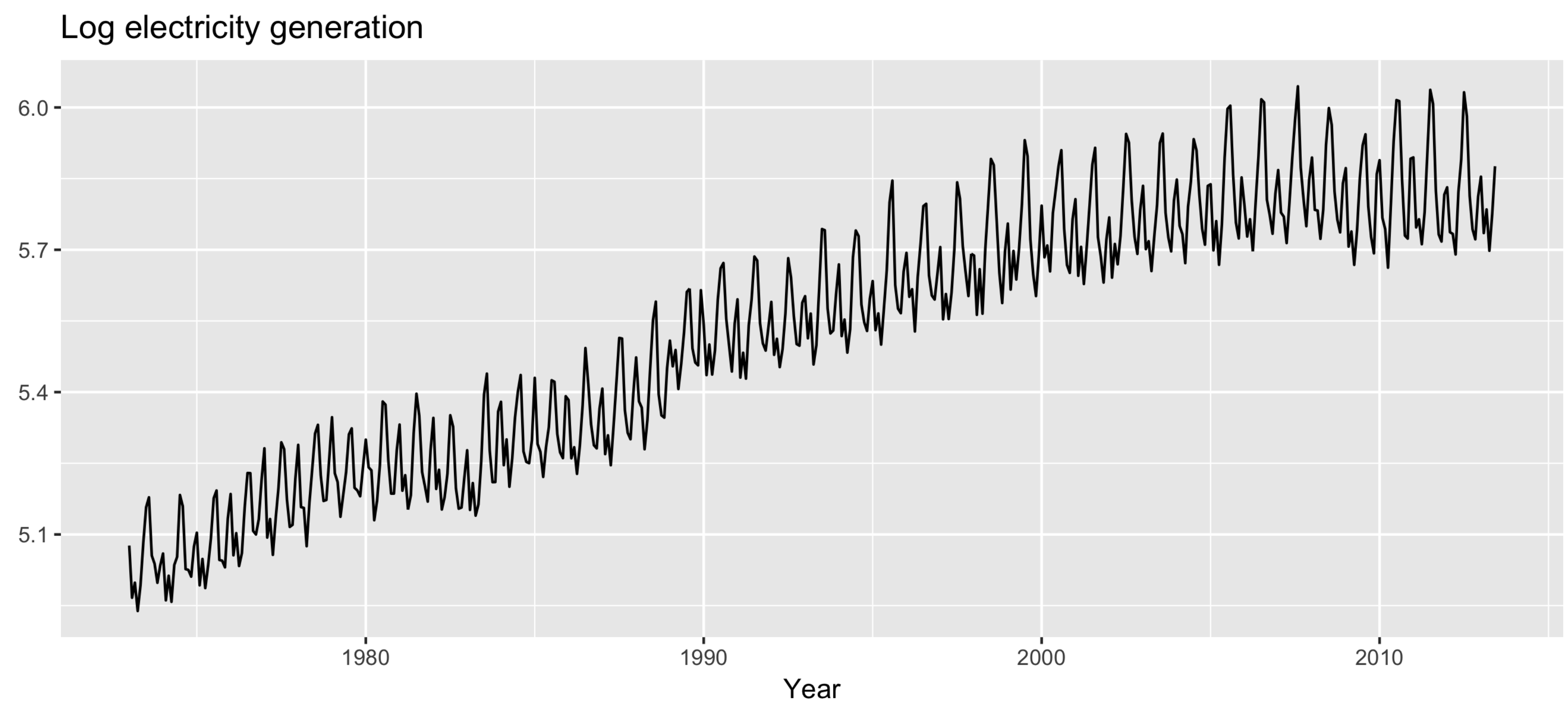
Variance stabilization

```
autoplot(usmelec^0.33333) +
  xlab("Year") + ylab("") +
  ggtitle("Cube root electricity generation")
```



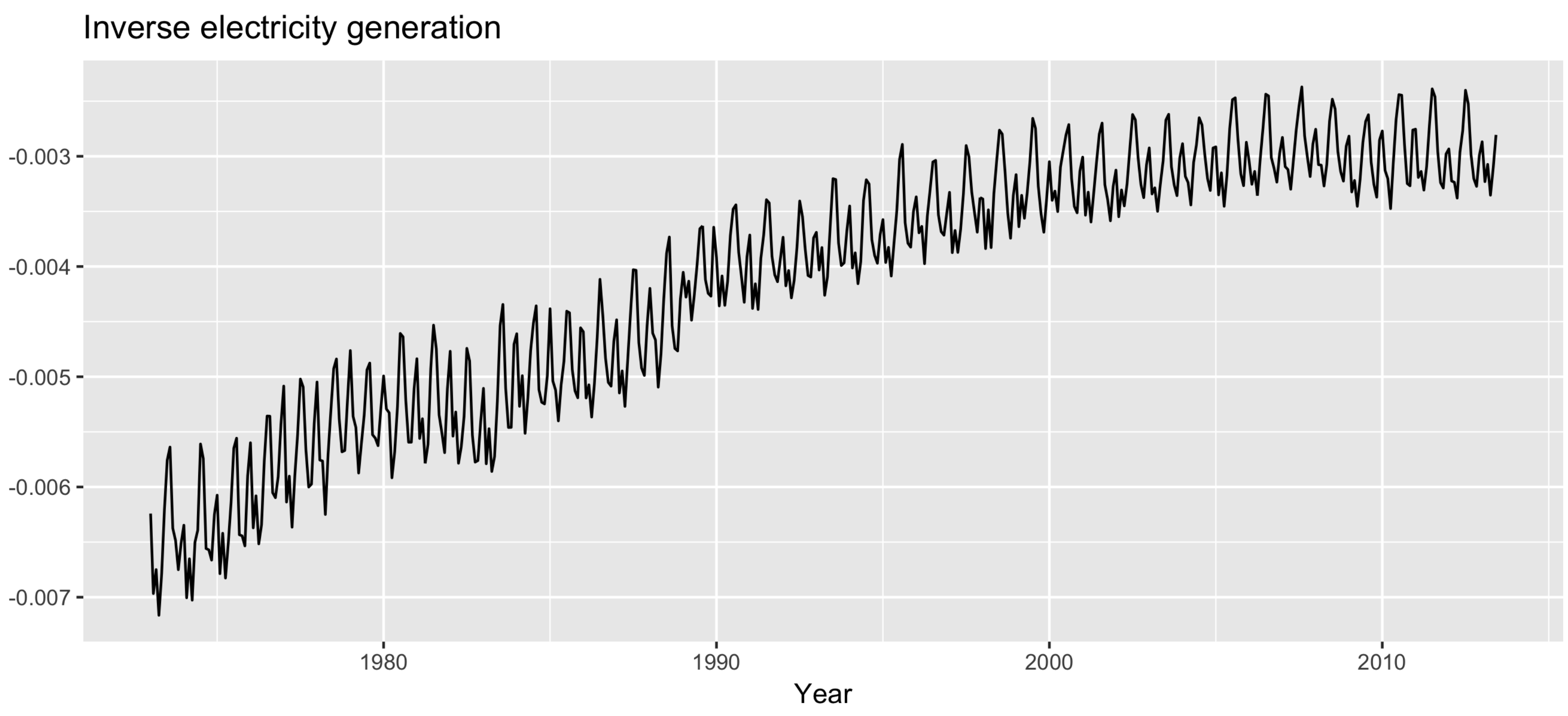
Variance stabilization

```
autoplot(log(usmelec)) +
  xlab("Year") + ylab("") +
  ggtitle("Log electricity generation")
```



Variance stabilization

```r
autoplot(-1/usmelec) +
  xlab("Year") + ylab("") +
  ggtitle("Inverse electricity generation")
```



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
- $\lambda = \frac{1}{3}$: Cube root plus linear transformation
- $\lambda = 0$: Natural logarithm transformation
- $\lambda = -1$: Inverse transformation

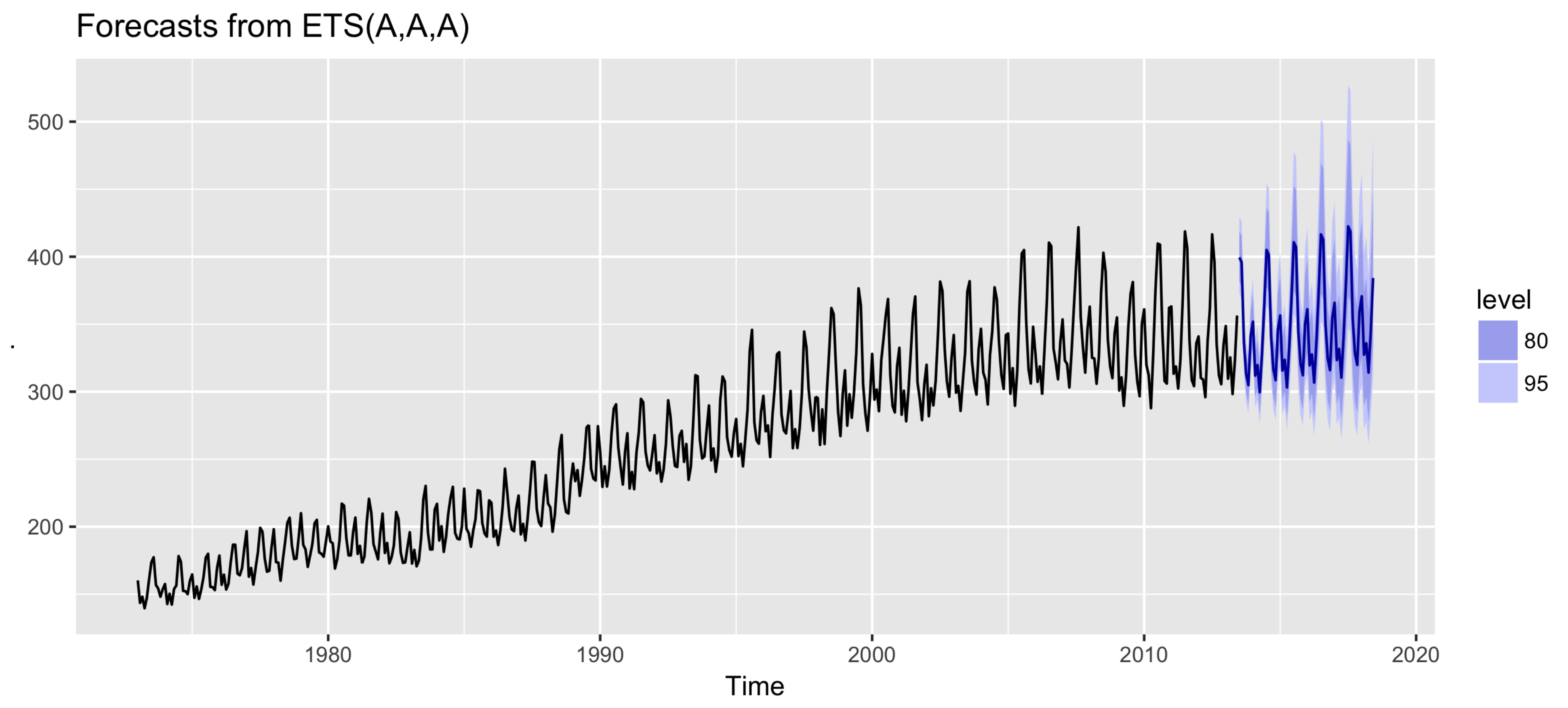
Box-Cox transformations

```
BoxCox.lambda(usmelec)
```

```
-0.5738331
```

Back-transformation

```
usmelec %>%
  ets(lambda = -0.57) %>%
  forecast(h = 60) %>%
  autoplot()
```



Let's practice!

FORECASTING IN R

ARIMA models

FORECASTING IN R

Rob J. Hyndman
Professor of Statistics at Monash University

ARIMA models

Autoregressive (AR) models
$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t, \qquad e_t \sim \text{white noise}$
Multiple regression with lagged observations as predictors

ARIMA models

Autoregressive (AR) models
$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t, \qquad e_t \sim \text{white noise}$
Multiple regression with lagged observations as predictors

Moving Average (MA) models
$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}, \qquad e_t \sim \text{white noise}$
Multiple regression with lagged errors as predictors

ARIMA models

Autoregressive (AR) models
$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t, \qquad e_t \sim \text{white noise}$
Multiple regression with lagged observations as predictors

Moving Average (MA) models
$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}, \qquad e_t \sim \text{white noise}$
Multiple regression with lagged errors as predictors

Autoregressive Moving Average (ARMA) models
$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$
Multiple regression with lagged observations and lagged errors as

ARIMA models

Autoregressive (AR) models
$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t, \qquad e_t \sim \text{white noise}$
Multiple regression with lagged observations as predictors

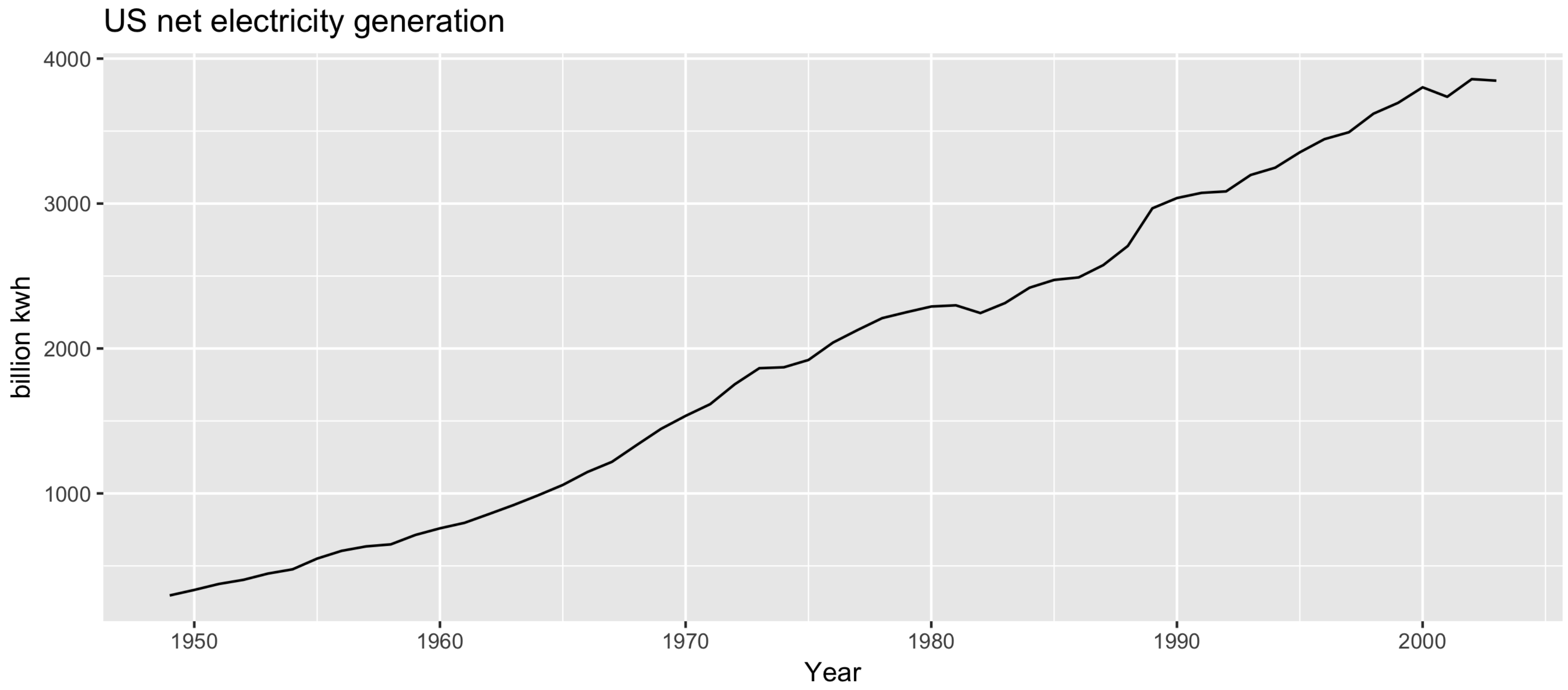
Moving Average (MA) models
$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}, \qquad e_t \sim \text{white noise}$
Multiple regression with lagged errors as predictors

Autoregressive Moving Average (ARMA) models
$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t$
Multiple regression with lagged observations and lagged errors as

ARIMA(p, d, q) models
Combine ARMA model with d - lots of differencing

US net electricity generation

```
autoplot(usnetelec) +
  xlab("Year") +
  ylab("billion kwh") +
  ggtitle("US net electricity generation")
```



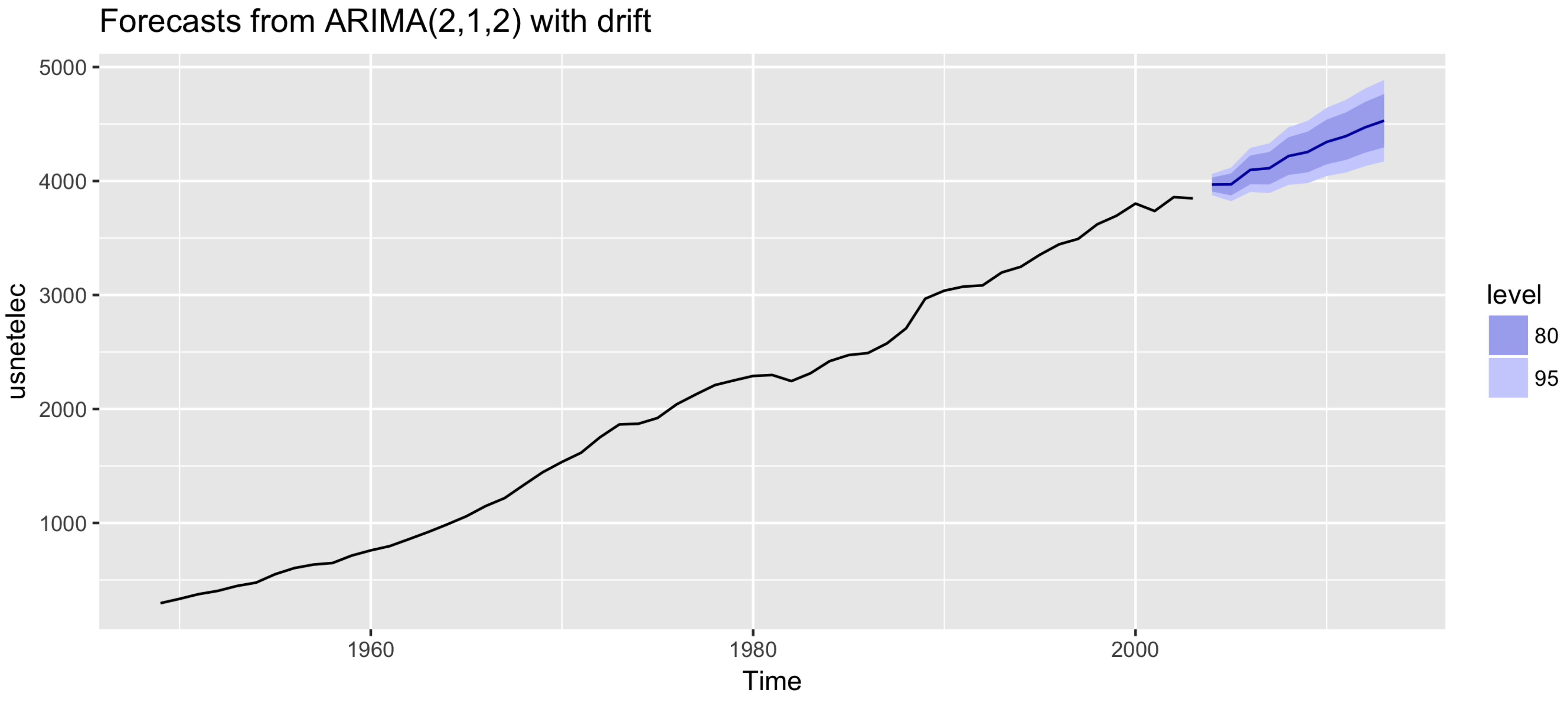
US net electricity generation

```
fit <- auto.arima(usnetelec)
summary(fit)
```

```
Series: usnetelec
ARIMA(2,1,2) with drift
Coefficients:
         ar1      ar2    ma1    ma2   drift
      -1.303  -0.433  1.528  0.834  66.159
s.e.   0.212   0.208  0.142  0.119   7.559
sigma^2 estimated as 2262:  log likelihood=-283.3
AIC=578.7   AICc=580.5   BIC=590.6
Training set error measures:
                 ME  RMSE   MAE     MPE  MAPE   MASE    ACF1
Training set 0.0464 44.89 32.33 -0.6177 2.101 0.4581 0.02249
```

US net electricity generation

```
fit %>% forecast() %>% autoplot()
```



How does auto.arima() work?

Hyndman-Khandakar algorithm:

- Select number of differences d via unit root tests
- Select p and q by minimizing AIC_c
- Estimate parameters using *maximum likelihood estimation*
- Use stepwise search to traverse model space, to save time

Let's practice!

FORECASTING IN R

Seasonal ARIMA models

FORECASTING IN R

Rob J. Hyndman
Professor of Statistics at Monash University

ARIMA models

ARIMA	(p, d, q)	(P, D, Q)m
	Non-seasonal part of the model	Seasonal part of the model

- d = Number of lag-1 differences
- p = Number of ordinary AR lags:
- q = Number of ordinary MA lags:

ARIMA models

ARIMA	(p, d, q)	(P, D, Q)m
	Non-seasonal part of the model	Seasonal part of the model

- d = Number of lag-1 differences
- p = Number of ordinary AR lags:
- q = Number of ordinary MA lags:

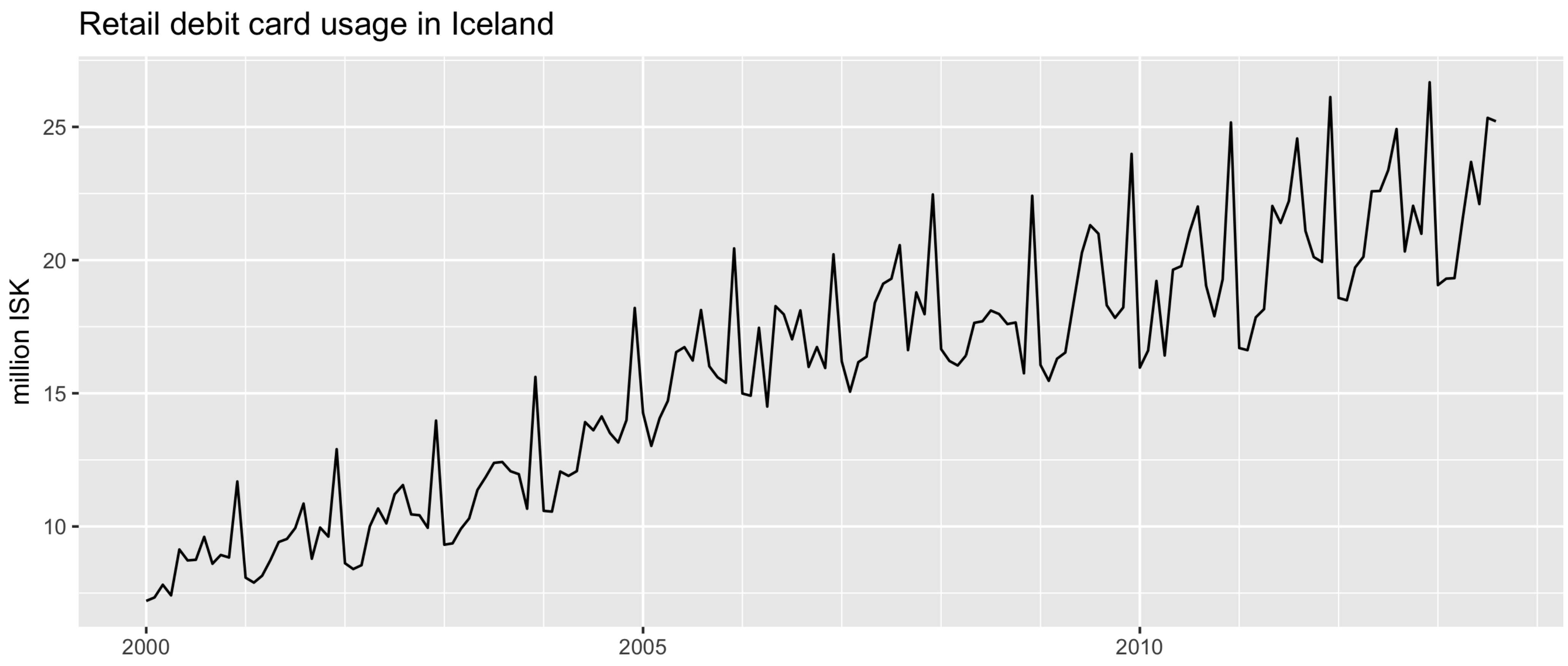
ARIMA models

ARIMA	(p, d, q)	(P, D, Q)m
	Non-seasonal part of the model	Seasonal part of the model

- d = Number of lag-1 differences
- p = Number of ordinary AR lags: $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$
- q = Number of ordinary MA lags: $\epsilon_{t-1}, \epsilon_{t-2}, \ldots, \epsilon_{t-q}$
- D = Number of seasonal differences
- P = Number of seasonal AR lags: $y_{t-m}, y_{t-2m}, \ldots, y_{t-Pm}$
- Q = Number of seasonal MA lags: $\epsilon_{t-m}, \epsilon_{t-2m}, \ldots, \epsilon_{t-Qm}$
- m = Number of observations per year

Example: Monthly retail debit card usage in Iceland

```
autoplot(debitcards) +
  xlab("Year") + ylab("million ISK") +
  ggtitle("Retail debit card usage in Iceland")
```



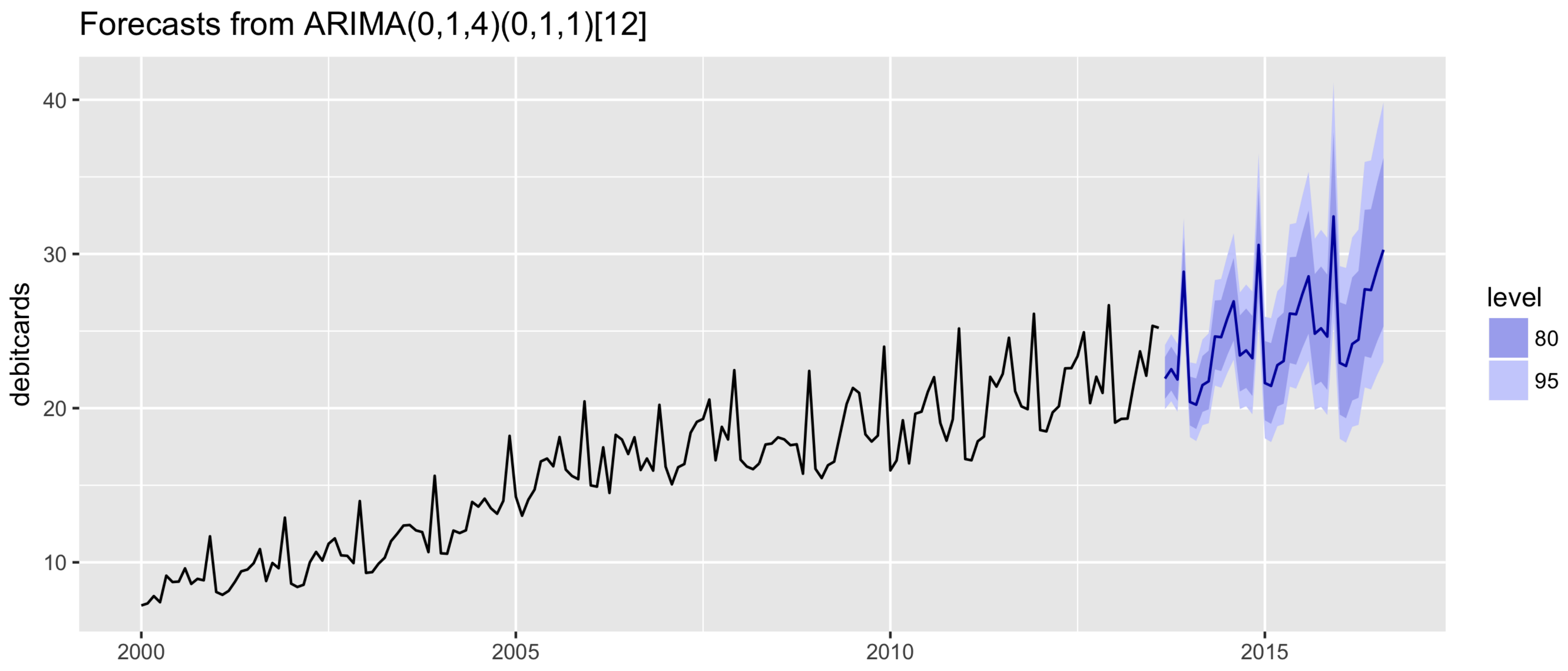
Example: Monthly retail debit card usage in Iceland

```
fit <- auto.arima(debitcards, lambda = 0)
fit
```

```
Series: debitcards
ARIMA(0,1,4)(0,1,1)[12]
Box Cox transformation: lambda= 0
Coefficients:
         ma1    ma2    ma3     ma4    sma1
      -0.796  0.086  0.263  -0.175  -0.814
s.e.   0.082  0.099  0.100   0.080   0.112
sigma^2 estimated as 0.00232:  log likelihood=239.3
AIC=-466.7   AICc=-466.1   BIC=-448.6
```

Example: Monthly retail debit card usage in Iceland

```
fit %>%
  forecast(h = 36) %>%
  autoplot() + xlab("Year")
```



Let's practice!

FORECASTING IN R