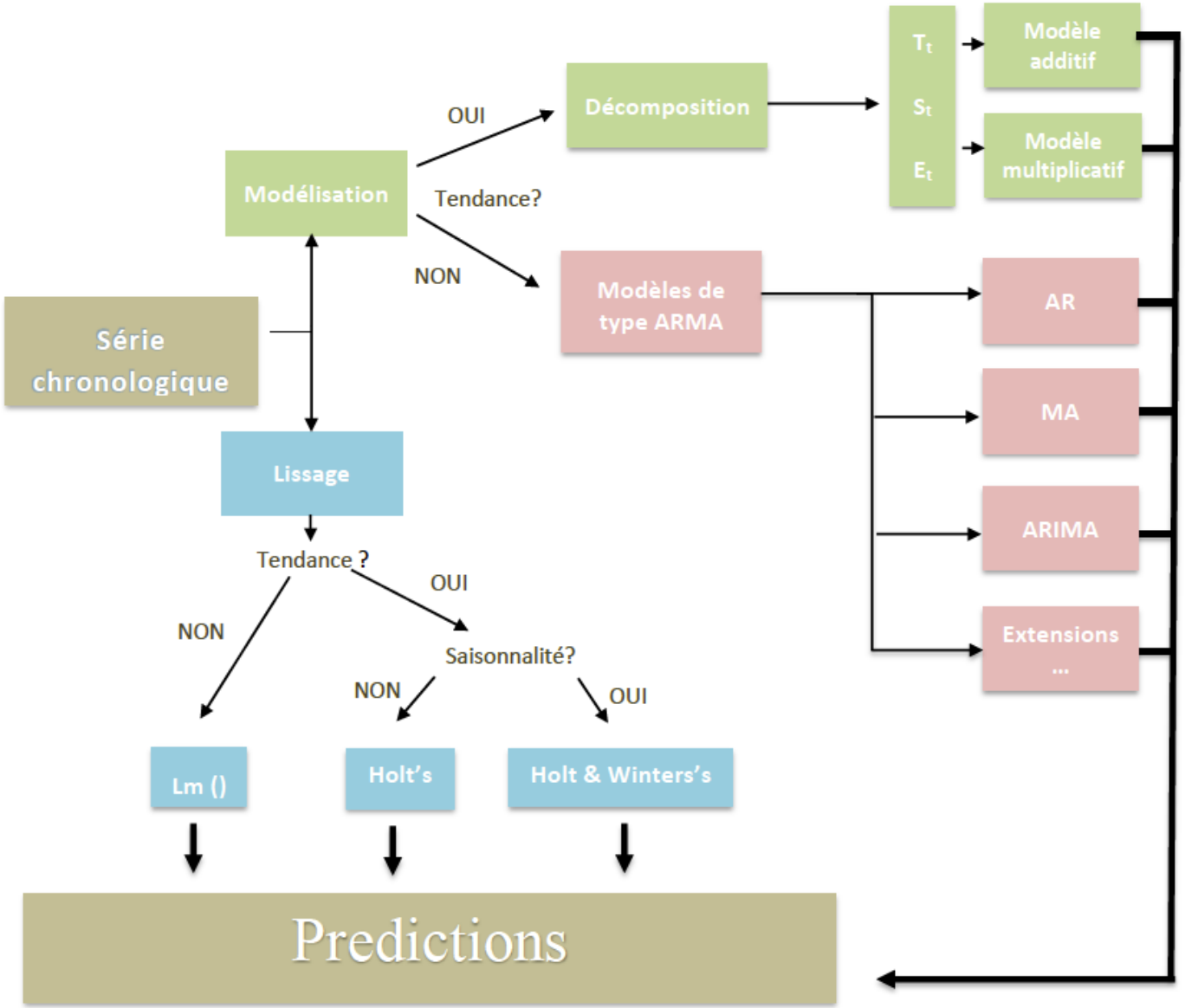
An introduction to Time Series using R

Benoit Liquet

October 25, 2016

Outline


Série chronologique
Modélisation
OUI
Décomposition
Tendance?
NON
Modèles de type ARMA
T_t
S_t
E_t
Modèle additif
Modèle multiplicatif
AR
MA
ARIMA
Extensions ...
Lissage
Tendance ?
NON
OUI
Saisonnalité?
NON
OUI
Lm ()
Holt's
Holt & Winters's
Predictions

Materials for time series

- Free online book:
  Time Series Analysis and Its Applications with R Examples
  http://www.stat.pitt.edu/stoffer/tsa3/

- Free online book:
  Forecasting: principles and practice
  https://www.otexts.org/fpp/

R package for time series

```
require(fpp)
```

```
require(forecast)
require(astsa)
```

Data used during the lecture: `CardioVascMort.tsm`

Time Series: Definition

Definition: analysis of experimental data that have been observed at different points in time.

Dependance: the obvious correlation introduced by the sampling of adjacent points in time can severely restrict the applicability of the many conventional statistical methods traditionally dependent on the assumption that these adjacent observations are independent and identically distributed.

Scientific applications:

- Economics, where we are continually exposed to daily stock market quotations or monthly unemployment figures.
- Social scientists follow population series, such as birthrates or school enrollments.
- An epidemiologist might be interested in the number of influenza cases observed over some time period.
- In medicine, blood pressure measurements traced over time could be useful for evaluating drugs used in treating hy- pertension.
- Functional magnetic resonance imaging of brain-wave time series patterns might be used to study how the brain reacts to certain stimuli under various experimental conditions.

Physical and environmental sciences

One of the earliest recorded series is the monthly sunspot numbers studied by Schuster (1906). More modern investigations may center on

- whether a warming is present in global temperature measurements
- whether levels of pollution may influence daily mortality in Los Angeles.
- Seismic recordings can aid in mapping fault lines or in distinguishing between earthquakes and nuclear explosions.

Physical and environmental sciences

One of the earliest recorded series is the monthly sunspot numbers studied by Schuster (1906). More modern investigations may center on

- whether a warming is present in global temperature measurements
- whether levels of pollution may influence daily mortality in Los Angeles.
- Seismic recordings can aid in mapping fault lines or in distinguishing between earthquakes and nuclear explosions.

First step analysis any time series investigation always involves careful scrutiny of the recorded data plotted over time.

$\hookrightarrow$ suggests the method of analysis as well as statistics that will be of use in summarizing the information in the data.

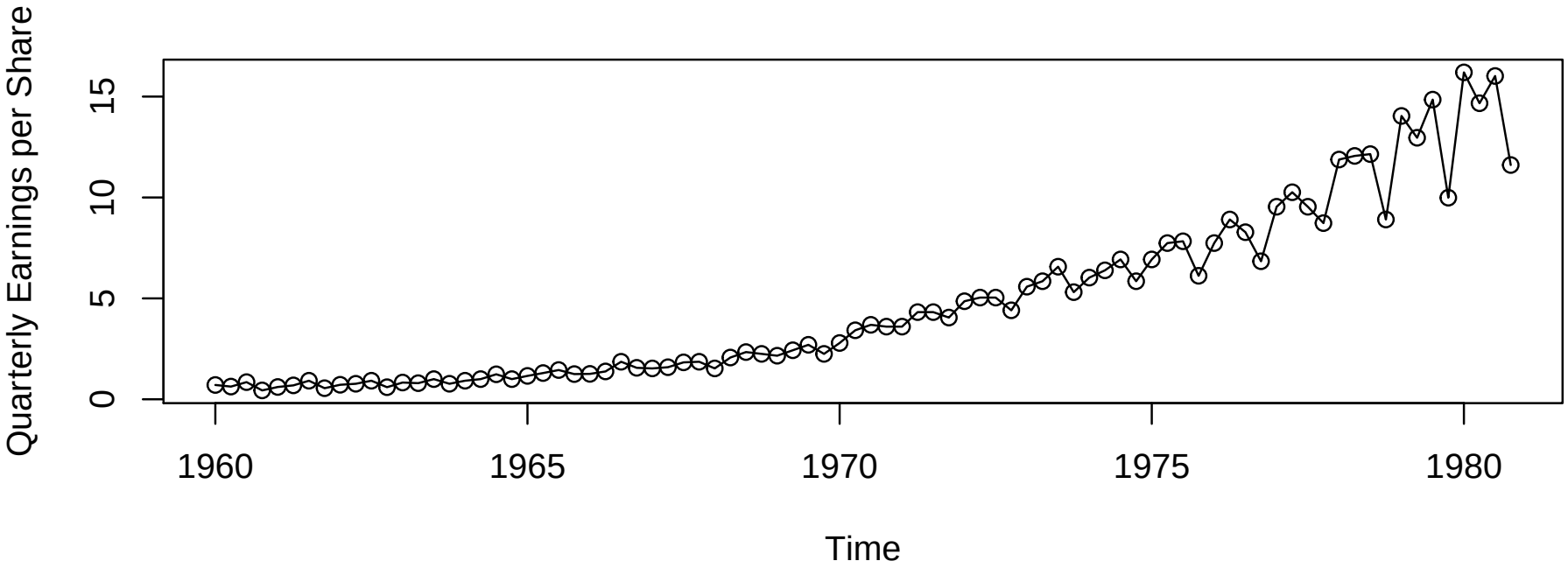
The Nature of Time Series Data

Johnson & Johnson Quarterly Earnings

Time serie data about quarterly earnings per share for the U.S. company Johnson & Johnson, furnished by Professor Paul Griffin of the Graduate School of Management, University of California, Davis. There are 84 quarters (21 years) measured from the first quarter of 1960 to the last quarter of 1980.

```
plot(jj, type = "o", ylab = "Quarterly Earnings per Share")
```

Johnson & Johnson Quarterly Earnings



Modeling

Modeling such series begins by observing the primary patterns in the time history. In this case, note the gradually increasing underlying trend and the rather regular variation superimposed on the trend that seems to repeat over quarters. Methods for analyzing data such as these are explored using regression techniques.

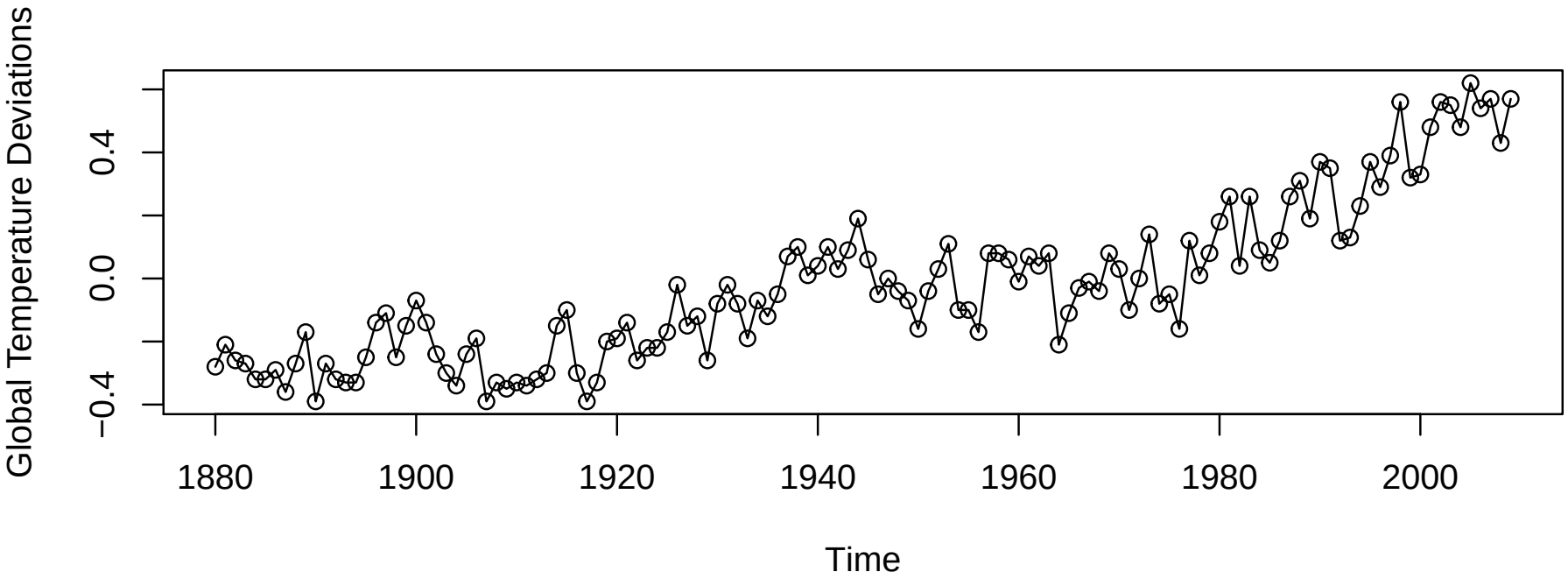
Global Warming

Global Warming

Consider the global temperature series record. The data are the global mean land-ocean temperature index from 1880 to 2009, with the base period 1951-1980. In particular, the data are deviations, measured in degrees centigrade, from the 1951-1980 average, and are an update of Hansen et al. (2006).

```
plot(gtemp, type = "o", ylab = "Global Temperature Deviations")
```

Global Warming



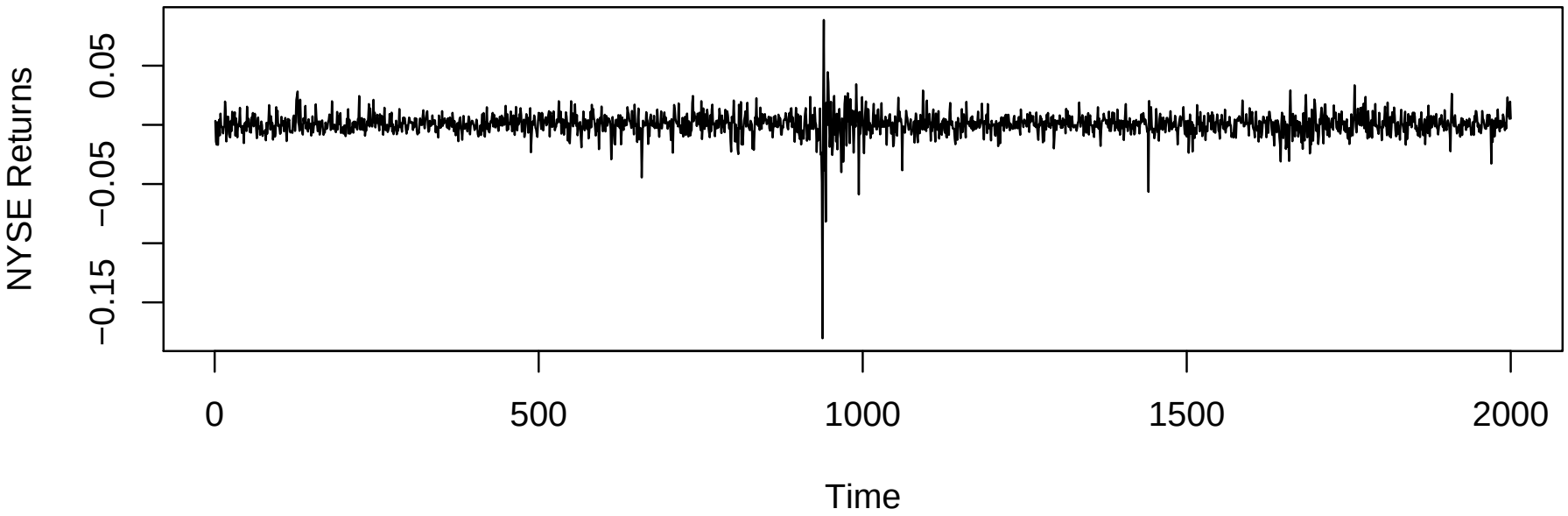
Modeling

We note an apparent upward trend in the series during the latter part of the twentieth century that has been used as an argument for the global warming hypothesis. Note also the leveling off at about 1935 and then another rather sharp upward trend at about 1970. The question of interest for global warming proponents and opponents is whether the overall trend is natural or whether it is caused by some human-induced interface.

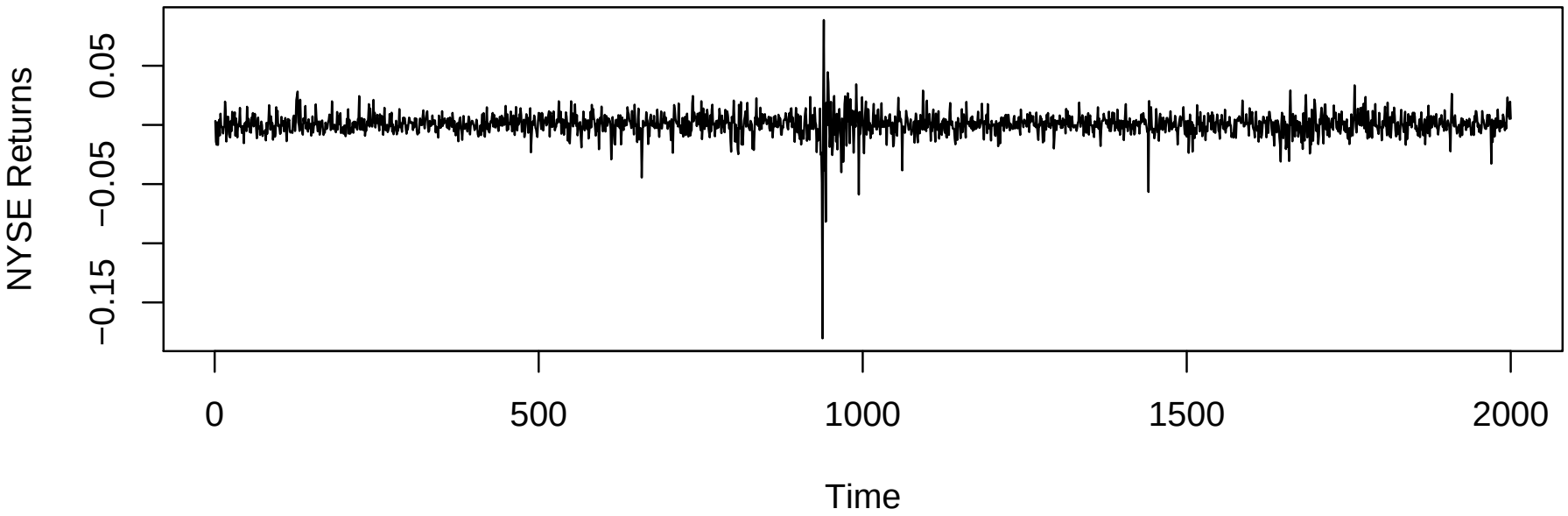
New York Stock Exchange

New York Stock Exchange

As an example of financial time series data is the daily returns (or percent change) of the New York Stock Exchange (NYSE) from February 2, 1984 to December 31, 1991. It is easy to spot the crash of October 19, 1987 in the figure produced by `plot(nyse, ylab="NYSE Returns")`



New York Stock Exchange



Modeling

The mean of the series appears to be stable with an average return of approximately zero, however, the volatility (or variability) of data changes over time. In fact, the data show volatility clustering; that is, highly volatile periods tend to be clustered together. A problem in the analysis of these type of financial data is to forecast the volatility of future returns. Models such as ARCH and GARCH models (Engle, 1982; Bollerslev, 1986) and stochastic volatility models (Harvey, Ruiz and Shephard, 1994) have been developed to handle these problems.

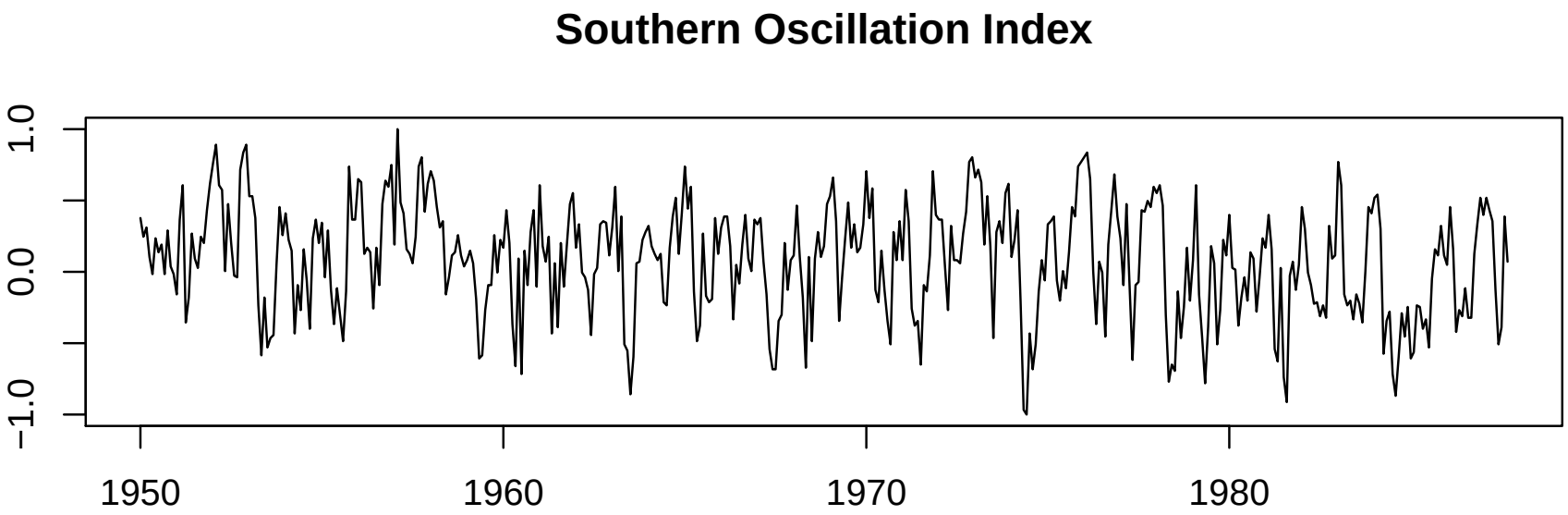
Analyzing several time series at once

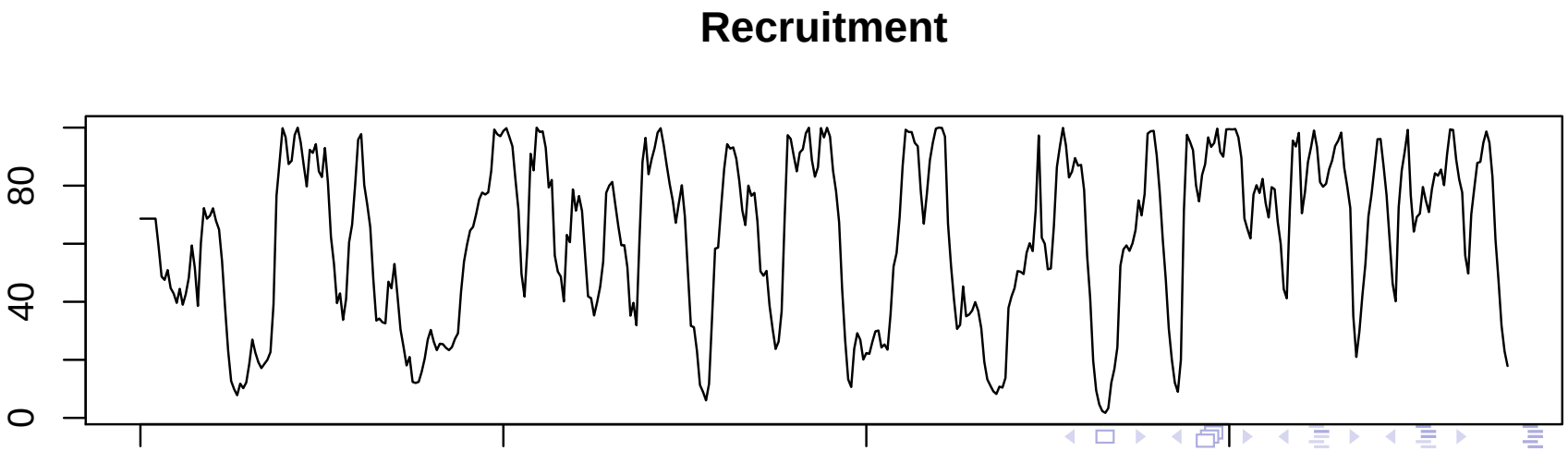
El Nino and Fish Population

Figure (see below) shows monthly values of an environmental series called the Southern Oscillation Index (SOI) and associated Recruitment (number of new fish) furnished by Dr. Roy Mendelssohn of the Pacific Environmental Fisheries Group (personal communication). Both series are for a period of 453 months ranging over the years 1950-1987. The SOI measures changes in air pressure, related to sea surface temperatures in the central Pacific Ocean. The central Pacific warms every three to seven years due to the El Nino effect, which has been blamed, in particular, for the 1997 floods in the midwestern portions of the United States.

```
par(mfrow = c(2, 1))  # set up the graphics
plot(soi, ylab = "", xlab = "", main = "Southern Oscillation Index")
plot(rec, ylab = "", xlab = "", main = "Recruitment")
```

El Nino and Fish Population





Modeling

- Both series tend to exhibit repetitive behavior, with regularly repeating cycles that are easily visible. This periodic behavior is of interest because underlying processes of interest may be regular and the rate or frequency of oscillation characterizing the behavior of the underlying series would help to identify them.
- The cycles of the SOI are repeating at a faster rate than those of the Recruitment series.
- The Recruitment series also shows several kinds of oscillations, a faster frequency that seems to repeat about every 12 months and a slower frequency that seems to repeat about every 50 months.
- The two series also tend to be somewhat related; it is easy to imagine that somehow the fish population is dependent on the SOI. Perhaps even a lagged relation exists, with the SOI signaling changes in the fish population.
- This possibility suggests trying some version of regression analysis as a procedure for relating the two series.

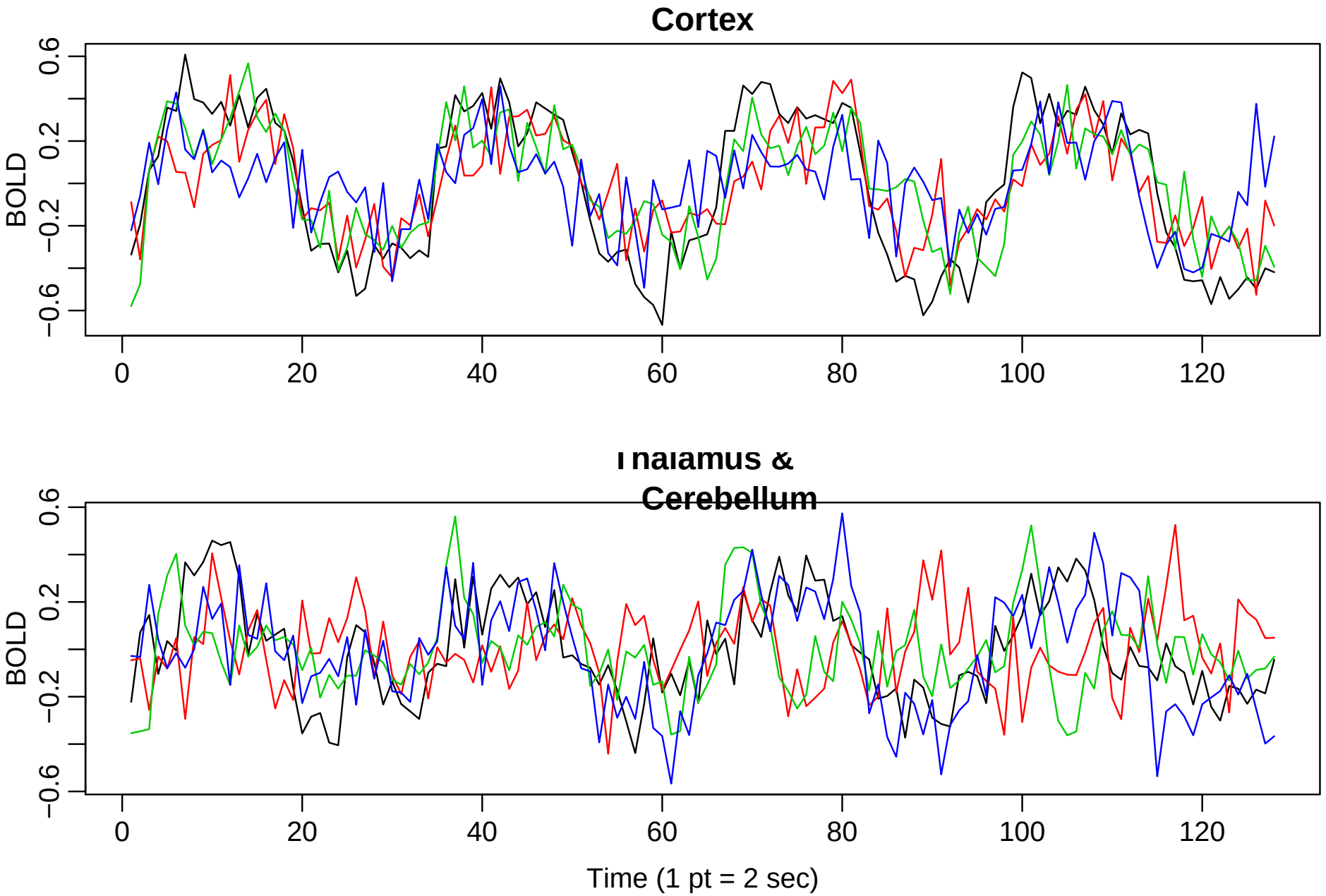
Analyzing several time series at once

fMRI Imaging

We observe data collected from various locations in the brain via functional magnetic resonance imaging (fMRI). In this example, five subjects were given periodic brushing on the hand. The stimulus was applied for 32 seconds and then stopped for 32 seconds; thus, the signal period is 64 seconds. The sampling rate was one observation every 2 seconds for 256 seconds (n = 128). For this example, we averaged the results over subjects (these were evoked responses, and all subjects were in phase). The data are consecutive measures of blood oxygenation- level dependent (bold) signal intensity, which measures areas of activation in the brain.

```
par(mfrow = c(2, 1), mar = c(3, 2, 1, 0) + 0.5, mgp = c(1.6,
    0.6, 0))
ts.plot(fmri1[, 2:5], col = 1:4, ylab = "BOLD", xlab = "", main = "Cortex")
ts.plot(fmri1[, 6:9], col = 1:4, ylab = "BOLD", xlab = "", main = "Thalamus &\n
mtext("Time (1 pt = 2 sec)", side = 1, line = 2)
```

fMRI Imaging



Modeling

- The periodicities appear strongly in the motor cortex series and less strongly in the thalamus and cerebellum.
- The fact that one has series from different areas of the brain suggests testing whether the areas are responding differently to the brush stimulus.
- Analysis of variance techniques accomplish this in classical statistics
- These classical techniques extend to the time series case, leading to a spectral analysis of variance.

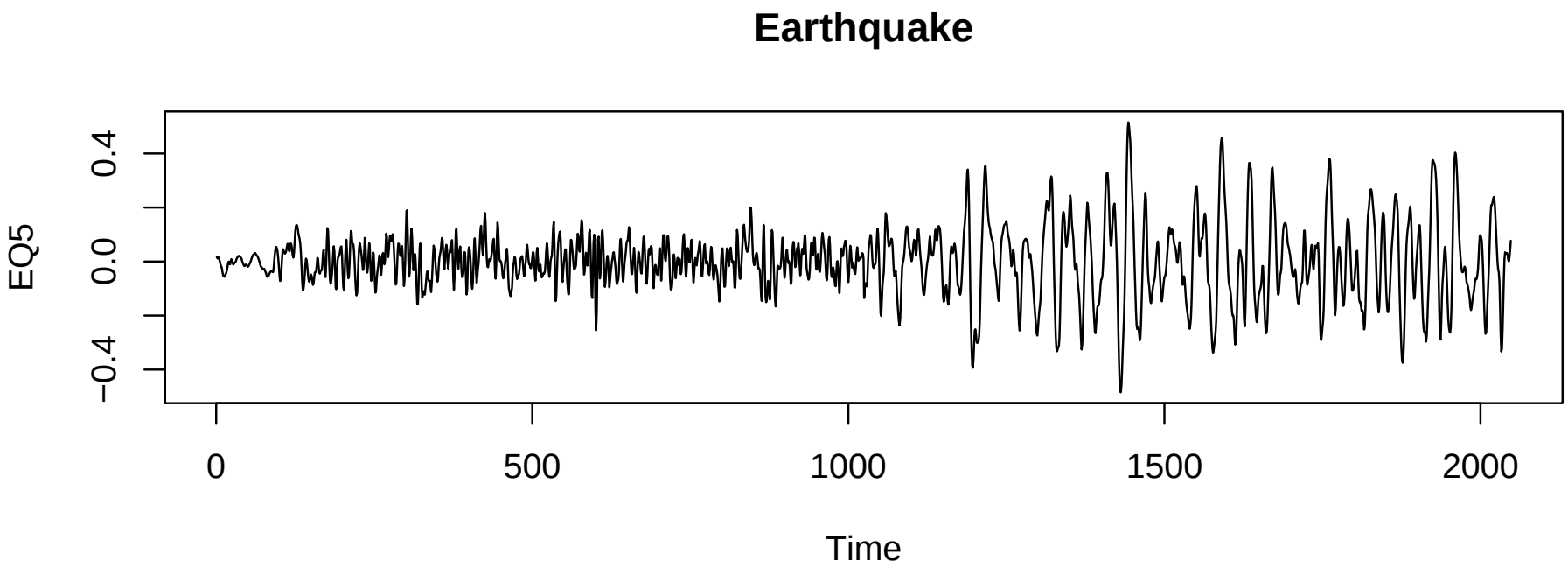
Analyzing several time series at once

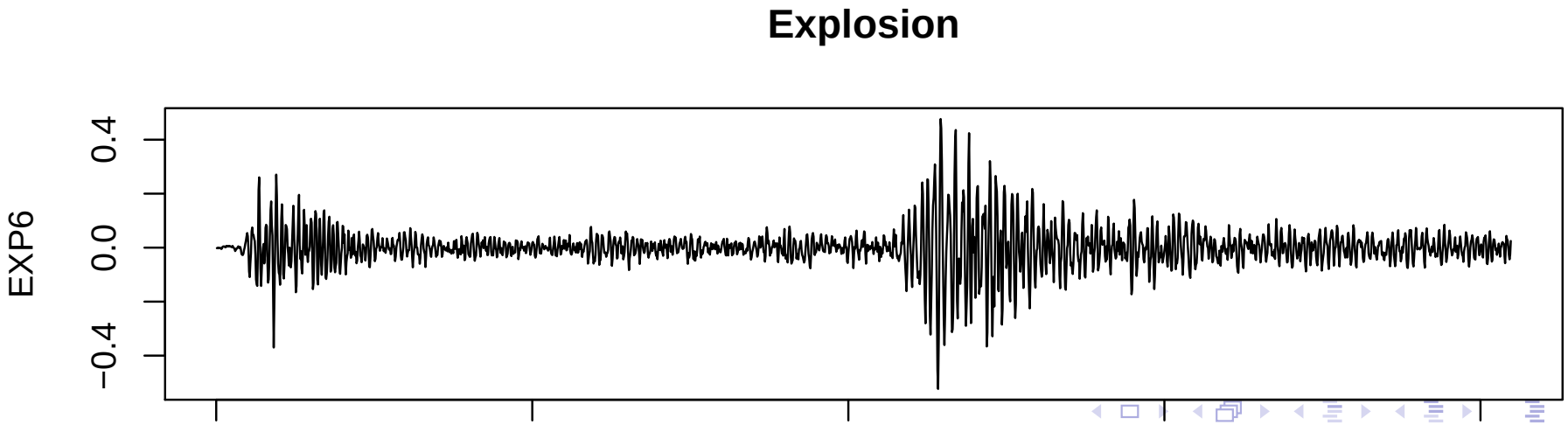
Earthquakes and Explosions

The series in Figure (see below) represent two phases or arrivals along the surface, denoted by P $(t = 1, \ldots, 1024)$ and S $(t = 1025, \ldots, 2048)$, at a seismic recording station. The recording instruments in Scandinavia are observing earthquakes and mining explosions

```
par(mfrow = c(2, 1))
plot(EQ5, main = "Earthquake")
plot(EXP6, main = "Explosion")
```

Earthquakes and Explosions





Modeling

- The general problem of interest is in distinguishing or discriminating between waveforms generated by earthquakes and those generated by explosions.
- In the case of the two events, the ratio of maximum amplitudes appears to be somewhat less than .5 for the earthquake and about 1 for the explosion.
- We can again think about spectral analysis of variance for testing the equality of the periodic components of earthquakes and explosions.
- We would also like to be able to classify future P and S components from events of unknown origin, leading to the time series discriminant analysis.

Basic Objectives of the Analysis

The basic objective usually is to determine a model that describes the pattern of the time series. Uses for such a model are:

- To describe the important features of the time series pattern.
- To explain how the past affects the future or how two time series can *interact*.
- To forecast future values of the series.

Some simple forecasting methods

- Average method: the forecasts of all future values are equal to the mean of the historical data.

- Naive method: All forecasts are simply set to be the value of the last observation.
- Seasonal naive method: we set each forecast to be equal to the last observed value from the same season of the year.
- Drift method: A variation on the naive method is to allow the forecasts to increase or decrease over time, where the amount of change over time (called the drift) is set to be the average change seen in the historical data. So the forecast for time T+h is given by

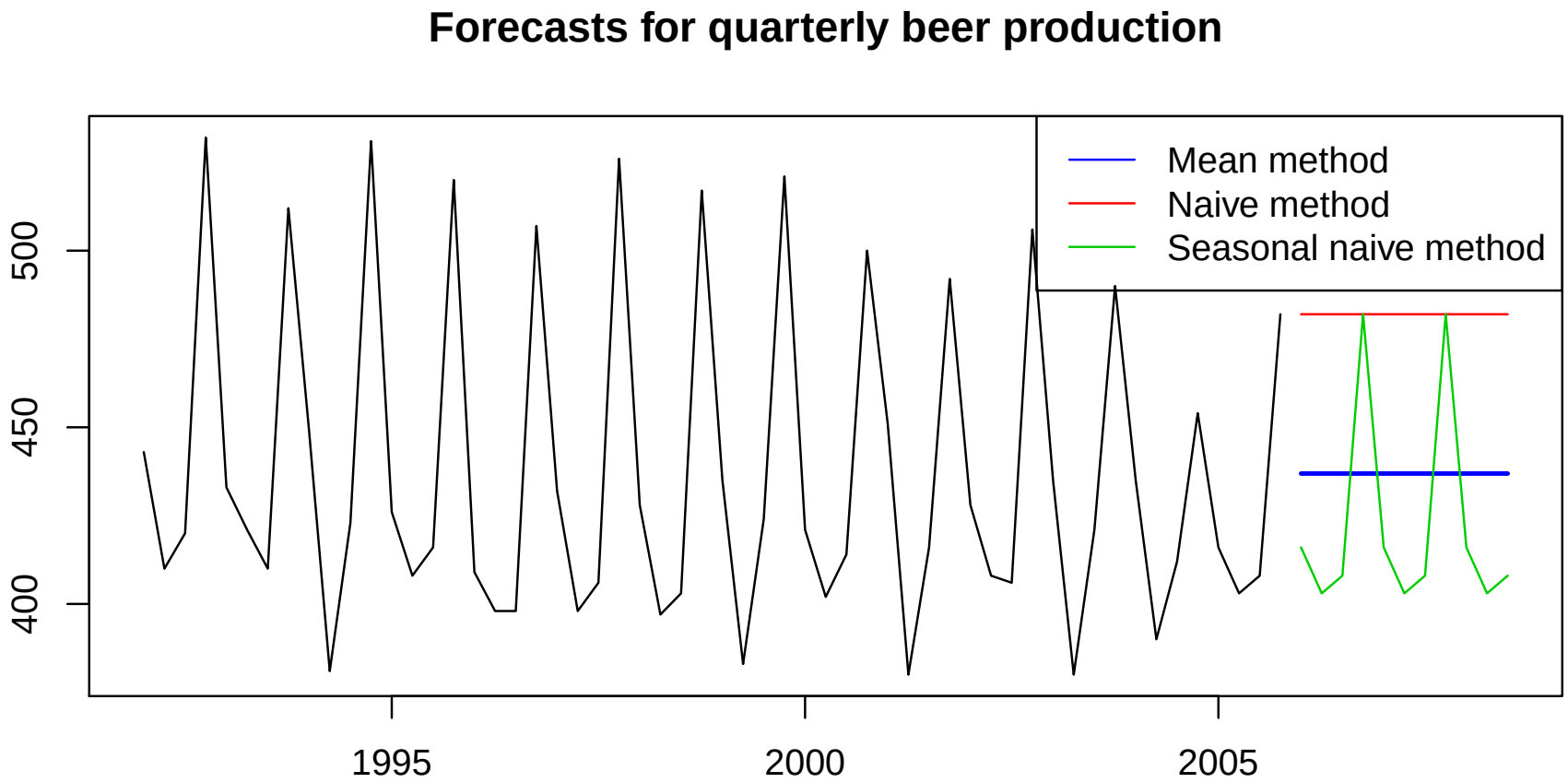
$$y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}).$$

This is equivalent to drawing a line between the first and last observation, and extrapolating it into the future.

Example: quarterly beer production

```
beer2 <- window(ausbeer, start = 1992, end = 2006 - 0.1)
beerfit1 <- meanf(beer2, h = 11)
beerfit2 <- naive(beer2, h = 11)
beerfit3 <- snaive(beer2, h = 11)
plot(beerfit1, plot.conf = FALSE, main = "Forecasts for quarterly beer production")
lines(beerfit2$mean, col = 2)
lines(beerfit3$mean, col = 3)
legend("topright", lty = 1, col = c(4, 2, 3), legend = c("Mean method",
    "Naive method", "Seasonal naive method"))
```

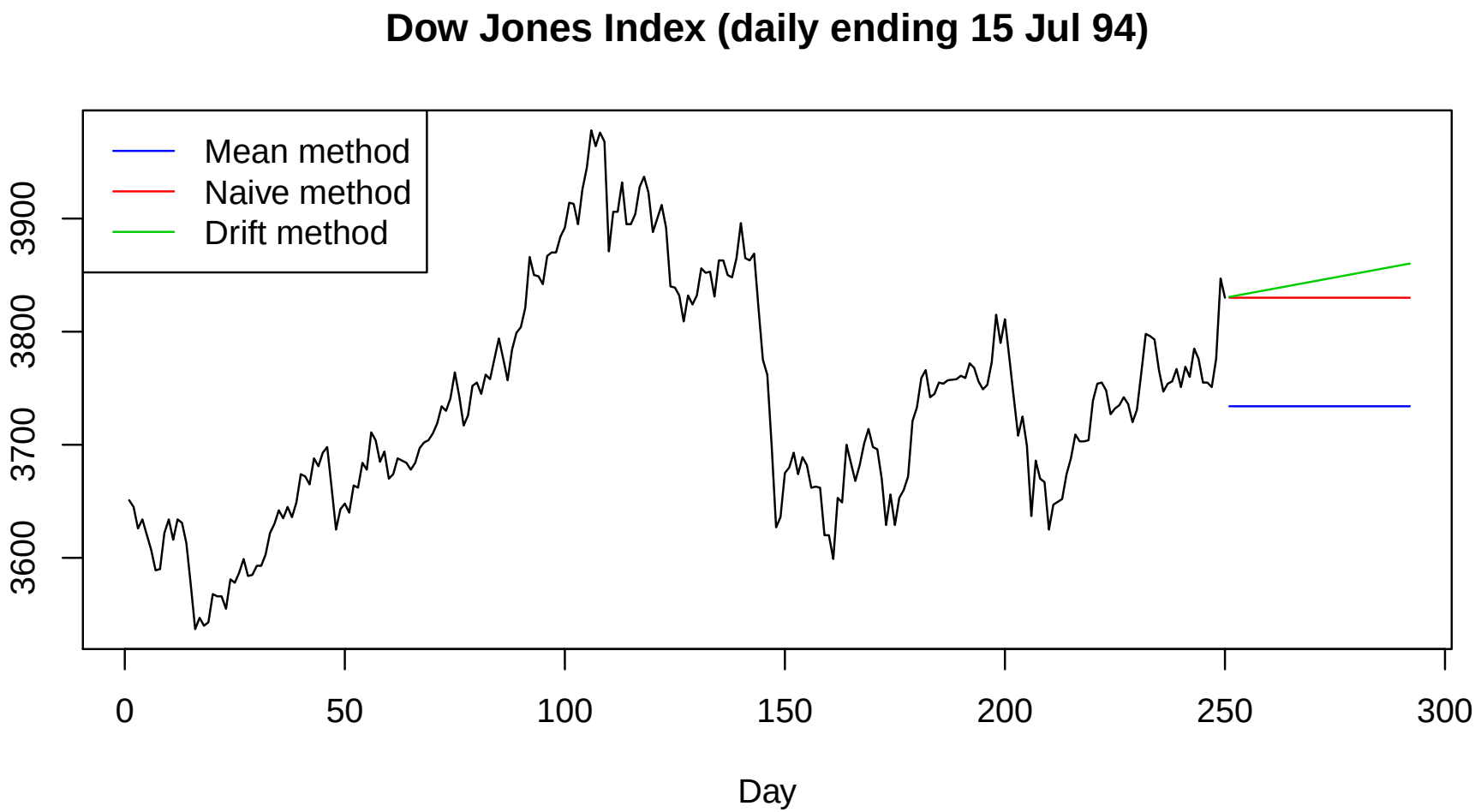
Example: quarterly beer production



Example: Dow Jones Index

```
dj2 <- window(dj, end = 250)
plot(dj2, main = "Dow Jones Index (daily ending 15 Jul 94)",
    ylab = "", xlab = "Day", xlim = c(2, 290))
lines(meanf(dj2, h = 42)$mean, col = 4)
lines(rwf(dj2, h = 42)$mean, col = 2)
lines(rwf(dj2, drift = TRUE, h = 42)$mean, col = 3)
legend("topleft", lty = 1, col = c(4, 2, 3), legend = c("Mean method",
    "Naive method", "Drift method"))
```

Example: Dow Jones Index



Types of Models

There are two basic types of time domain models.

- Models that relate the present value of a series to past values and past prediction errors - these are called ARIMA models (for Autoregressive Integrated Moving Average).
- Ordinary regression models that use time indices as x-variables. These can be helpful for an initial description of the data and form the basis of several simple forecasting methods.

Important Characteristics to Consider First

Some important questions to first consider when first looking at a time series are:

- Is there a trend, meaning that, on average, the measurements tend to increase (or decrease) over time?
- Is there seasonality, meaning that there is a regularly repeating pattern of highs and lows related to calendar time such as seasons, quarters, months, days of the week, and so on?
- Are their outliers? In regression, outliers are far away from your line. With time series data, your outliers are far away from your other data.
- Is there a long-run cycle or period unrelated to seasonality factors?
- Is there constant variance over time, or is the variance non-constant?
- Are there any abrupt changes to either the level of the series or the variance?

ARIMA Model

A brief introduction of ARIMA model:

- Autocorrelation
  - Definition
  - White noise
- Autoregressive model
  - Definition
  - AR(1)
  - Example
- Moving average model
  - Definition
  - MA(1)
  - Partial autocorrelation
- ARIMA model

Autocorrelation

Autocorrelation

Autocorrelation measures the linear relationship between lagged values of a time series. There are several autocorrelation coefficients, depending on the lag length.

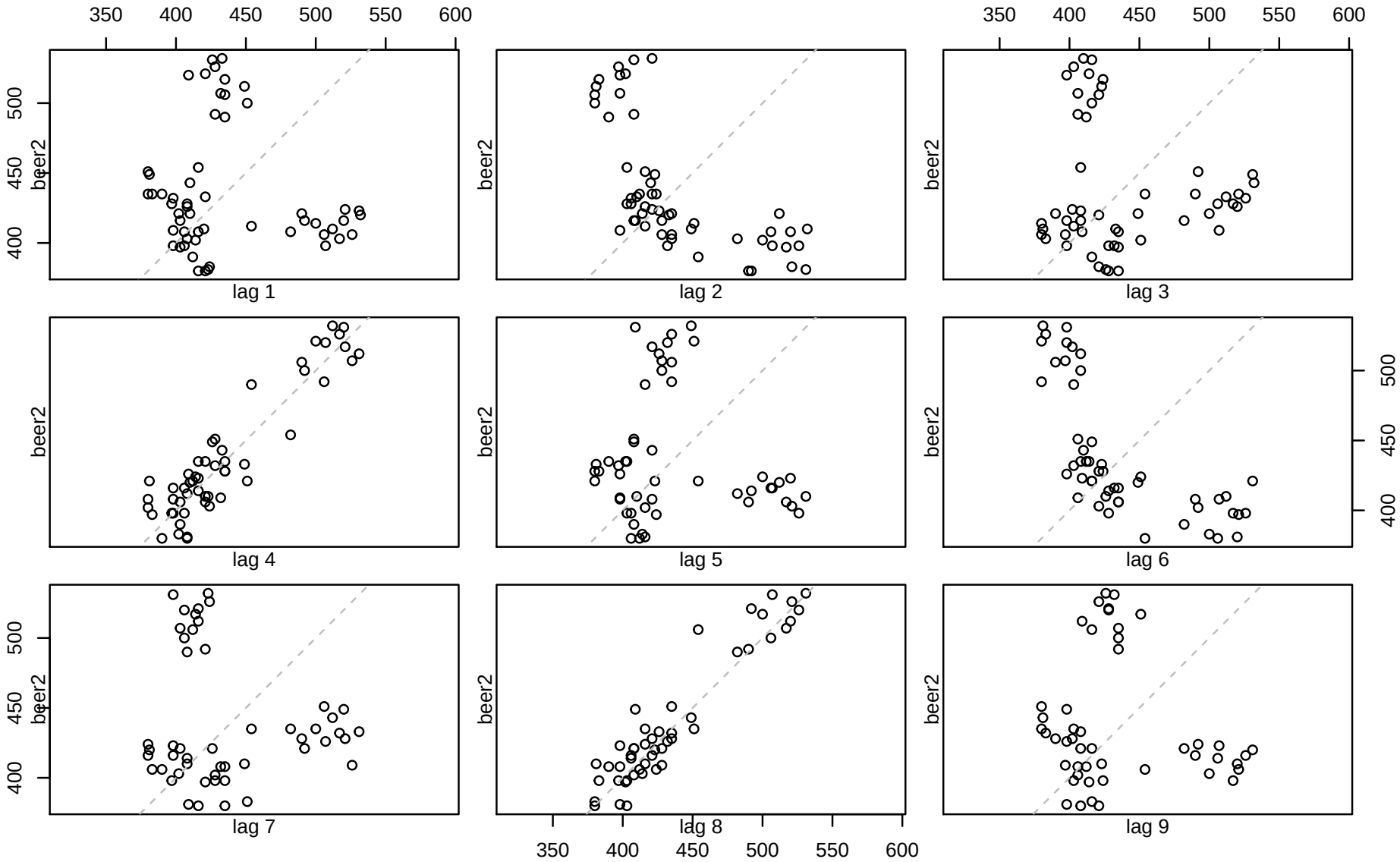
For example, r_1 measures the relationship between y_t and y_{t-1}, r_2 measures the relationship between y_t and y_{t-2}, and so on.

```
beer2 <- window(ausbeer, start = 1992, end = 2006 - 0.1)
lag.plot(beer2, lags = 9, do.lines = FALSE)
```

This code enables us to display scatterplots of the beer production time series where the horizontal axis shows lagged values of the time series.

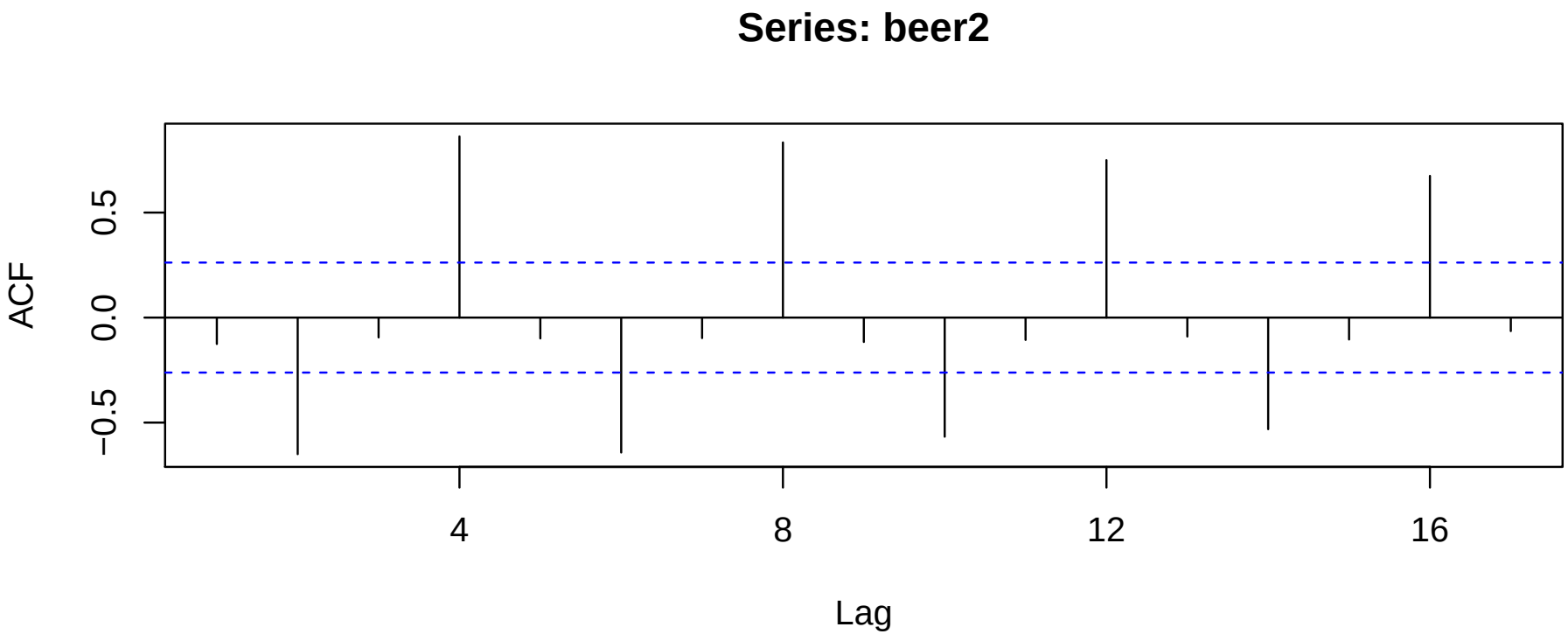
- Each graph shows y_t plotted against y_{t-k} for different values of k.
- The autocorrelations are the correlations associated with these scatterplots.

Autocorrelation



The autocorrelation coefficients are normally plotted to form the autocorrelation function or ACF. The plot is also known as a correlogram.

```
Acf(beer2)
```

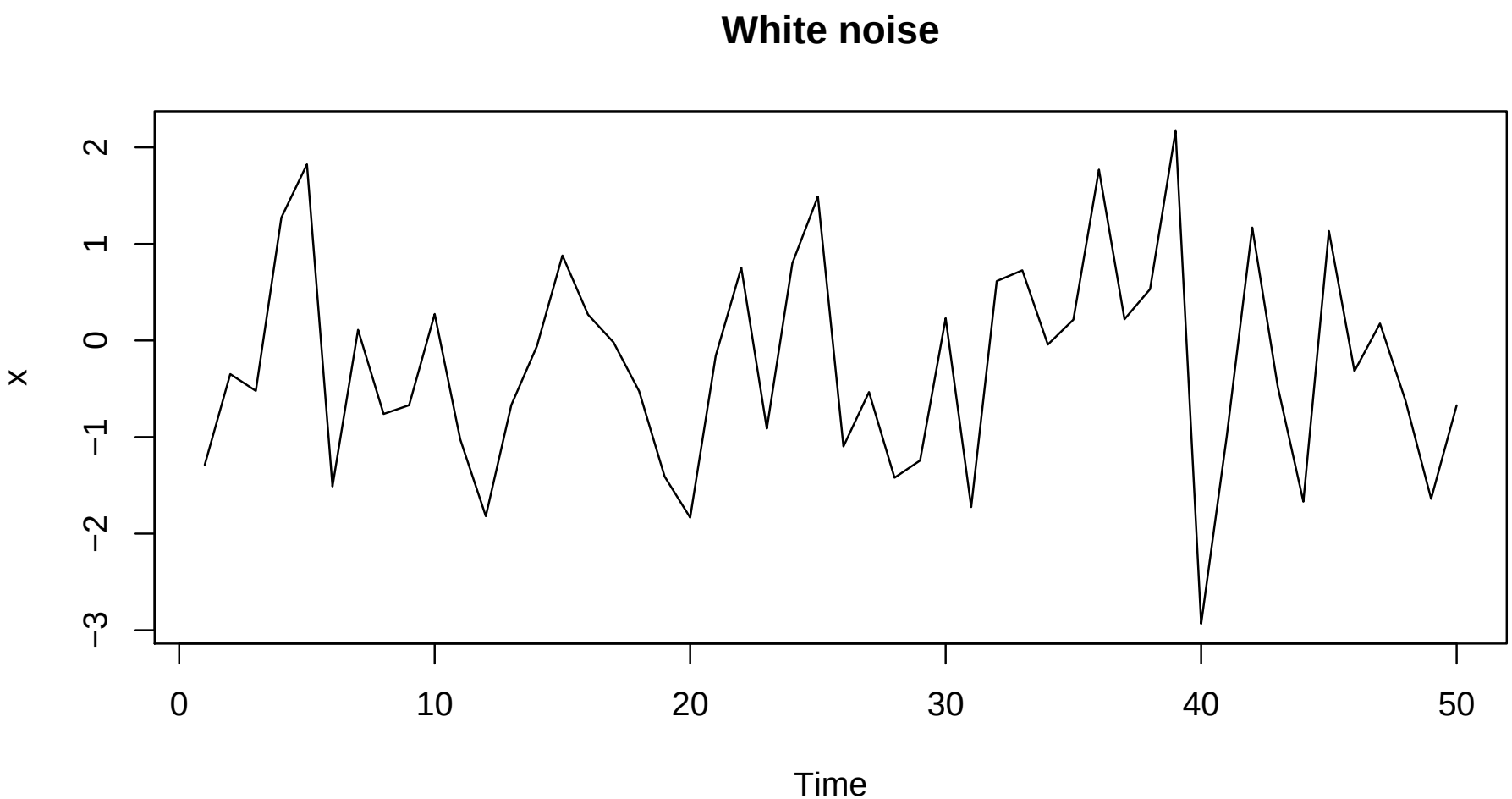


- r_4 is higher than for the other lags. This is due to the seasonal pattern in the data: the peaks tend to be four quarters apart and the troughs tend to be two quarters apart.
- r_2 is more negative than for the other lags because troughs tend to be two quarters behind peaks.

White noise

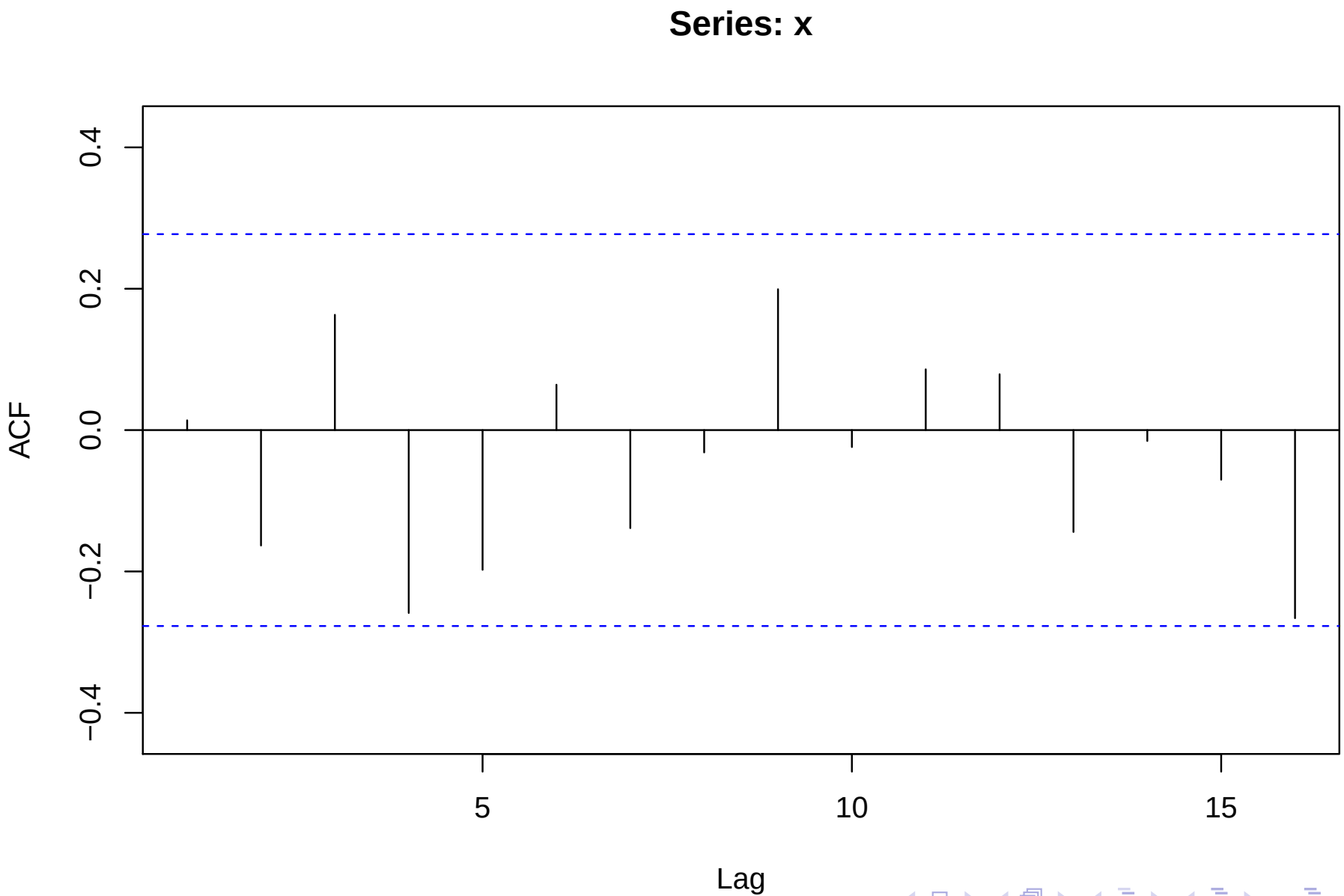
Time series that show no autocorrelation are called white noise.

```
set.seed(30)
x <- ts(rnorm(50))
plot(x, main = "White noise")
```



White noise

```
Acf(x)
```



Example 1: ARIMA model

AR(1):autoregressive model of order 1

One of the simplest ARIMA type models is a model in which we use a linear model to predict the value at the present time using the value at the previous time. This is called an AR(1) model, standing for autoregressive model of order 1. The order of the model indicates how many previous times we use to predict the present time.

Example 1: ARIMA model

AR(1):autoregressive model of order 1

One of the simplest ARIMA type models is a model in which we use a linear model to predict the value at the present time using the value at the previous time. This is called an AR(1) model, standing for autoregressive model of order 1. The order of the model indicates how many previous times we use to predict the present time.

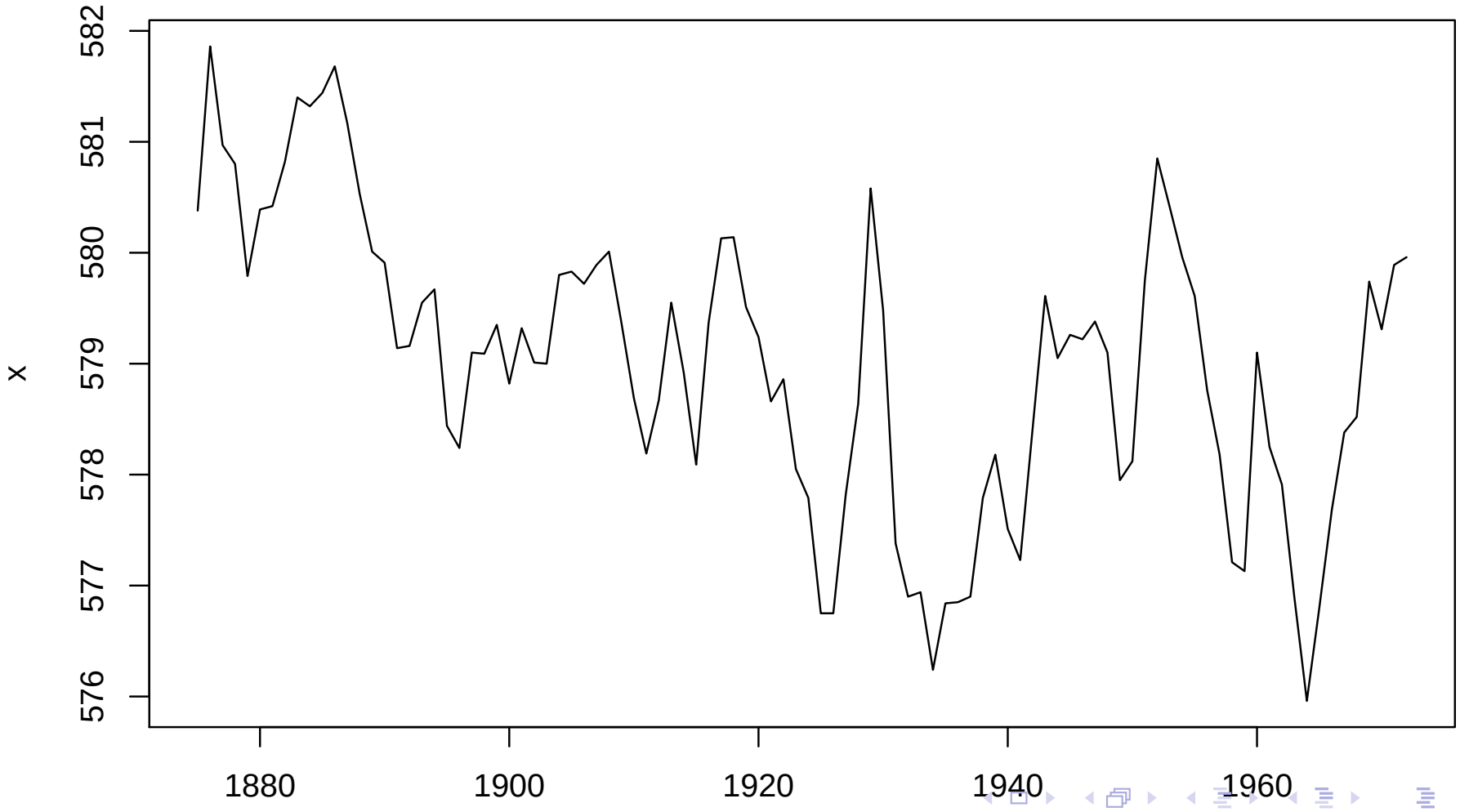
The AR(1) model

$$y_t = \delta + \phi y_{t-1} + w_t,$$

where $w_t \sim N(0, \sigma_w^2)$ are i.i.d., meaning that the errors are independently distributed with a normal distribution that has mean 0 and constant variance. Moreover, the errors w_t are independent of y_t.

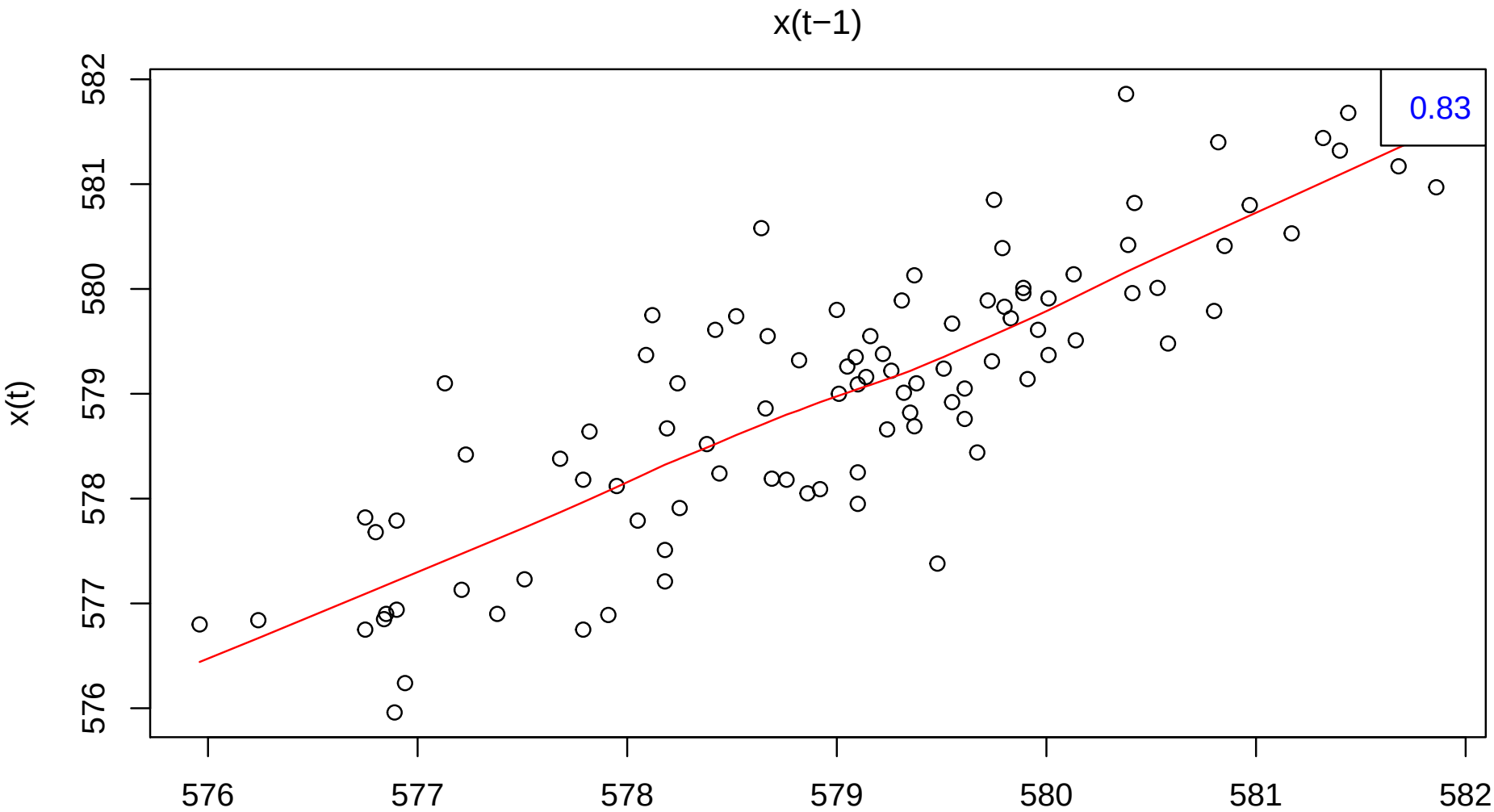
Example: LakeHuron dataset

```
data(LakeHuron)
x <- LakeHuron
plot(x)
```



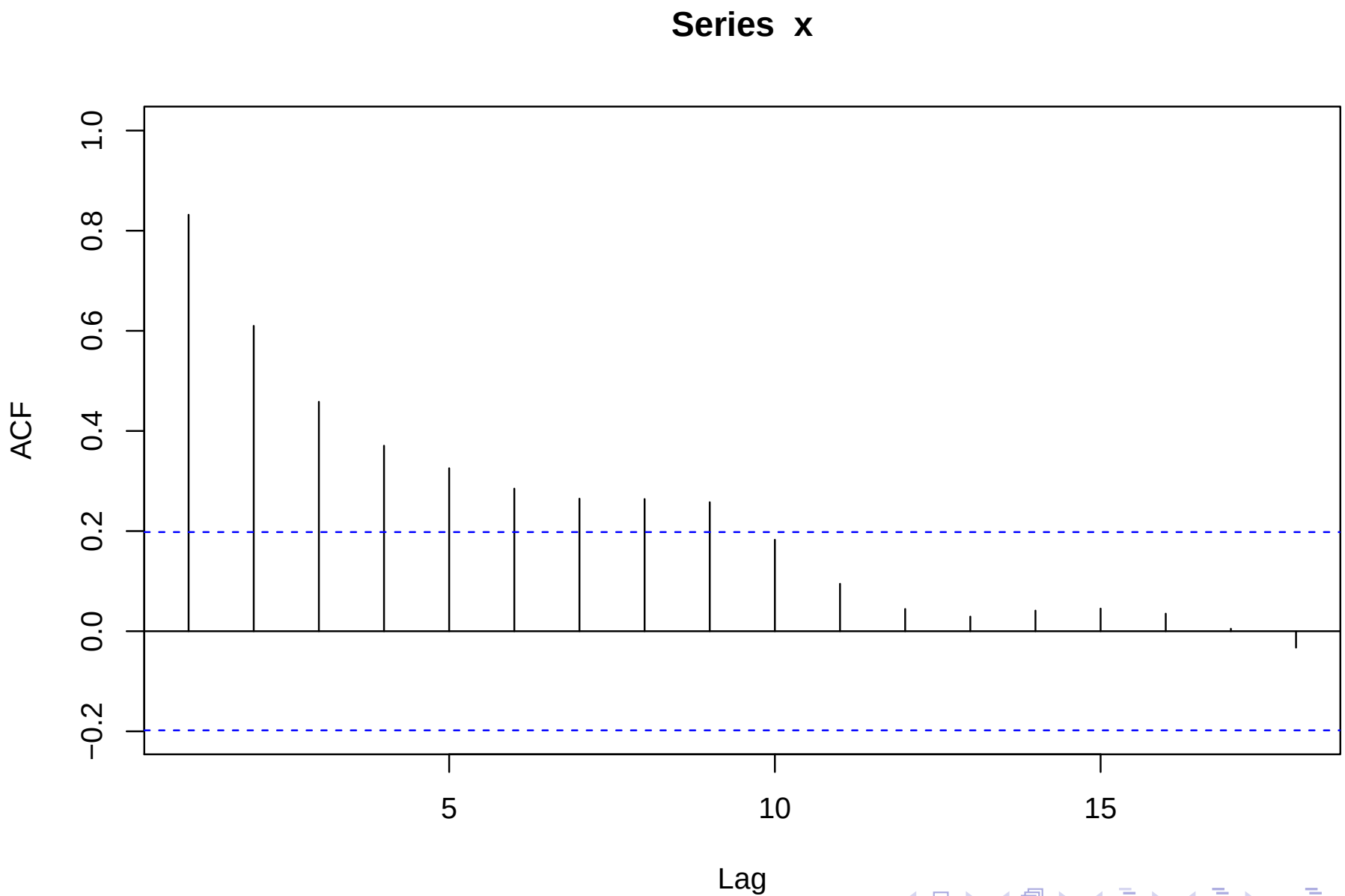
plot of y_t versus y_{t-1}:

```
lag1.plot(x, 1)
```



Autocorrelation Function

```
acf(x, xlim = c(1, 18))
```



Estimation of our AR(1)

$$y_t = \delta + \phi y_{t-1} + w_t,$$

This is essentially the ordinary simple linear regression equation, but there is one difference. Although it's not usually true, in ordinary least squares regression we assume that the x-variable is not random but instead is something we can control.

That's not the case here, but in our first encounter with time series we'll overlook that and use ordinary regression methods. We'll do things the "right" way later in the course.

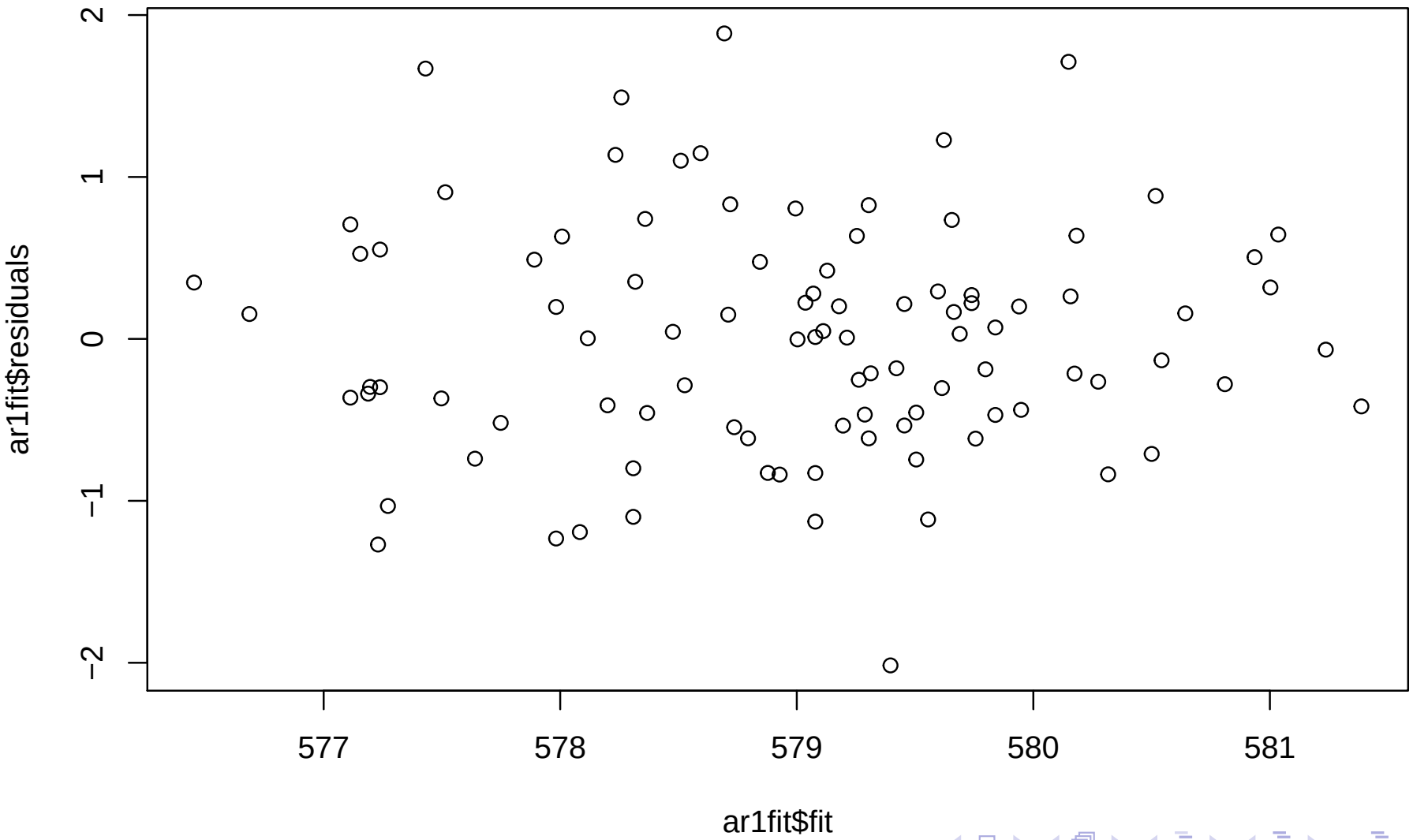
```
xlag1 <- lag(x, -1)  # Creates a lag 1 of x variable. See note 2
y <- cbind(x, xlag1)  # See note 3 below
ar1fit <- lm(y[, 1] ~ y[, 2])
```

Results of our AR(1)

```
summary(ar1fit)


Call:
lm(formula = y[, 1] ~ y[, 2])

Residuals:
     Min       1Q   Median       3Q      Max
-2.01620 -0.46747  0.00781  0.47583  1.88638

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 94.71257   32.23790   2.938  0.00415 **
y[, 2]       0.83641    0.05568  15.022  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7209 on 95 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.7037,Adjusted R-squared:  0.7006
F-statistic: 225.7 on 1 and 95 DF,  p-value: < 2.2e-16
```

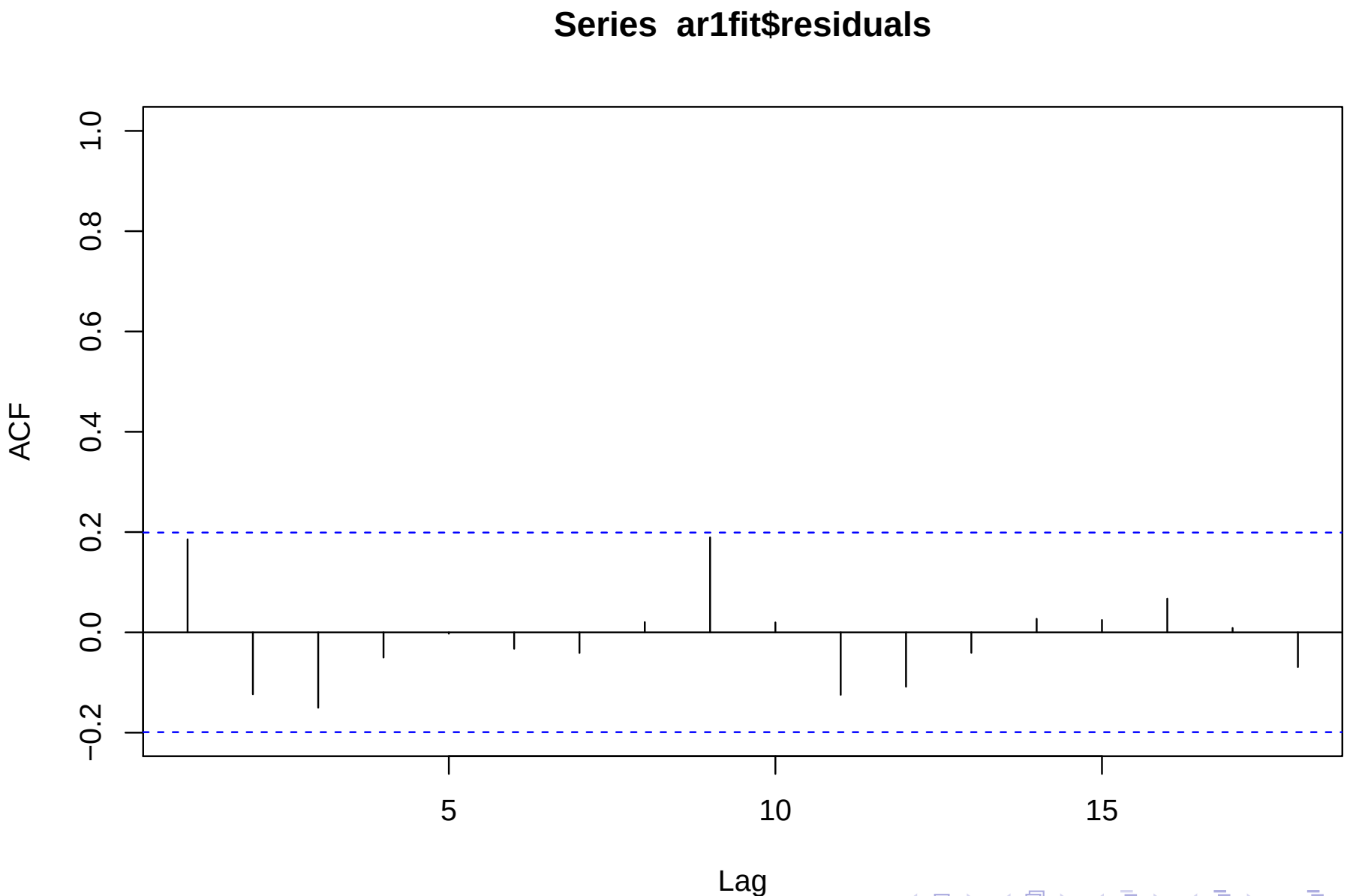
Residuals analysis

```
plot(ar1fit$fit, ar1fit$residuals)  #plot of residuals versus fits
```



Residuals analysis: ACF

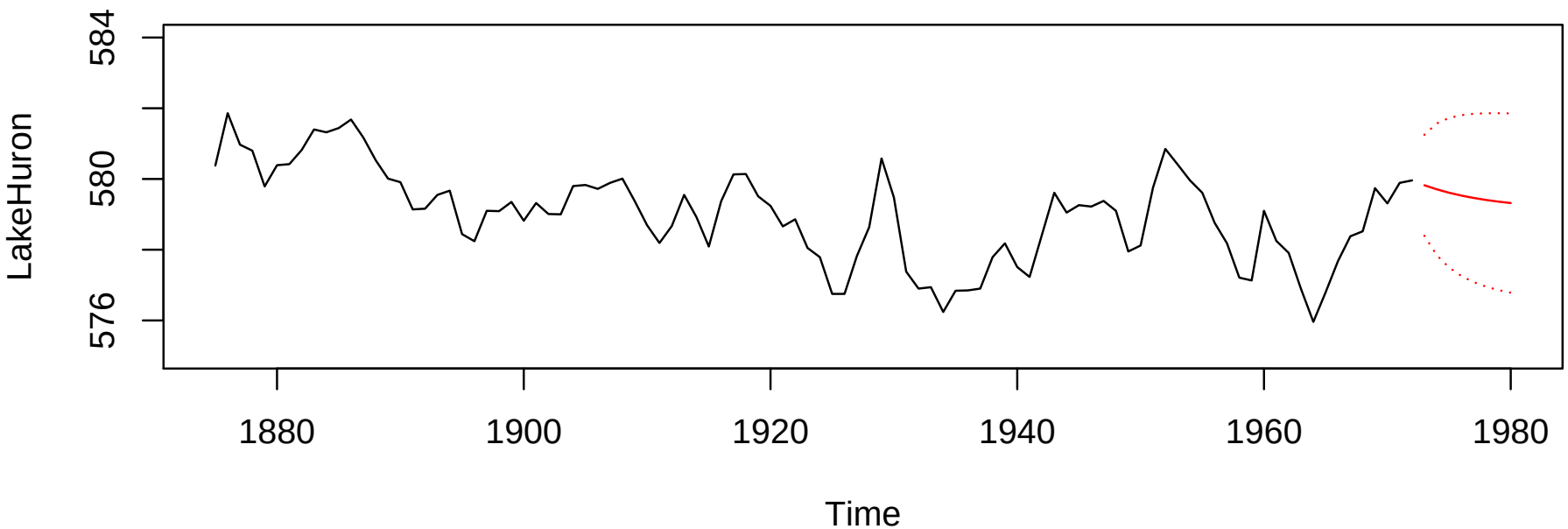
```
acf(ar1fit$residuals, xlim = c(1, 18))
```



Prediction for an ARIMA model

We will see it latter by using:

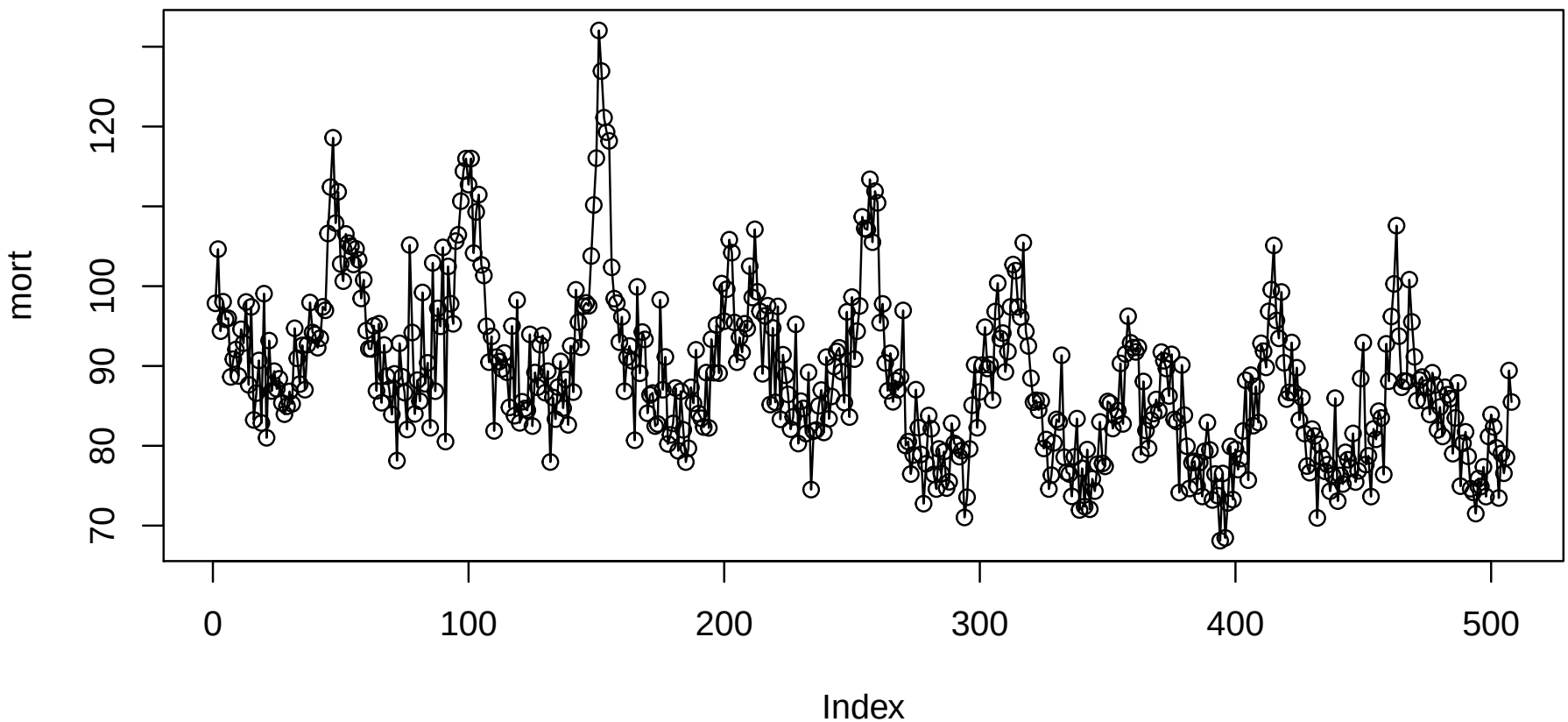
```
fit <- arima(LakeHuron, order = c(1, 0, 0))
LH.pred <- predict(fit, n.ahead = 8)
plot(LakeHuron, xlim = c(1875, 1980), ylim = c(575, 584))
LH.pred <- predict(fit, n.ahead = 8)
lines(LH.pred$pred, col = "red")
lines(LH.pred$pred + 2 * LH.pred$se, col = "red", lty = 3)
lines(LH.pred$pred - 2 * LH.pred$se, col = "red", lty = 3)
```



AR(1): Mortality example

Here's a time series of the daily cardiovascular mortality rate in Los Angeles County, 1970-1979.

```
mort <- scan("/Users/liquetwe/Dropbox/ANGLET/TEACHING/M2/Serie-CHRONO/ARIMA-lecture/C
# data: CardioVascMort.tsm
plot(mort, type = "o")  # plot of mortality rate
```



First differences

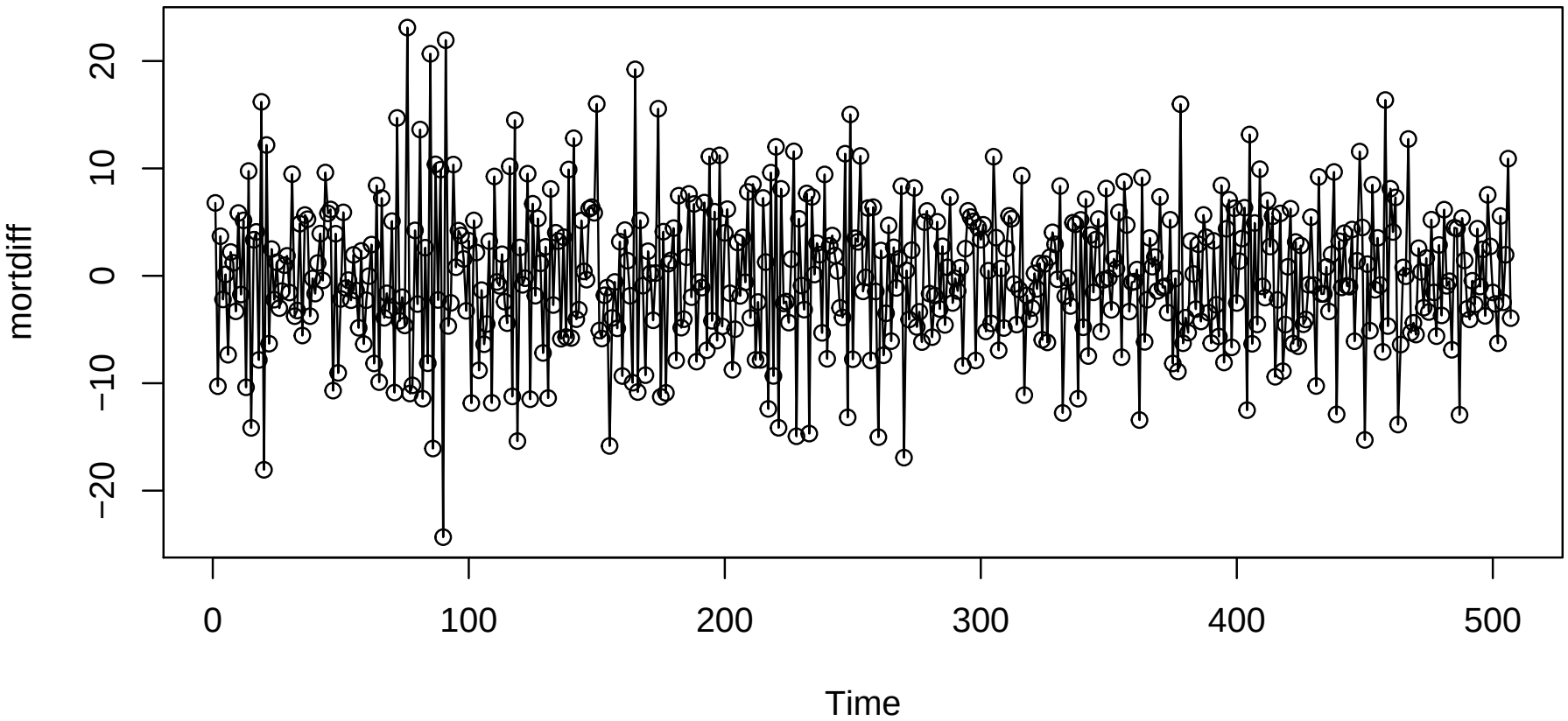
There is a slight downward trend, so the series may not be stationary. To create a (possibly) stationary series, we'll examine the first differences $y_t = x_t - x_{t-1}$. This is a common time series method for creating a de-trended series and thus potentially a stationary series.

stationary series

A stationary time series is one whose properties do not depend on the time at which the series is observed. So time series with trends, or with seasonality, are not stationary. The trend and seasonality will affect the value of the time series at different times. On the other hand, a white noise series is stationary. It does not matter when you observe it, it should look much the same at any period of time.

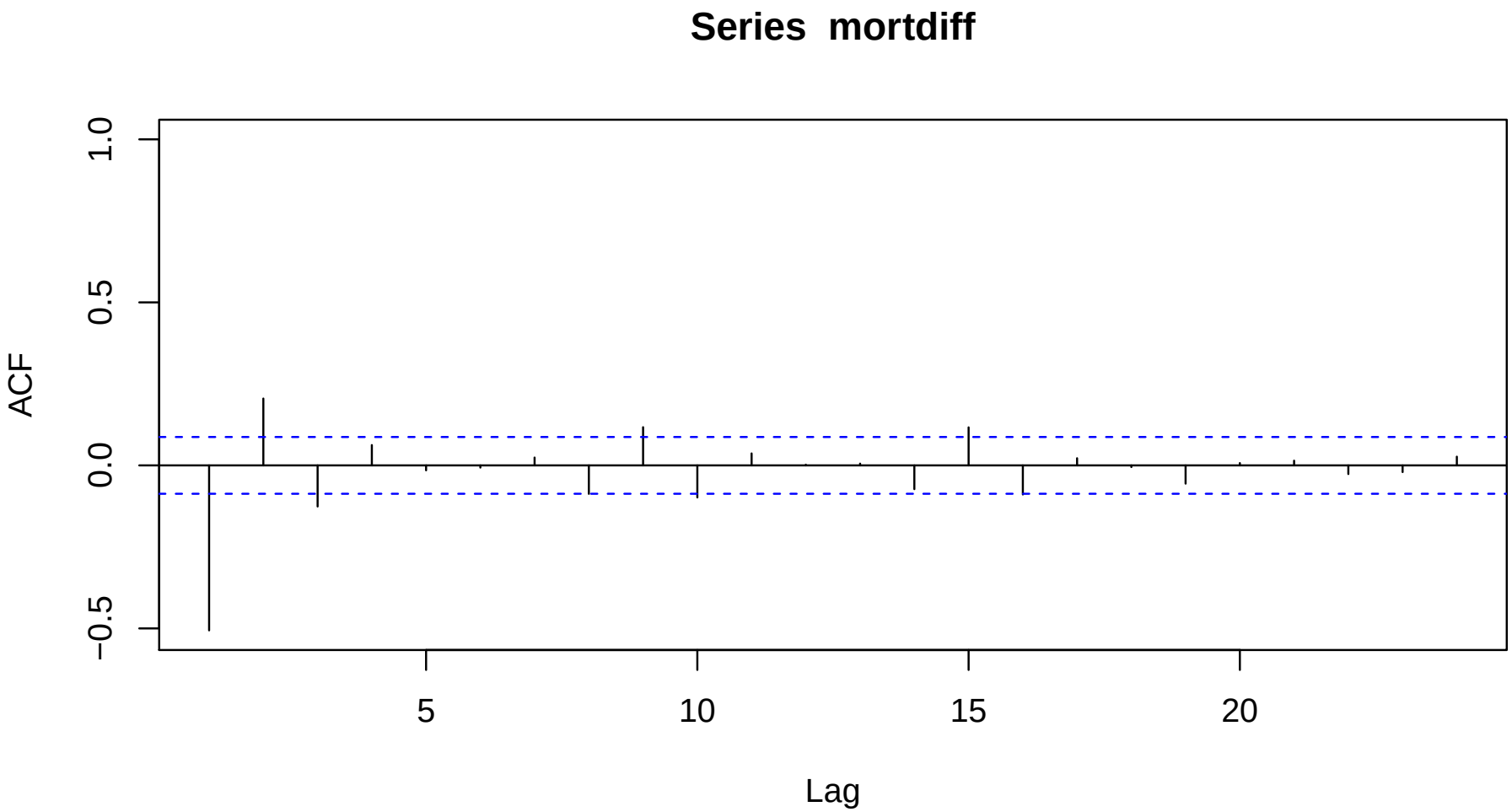
First differences

```
mortdiff <- diff(mort, 1)  # creates a variable = x(t) âĂŞ x(t-1)
mortdiff <- ts(mortdiff)
plot(mortdiff, type = "o")  # plot of first differences
```



ACF plot

```
acf(mortdiff, xlim = c(1, 24))
```



This looks like the pattern of an AR(1) with a negative lag 1 autocorrelation.

AR(1): Model

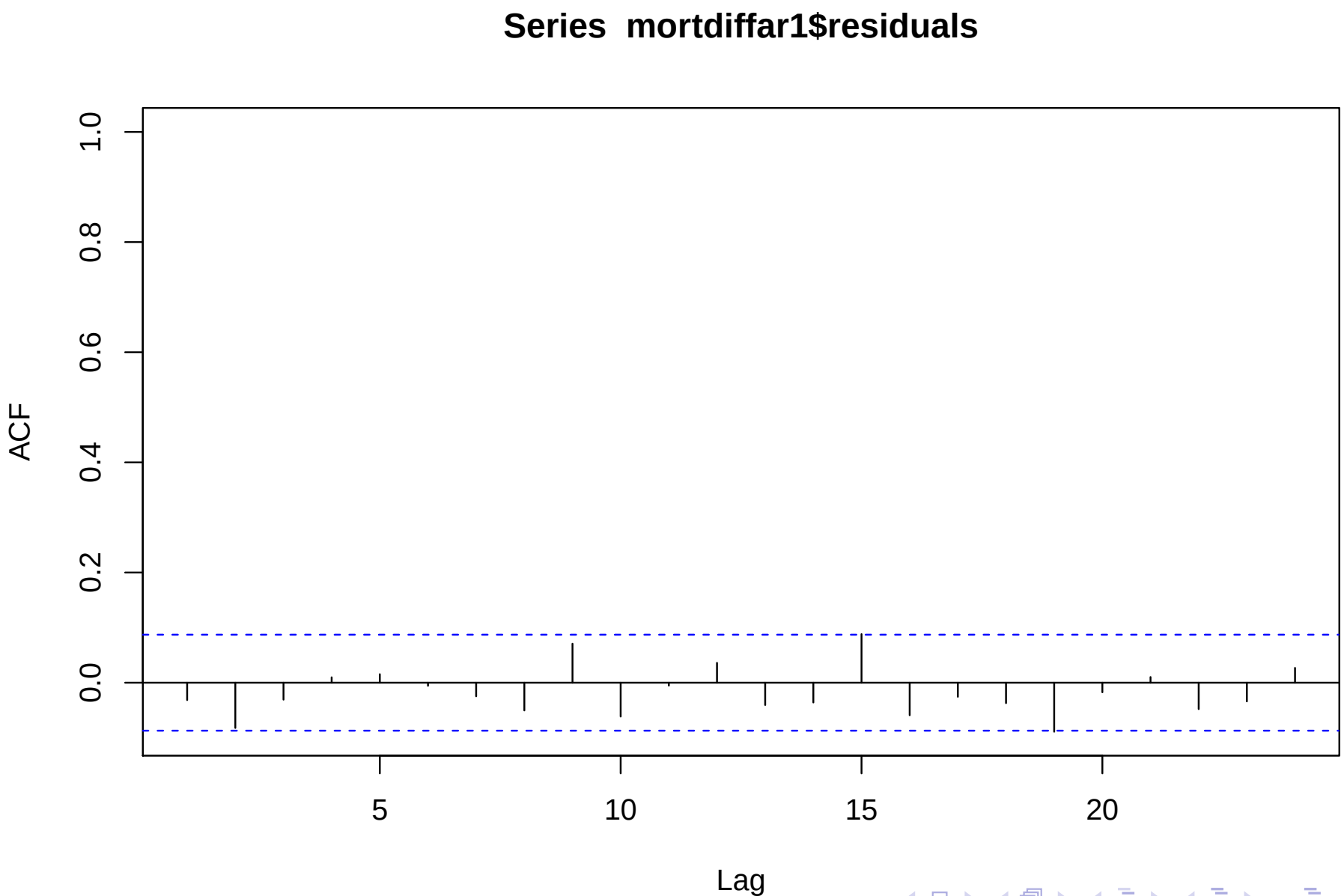
```
mortdifflag1 <- lag(mortdiff, -1)
yy <- cbind(mortdiff, mortdifflag1)
head(yy)

     mortdiff mortdifflag1
[1,]     6.79           NA
[2,]   -10.28         6.79
[3,]     3.69       -10.28
[4,]    -2.20         3.69
[5,]     0.13        -2.20
[6,]    -7.35         0.13

mortdiffar1 <- lm(yy[-c(1, 508), 1] ~ yy[-c(1, 508), 2])
# AR(1) regression for first differences
```

AR(1): Estimated Model

```
summary(mortdiffar1)


Call:
lm(formula = yy[-c(1, 508), 1] ~ yy[-c(1, 508), 2])

Residuals:
     Min       1Q   Median       3Q      Max
-19.2758  -3.8753  -0.0953   3.5725  20.8169

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       -0.04627    0.25900  -0.179    0.858
yy[-c(1, 508), 2] -0.50636    0.03838 -13.195   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.826 on 504 degrees of freedom
Multiple R-squared:  0.2568,Adjusted R-squared:  0.2553
F-statistic: 174.1 on 1 and 504 DF,  p-value: < 2.2e-16
```

AR(1): Residuals

```
acf(mortdiffar1$residuals, xlim = c(1, 24))  # ACF of residuals for 24 lags.
```



Moving average models

Rather than use past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q},$$

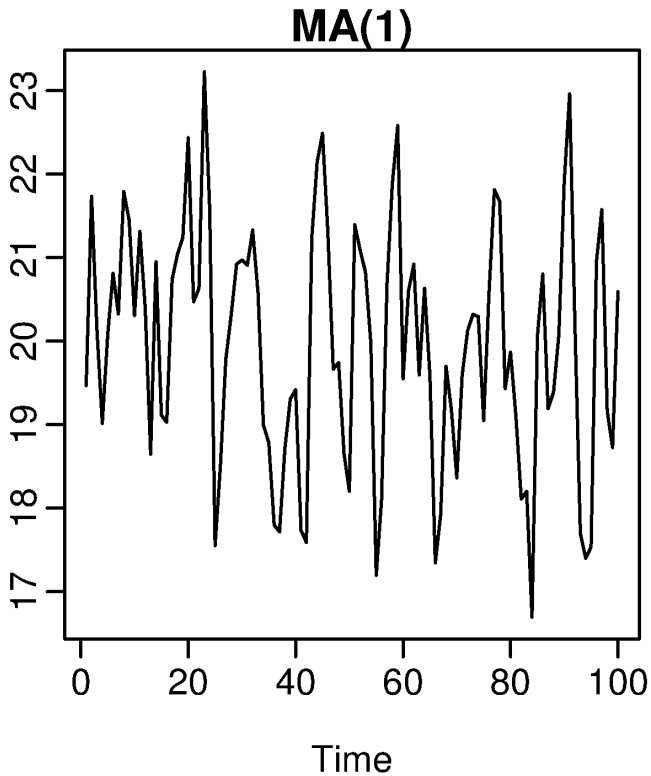
where e_t is white noise. We refer to this as an MA(q) model.

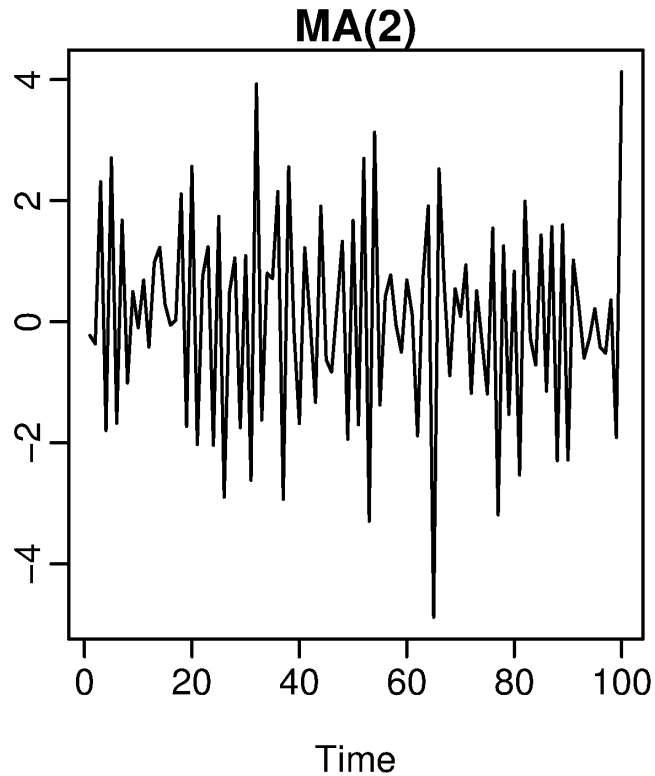
We do not *observe* the values of e_t, so it is not really regression in the usual sense.

Moving average *models* should not be confused with moving average *smoothing*. A moving average *model* is used for forecasting future values while moving average *smoothing* is used for estimating the trend-cycle of past values.

Moving average models

Left: MA(1) with $y_t = 20 + e_t + 0.8e_{t-1}$. Right: MA(2) with $y_t = e_t - e_{t-1} + 0.8e_{t-2}$.
e_t is normally distributed white noise with mean zero and variance one.





Non-seasonal ARIMA models

If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model.

ARIMA is an acronym for AutoRegressive Integrated Moving Average model (*integration* in this context is the reverse of differencing). The full model can be written as

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t, \qquad (1)$$

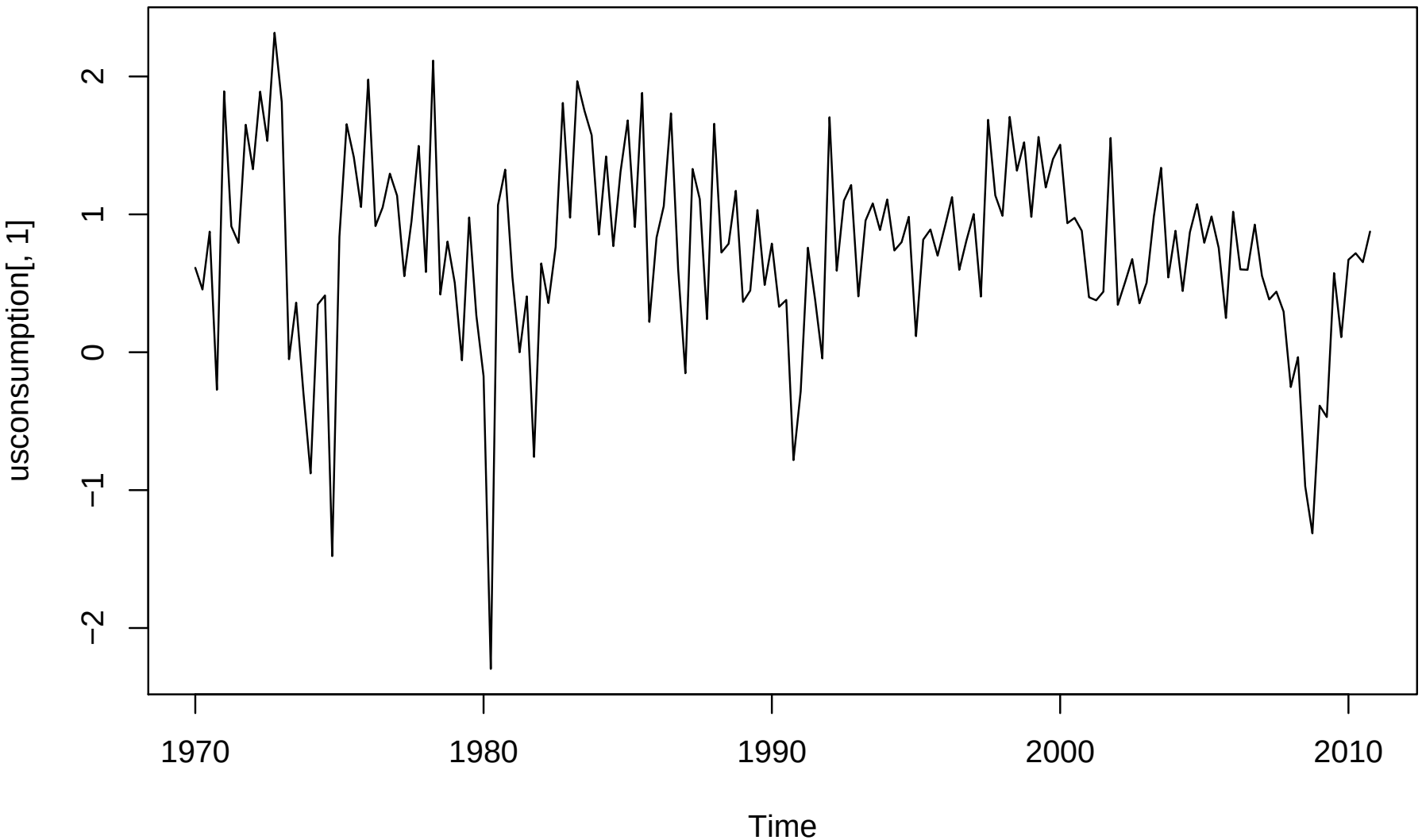
where y'_t is the differenced series (it may have been differenced more than once). The predictors on the right hand side include both lagged values of y_t and lagged errors.

We call this an ARIMA(p, d, q) model, where p = order of the autoregressive part; d = degree of first differencing involved; q = order of the moving average part.

Non-seasonal ARIMA models

Selecting appropriate values for p, d and q can be difficult. The `auto.arima()` function in `R` will do it for you automatically.

```r
plot(usconsumption[, 1])
```



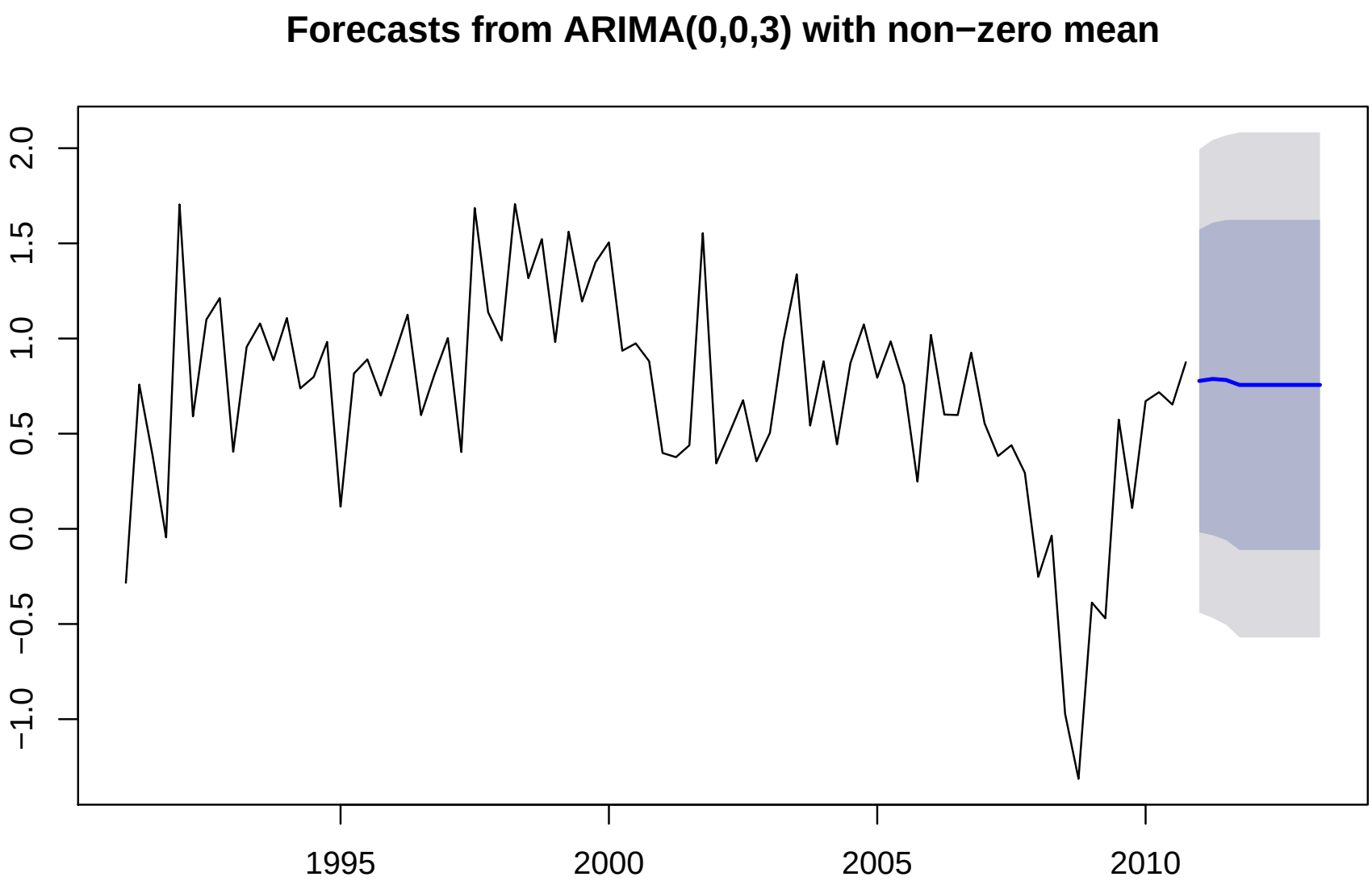
```
fit <- auto.arima(usconsumption[, 1], seasonal = FALSE)
fit

Series: usconsumption[, 1]
ARIMA(0,0,3) with non-zero mean

Coefficients:
         ma1     ma2     ma3  intercept
      0.2542  0.2260  0.2695     0.7562
s.e.  0.0767  0.0779  0.0692     0.0844

sigma^2 estimated as 0.3856:  log likelihood=-154.73
AIC=319.46   AICc=319.84   BIC=334.96
```

This is an ARIMA(0,0,3) or MA(3) model:

$$y_t = 0.756 + e_t + 0.254e_{t-1} + 0.226e_{t-2} + 0.269e_{t-3},$$

where e_t is white noise with standard deviation $0.62 = \sqrt{0.3856}$.

```r
plot(forecast(fit, h = 10), include = 80)
```



ACF and PACF plots

It is usually not possible to tell, simply from a time plot, what values of p and q are appropriate for the data. However, it is sometimes possible to use the ACF plot, and the closely related PACF plot, to determine appropriate values for p and q.

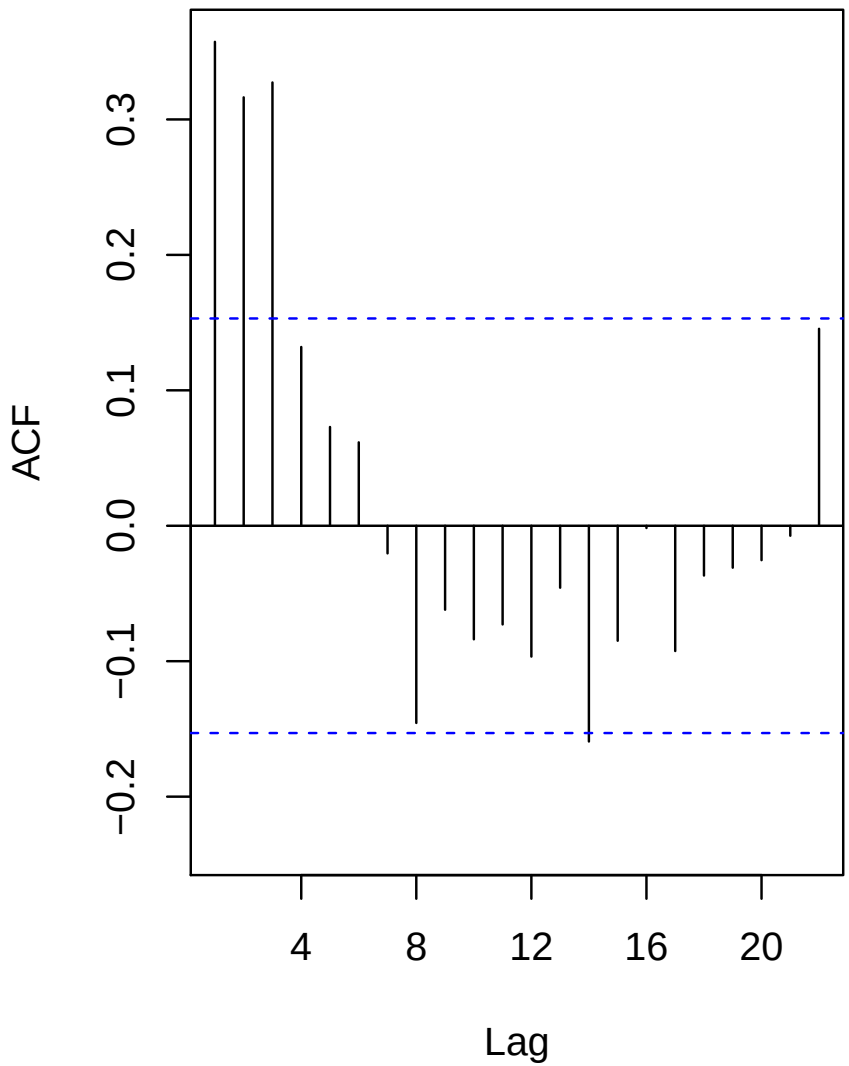
Recall that an ACF plot shows the autocorrelations which measure the relationship between y_t and y_{t-k} for different values of k. Now if y_t and y_{t-1} are correlated, then y_{t-1} and y_{t-2} must also be correlated. But then y_t and y_{t-2} might be correlated, simply because they are both connected to y_{t-1}.

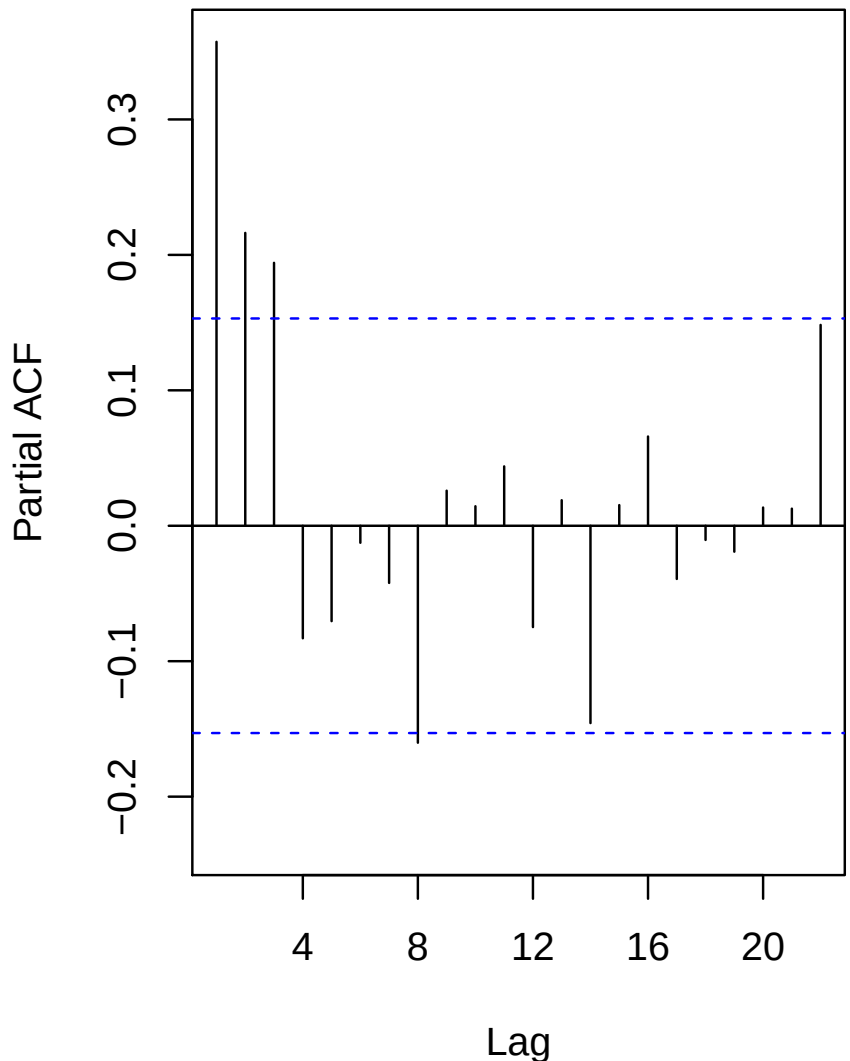
The partial autocorrelations measure the relationship between y_t and y_{t-k} after removing the effects of other time lags $1, 2, 3, \ldots, k-1$. So the first partial autocorrelation is identical to the first autocorrelation, because there is nothing between them to remove. The partial autocorrelations for lags 2, 3 and greater are calculated as follows:

$$\begin{aligned}\alpha_k &= k\text{th partial autocorrelation coefficient} \\ &= \text{the estimate of } \phi_k \text{ in the autoregression model} \\ y_t &= c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_k y_{t-k} + e_t.\end{aligned}$$

Varying the number of terms on the right hand side of this autoregression model gives α_k for different values of k.

```
par(mfrow = c(1, 2))
Acf(usconsumption[, 1], main = "")
Pacf(usconsumption[, 1], main = "")
```





The data may follow an ARIMA$(p, d, 0)$ model if the ACF and PACF plots of the differenced data show the following patterns:

- the ACF is exponentially decaying or sinusoidal;
- there is a significant spike at lag p in PACF, but none beyond lag p.

The data may follow an ARIMA$(0, d, q)$ model if the ACF and PACF plots of the differenced data show the following patterns:

- the PACF is exponentially decaying or sinusoidal;
- there is a significant spike at lag q in ACF, but none beyond lag q.

Maximum likelihood estimation

Once the model order has been identified (i.e., the values of p, d and q), we need to estimate the parameters $c, \phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q$. When `R` estimates the ARIMA model, it uses maximum likelihood estimation (MLE).

This technique finds the values of the parameters which maximize the probability of obtaining the data that we have observed.

For ARIMA models, MLE is very similar to the least squares estimates that would be obtained by minimizing

$$\sum_{t=1}^{T} e_t^2.$$

ARIMA modelling in R

The `auto.arima()` function in R uses a variation of the Hyndman and Khandakar algorithm, which combines unit root tests, minimization of the AICc and MLE to obtain an ARIMA model.

If you want to choose the model yourself, use the `Arima()` function in R . For example, to fit the ARIMA(0,0,3) model to the US consumption data, the following commands can be used: `fit <- Arima(usconsumption[,1], order=c(0,0,3))`

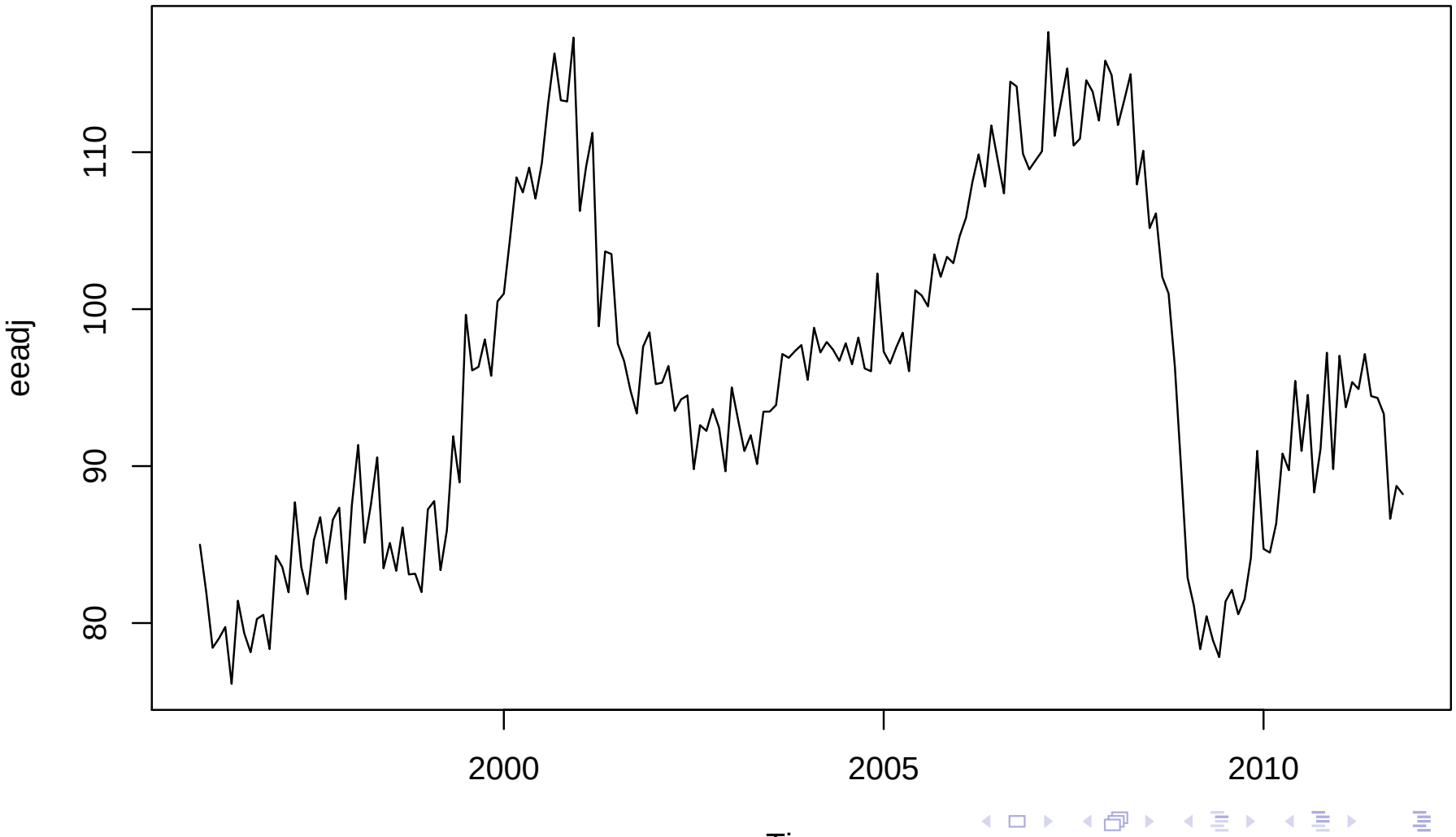
There is another function `arima()` in R which also fits an ARIMA model, but it has some shortcomings so don't use it.

Modelling procedure

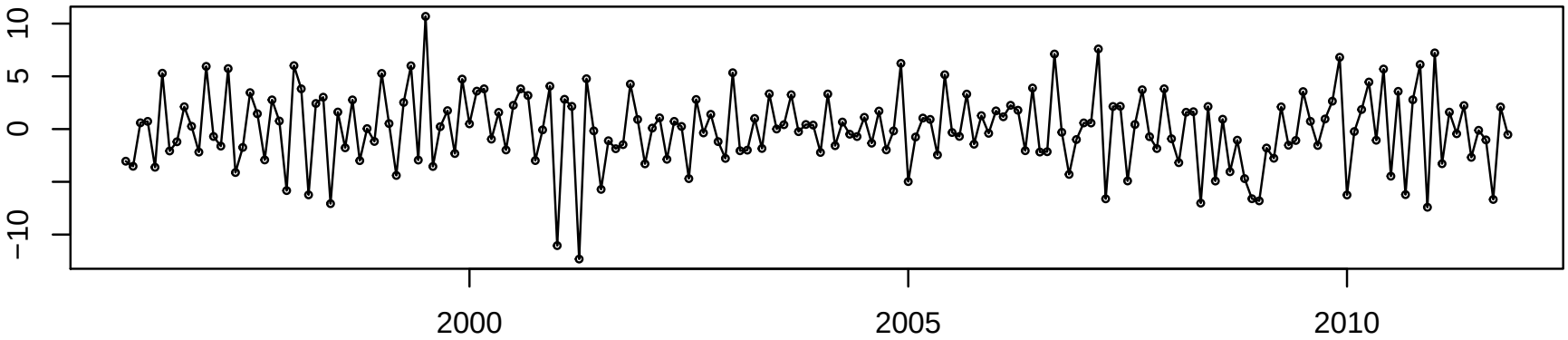
1. Plot the data. Identify any unusual observations.
2. If necessary, transform the data (using a Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary: take first differences of the data until the data are stationary.
4. Examine the ACF/PACF: Is an AR(p) or MA(q) model appropriate?
5. Try your chosen model(s), and use the AICc to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts.

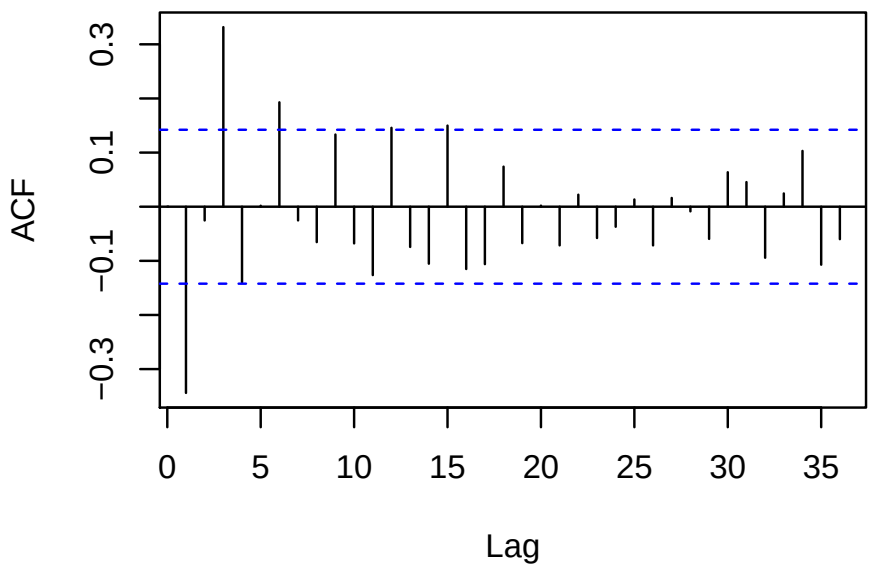
Example

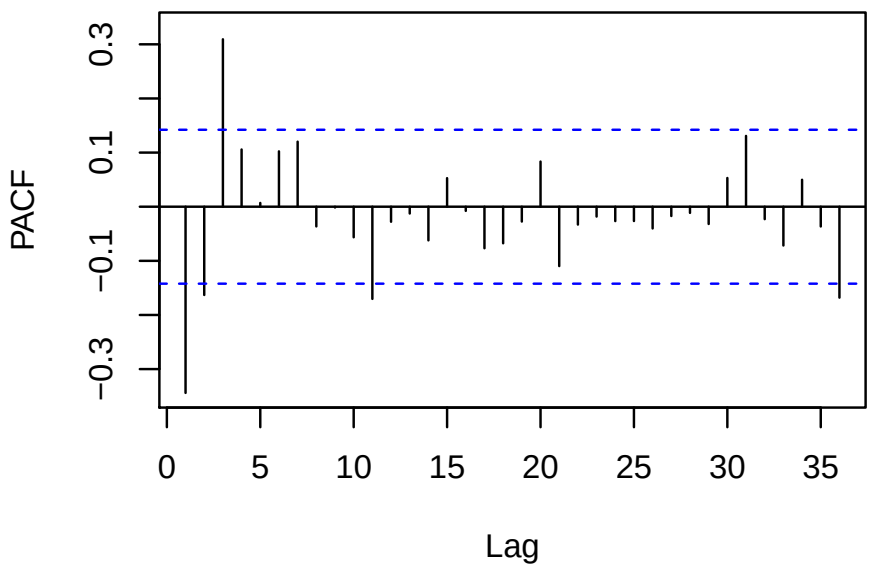
```
eeadj <- seasadj(stl(elecequip, s.window = "periodic"))
plot(eeadj)
```



```r
tsdisplay(diff(eeadj), main = "")
```







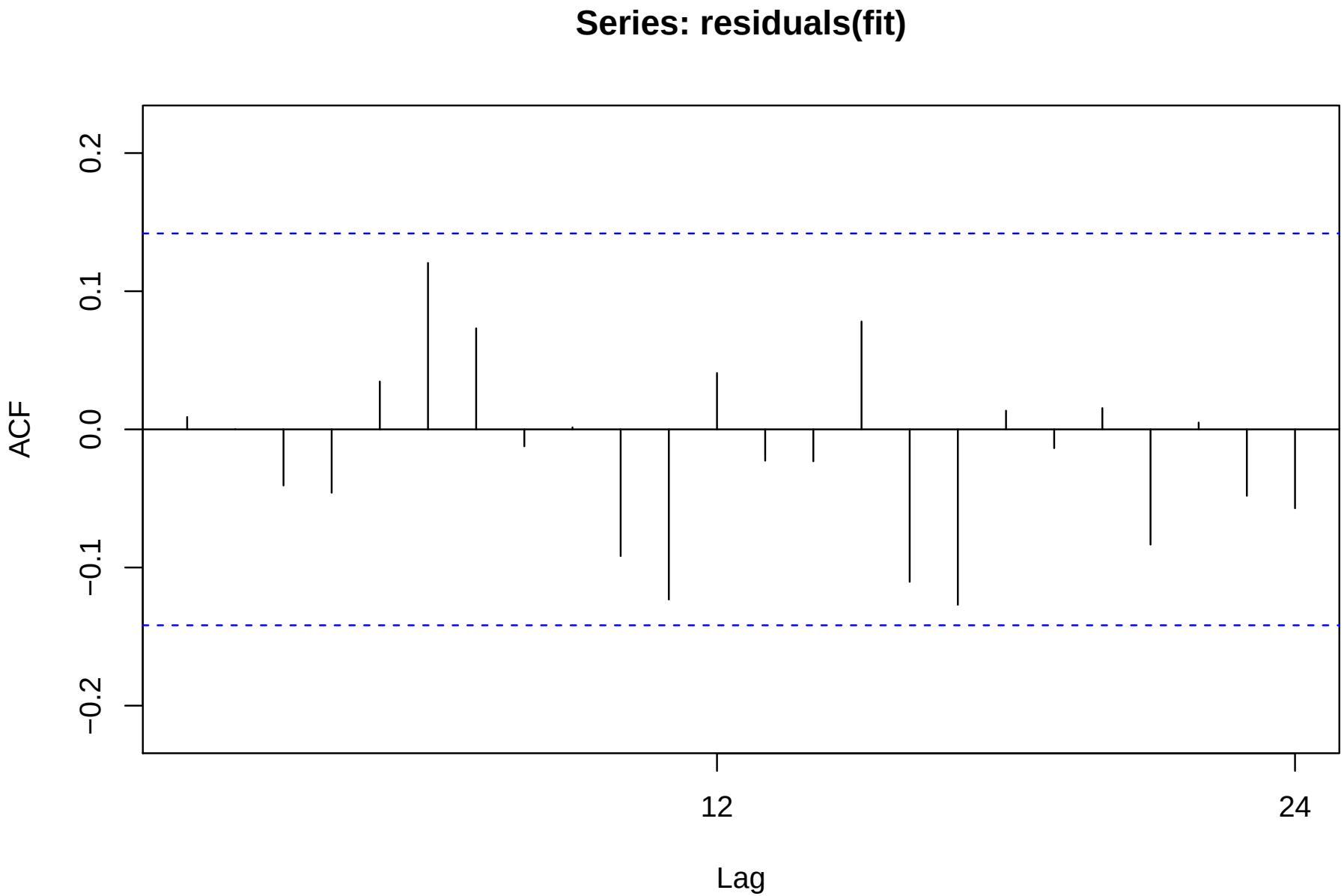
```r
fit <- Arima(eeadj, order = c(3, 1, 0))
summary(fit)
```

```
Series: eeadj
ARIMA(3,1,0)

Coefficients:
          ar1      ar2     ar3
      -0.3488  -0.0386  0.3139
s.e.   0.0690   0.0736  0.0694

sigma^2 estimated as 9.697:  log likelihood=-485.67
AIC=979.33   AICc=979.55   BIC=992.32

Training set error measures:
                     ME     RMSE      MAE          MPE     MAPE      MASE
Training set 0.01170679 3.105828 2.430723 -0.04353974 2.560168 0.2964478
                    ACF1
Training set -0.03463506
```

```r
fit <- Arima(eeadj, order = c(3, 1, 1))
summary(fit)
```

```
Series: eeadj
ARIMA(3,1,1)

Coefficients:
         ar1     ar2     ar3      ma1
      0.0519  0.1191  0.3730  -0.4542
s.e.  0.1840  0.0888  0.0679   0.1993

sigma^2 estimated as 9.532:  log likelihood=-484.08
AIC=978.17   AICc=978.49   BIC=994.4

Training set error measures:
                      ME     RMSE      MAE          MPE     MAPE      MASE
Training set -0.001227744 3.079373 2.389267 -0.04290849 2.517748 0.2913919
                    ACF1
Training set 0.008928479
```

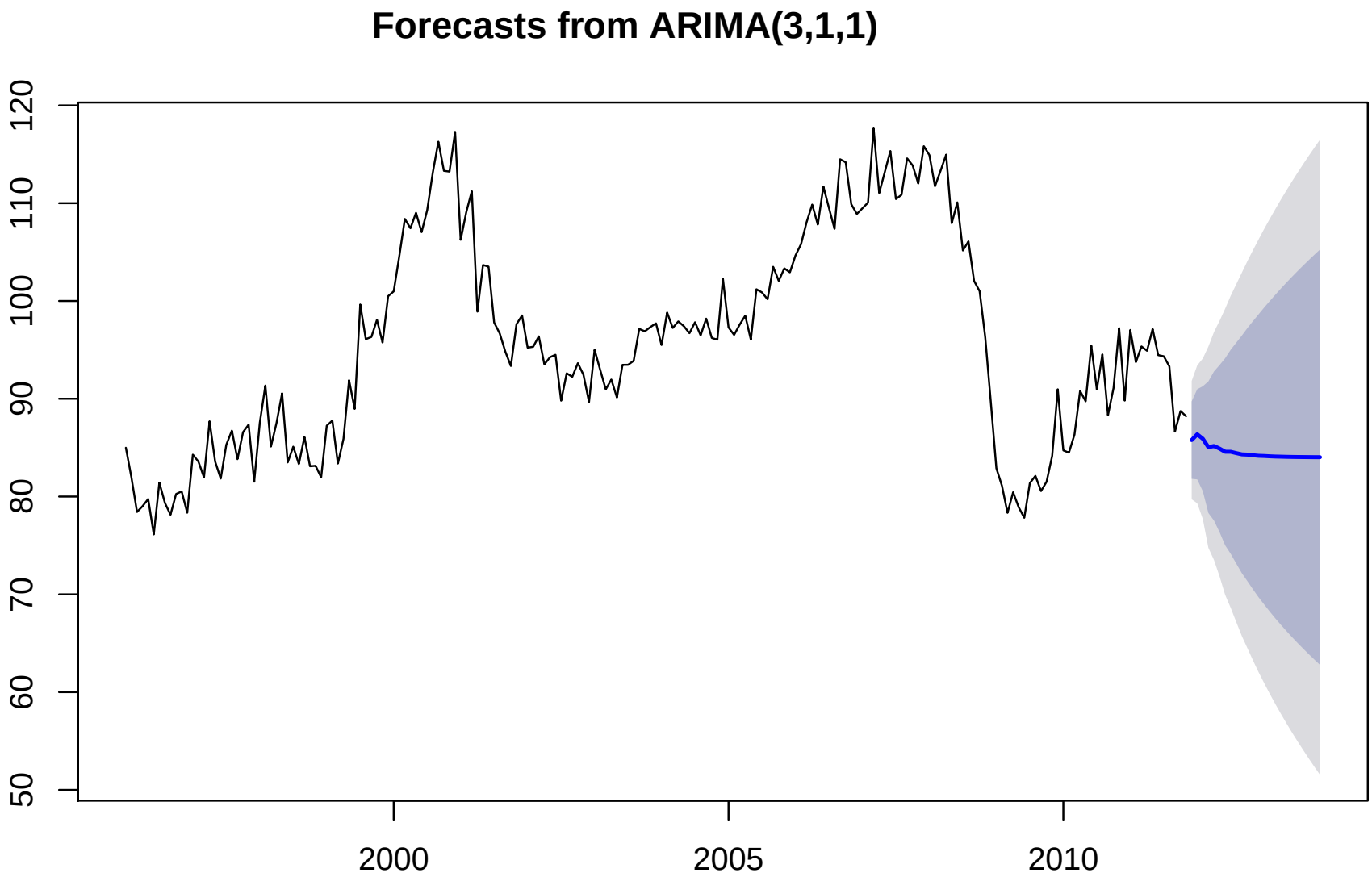
```r
Acf(residuals(fit))
```



```r
Box.test(residuals(fit), lag = 24, fitdf = 4, type = "Ljung")
```

```
Box-Ljung test

data:  residuals(fit)
X-squared = 20.496, df = 20, p-value = 0.4273
```

```
plot(forecast(fit))
```



Smoothing and Decomposition methods

Learning objectives:

1. Identify and interpret additive and multiplicative decompositions
2. Decompose a time series
   - Additive decomposition
   - Multiplicative decomposition
3. Apply a moving averages smoother
4. Apply a single exponential smoother
5. Forecasts

Time series decomposition

- Time series data can exhibit a huge variety of patterns and it is helpful to categorize some of the patterns and behaviours that can be seen in time series.
- It is also sometimes useful to try to split a time series into several components, each representing one of the underlying categories of pattern.
- we consider some common patterns and methods to extract the associated components from a time series.
- Often this is done to help understand the time series better, but it can also be used to improve forecasts.

Time series components

- Trend: A trend exists when there is a long-term increase or decrease in the data. It does not have to be linear. Sometimes we will refer to a trend "changing direction" when it might go from an increasing trend to a decreasing trend.

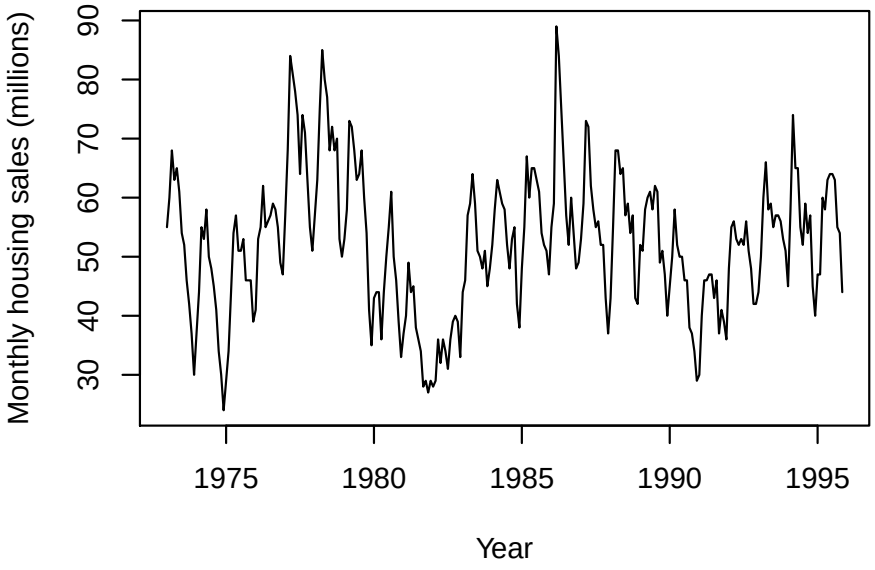
- Seasonal: A seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week). Seasonality is always of a fixed and known period.

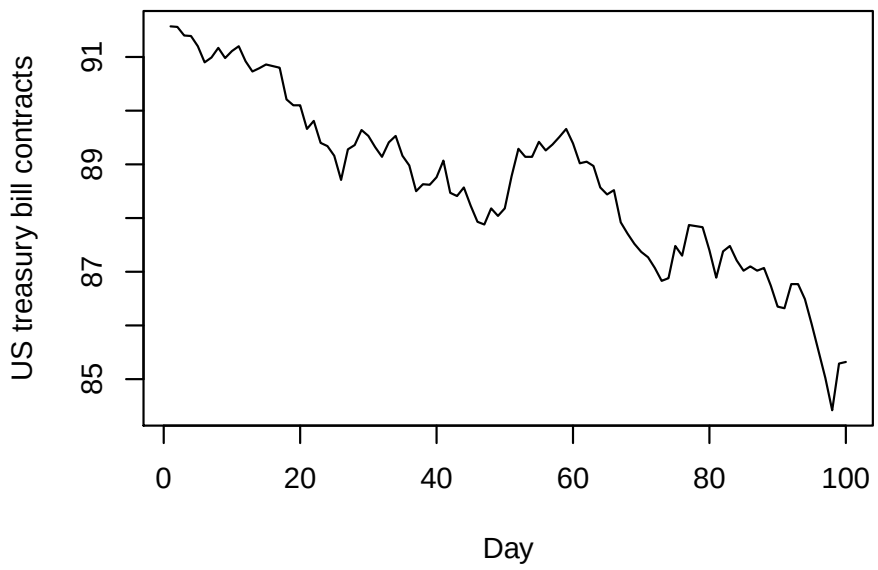
- Cyclic A cyclic pattern exists when data exhibit rises and falls that are not of fixed period. The duration of these fluctuations is usually of at least 2 years.

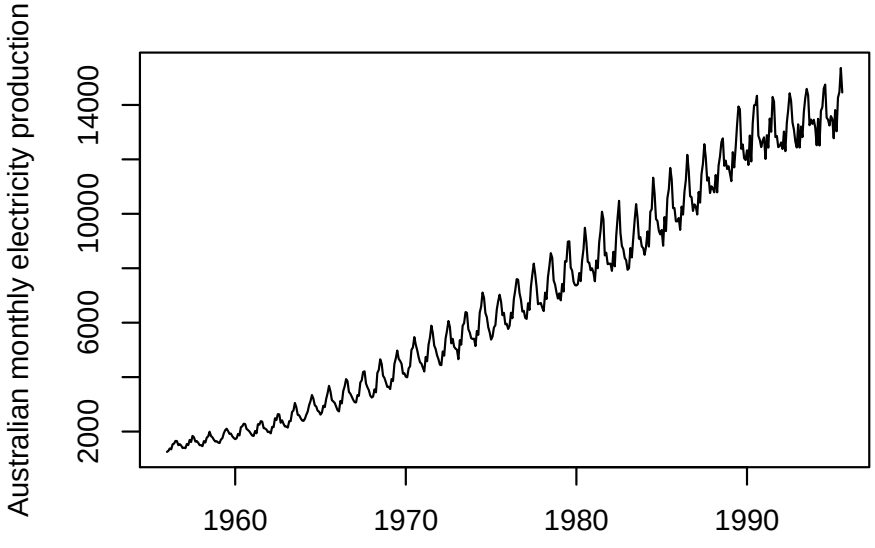
Difference between cyclic and seasonal behaviour

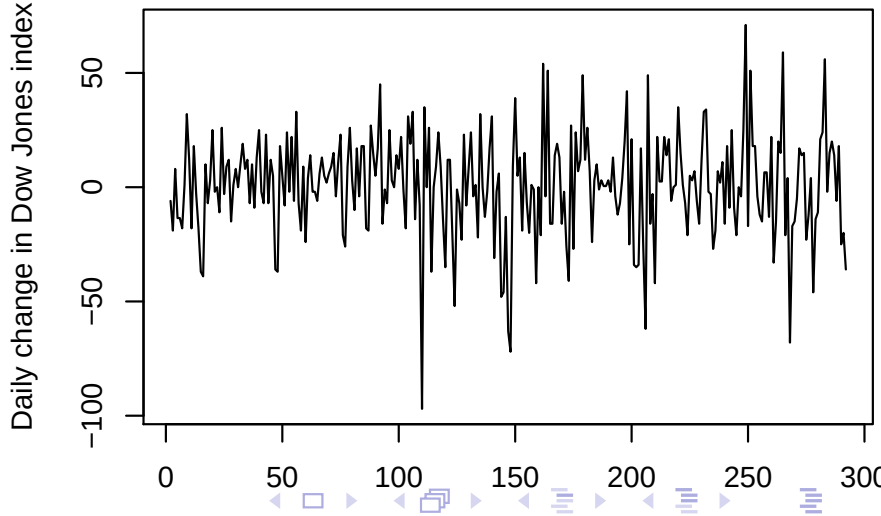
- If the fluctuations are not of fixed period then they are cyclic;
- if the period is unchanging and associated with some aspect of the calendar, then the pattern is seasonal.
- In general, the average length of cycles is longer than the length of a seasonal pattern, and the magnitude of cycles tends to be more variable than the magnitude of seasonal patterns.

Different types of time series patterns.









R code

```
par(mfrow = c(2, 2))
plot(hsales, xlab = "Year", ylab = "Monthly housing sales (millions)")
plot(ustreas, xlab = "Day", ylab = "US treasury bill contracts")
plot(elec, xlab = "Year", ylab = "Australian monthly electricity production")
plot(diff(dj), xlab = "Day", ylab = "Daily change in Dow Jones index")
```

Time series exhibiting different types of time series patterns.

1. The monthly housing sales (top left) show strong seasonality within each year, as well as some strong cyclic behaviour with period about 6-10 years. There is no apparent trend in the data over this period.
2. The US treasury bill contracts (top right) show results from the Chicago market for 100 consecutive trading days in 1981. Here there is no seasonality, but an obvious downward trend. Possibly, if we had a much longer series, we would see that this downward trend is actually part of a long cycle, but when viewed over only 100 days it appears to be a trend.
3. The Australian monthly electricity production (bottom left) shows a strong increasing trend, with strong seasonality. There is no evidence of any cyclic behaviour here.
4. The daily change in the Dow Jones index (bottom right) has no trend, seasonality or cyclic behaviour. There are random fluctuations which do not appear to be very predictable, and no strong patterns that would help with developing a forecasting model.

Time series decomposition

We shall think of the time series y_t as comprising three components: a seasonal component, a trend-cycle component (containing both trend and cycle), and a remainder component (containing anything else in the time series)

Additive model

$$Y_t = S_t + T_t + E_t \tag{2}$$

where Y_t is the data at period t, S_t is the seasonal component at period t, T_t is the trend-cycle component at period t and E_t is the remainder (or irregular or error) component at period t.

Multiplicative model

$$Y_t = S_t \times T_t \times E_t \tag{3}$$

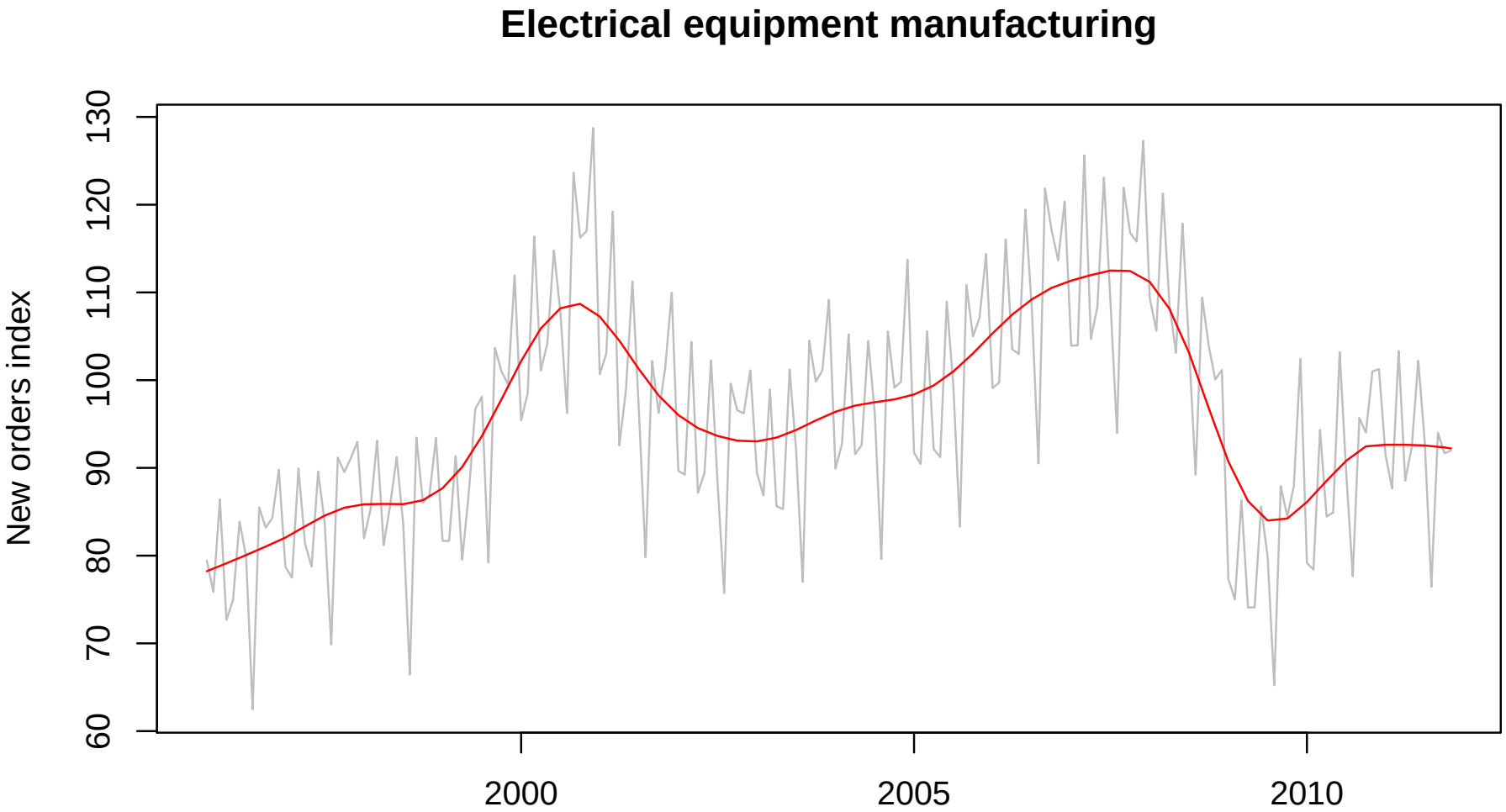
Choice between additive and multiplicative model

- The additive model is most appropriate if the magnitude of the seasonal fluctuations or the variation around the trend-cycle does not vary with the level of the time series.
- When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative model is more appropriate.
- An alternative to using a multiplicative model, is to first transform the data until the variation in the series appears to be stable over time, and then use an additive model. When a log transformation has been used, this is equivalent to using a multiplicative decomposition:

$$Y_t = S_t \times T_t \times E_t \Leftrightarrow logY_t = logS_t + logT_t + logE_t$$

Example: Electrical equipment manufacturing

These data show the number of new orders for electrical equipment (computer, electronic and optical products) in the Euro area (16 countries).

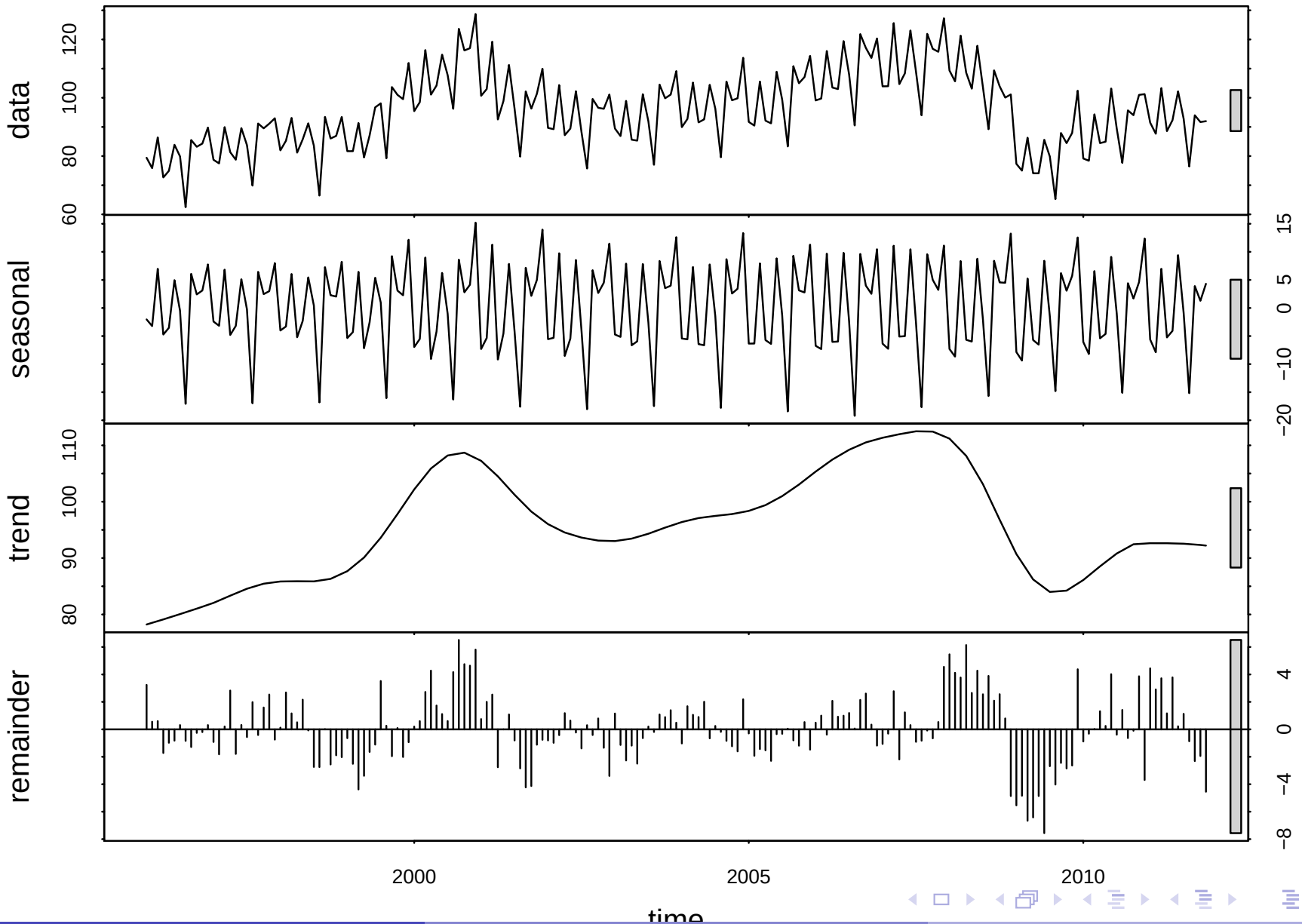


R code

```
fit <- stl(elecequip, s.window = 5)
plot(elecequip, col = "gray", main = "Electrical equipment manufacturing",
    ylab = "New orders index", xlab = "")
lines(fit$time.series[, 2], col = "red", ylab = "Trend")
```

Figure shows the trend-cycle component, T_t, in red and the original data, Y_t, in grey. The trend-cycle shows the overall movement in the series, ignoring the seasonality and any small random fluctuations.

Additive decomposition: STL method

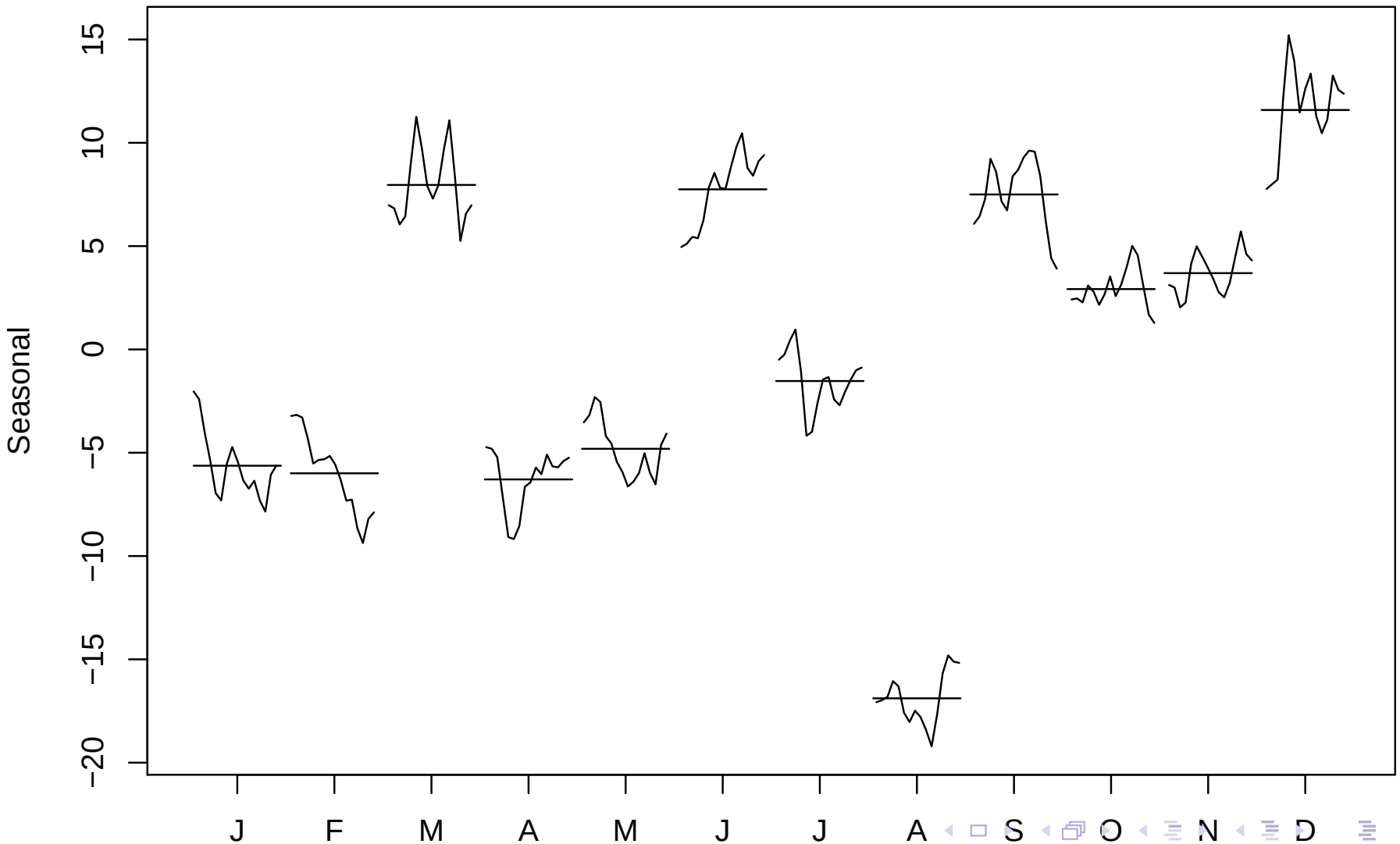


Additive decomposition: STL method

- The seasonal component changes very slowly over time, so any two consecutive years have very similar pattern, but years far apart may have different seasonal patterns.
- The remainder component shown in the bottom panel is what is left over when the seasonal and trend-cycle components have been subtracted from the data.
- In general, the average length of cycles is longer than the length of a seasonal pattern, and the magnitude of cycles tends to be more variable than the magnitude of seasonal patterns.

seasonal sub-series plots of the seasonal component

```
monthplot(fit$time.series[, "seasonal"], main = "", ylab = "Seasonal")
```



Seasonally adjusted data

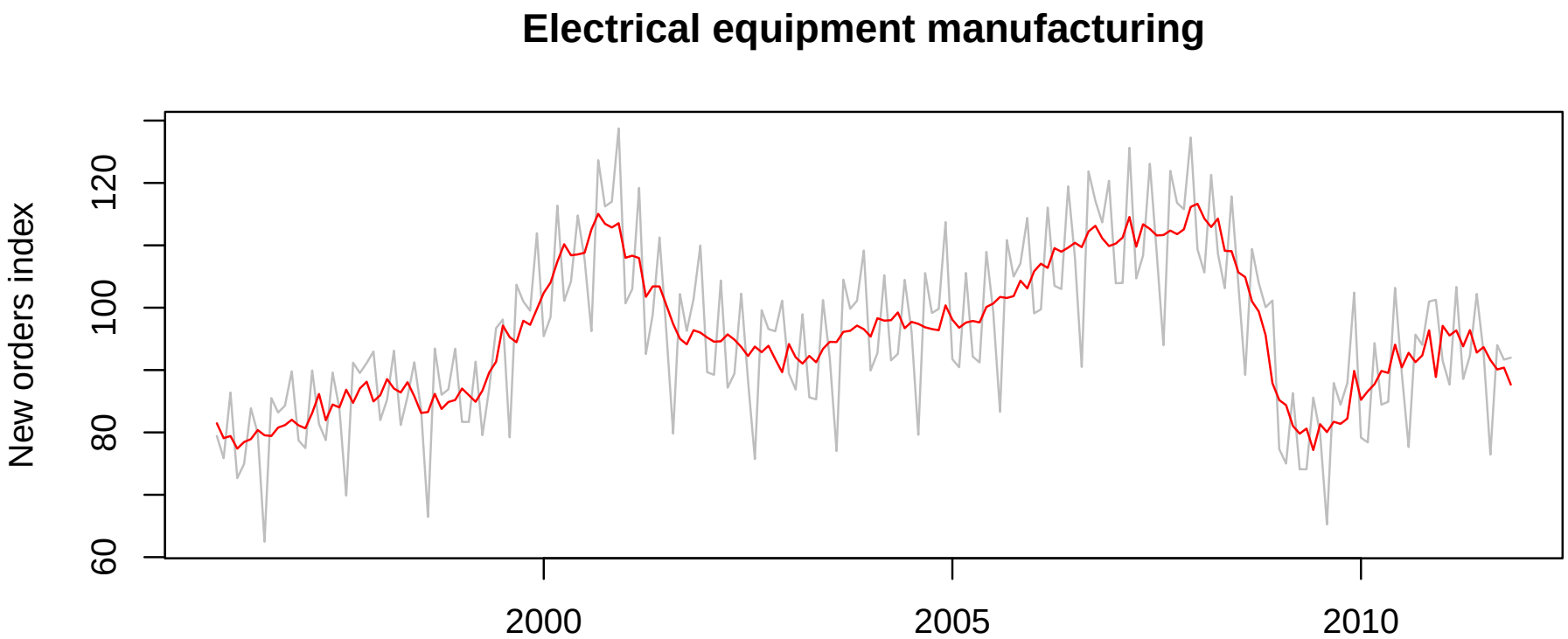
If the seasonal component is removed from the original data, the resulting values are called the seasonally adjusted data. For an additive model, the seasonally adjusted data are given by $Y_t - S_t$, and for multiplicative data, the seasonally adjusted values are obtained using Y_t / S_t.

```
plot(elecequip, col = "grey", main = "Electrical equipment manufacturing",
    xlab = "", ylab = "New orders index")
lines(seasadj(fit), col = "red", ylab = "Seasonally adjusted")
```

Seasonally adjusted data

If the variation due to seasonality is not of primary interest, the seasonally adjusted series can be useful.

```
plot(elecequip, col = "grey", main = "Electrical equipment manufacturing",
    xlab = "", ylab = "New orders index")
lines(seasadj(fit), col = "red", ylab = "Seasonally adjusted")
```



Moving averages

The first step in a classical decomposition is to use a moving average method to estimate the trend-cycle.

Moving average smoothing: called *m*-MA

A moving average of order m can be written as

$$\widehat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j}, \tag{4}$$

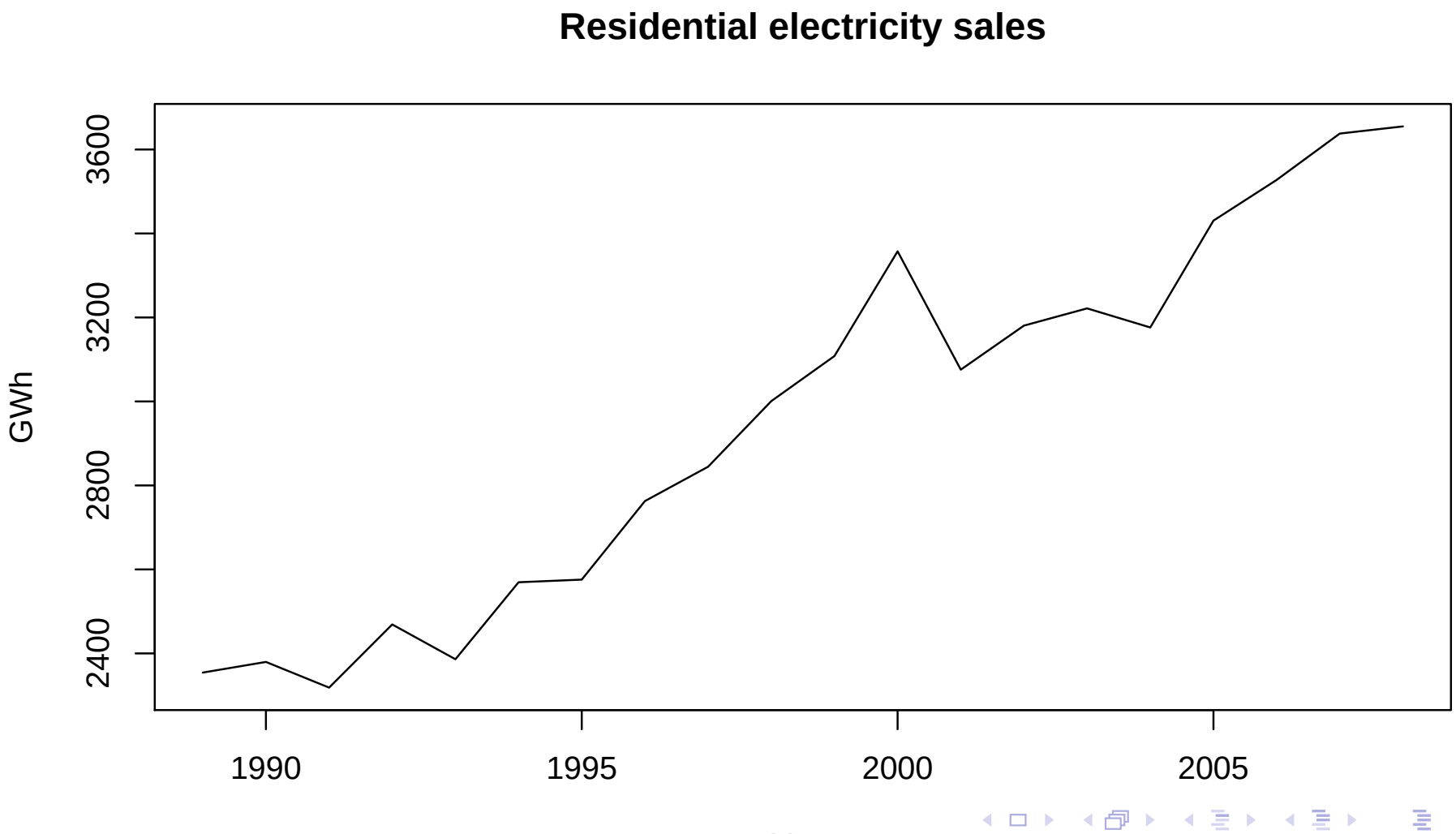
where $m = 2k + 1$.

The estimate of the trend-cycle at time t is obtained by averaging values of the time series within k periods of t. Observations that are nearby in time are also likely to be close in value, and the average eliminates some of the randomness in the data, leaving a smooth trend-cycle component.

Example

Example: the volume of electricity sold to residential customers in South Australia each year from 1989 to 2008 (hot water sales have been excluded).

```
plot(elecsales, main = "Residential electricity sales", ylab = "GWh",
    xlab = "Year")
```



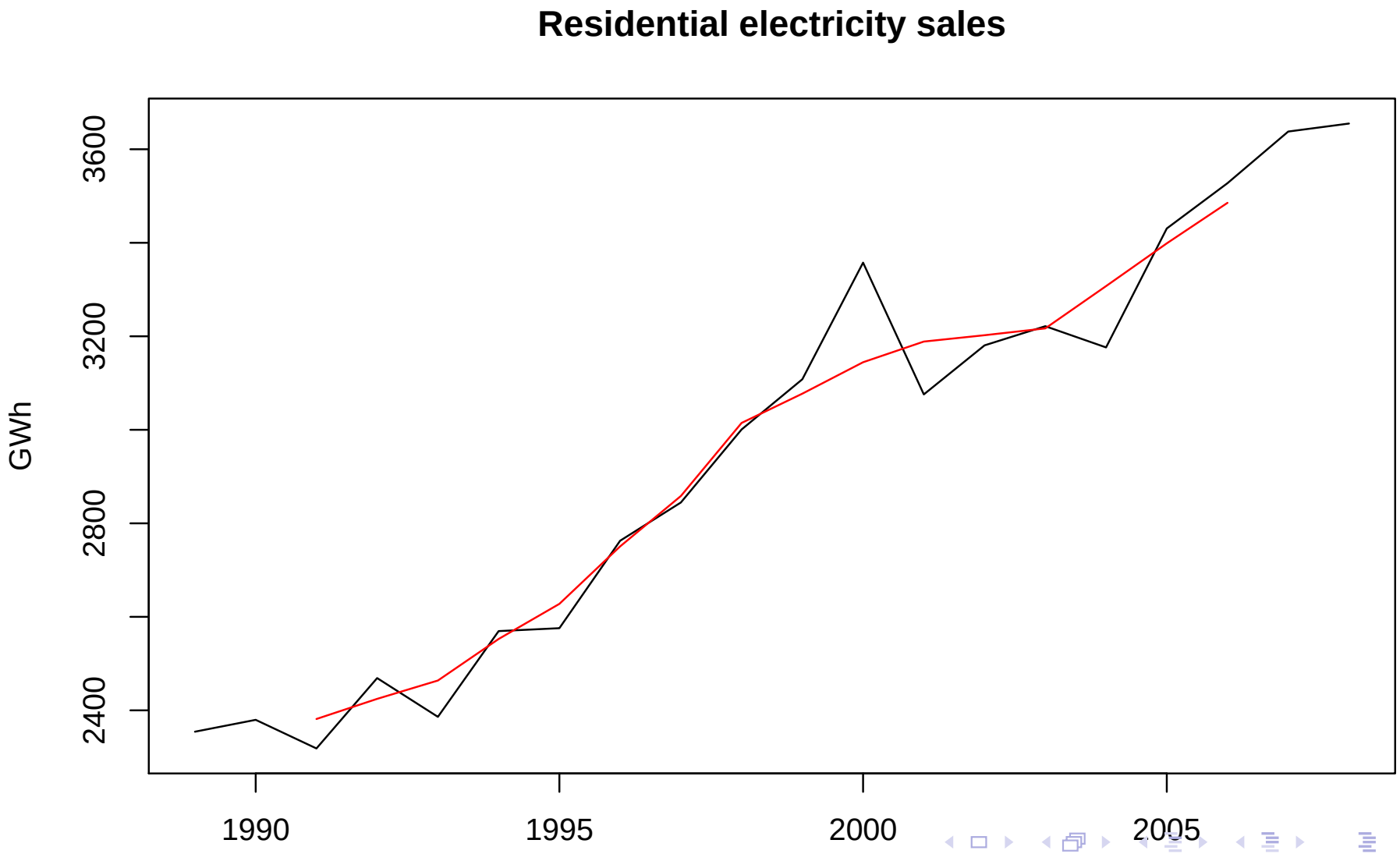
Example:a moving average of order 5

```
ma(elecsales, order = 5)

Time Series:
Start = 1989
End = 2008
Frequency = 1
 [1]       NA       NA 2381.530 2424.556 2463.758 2552.598 2627.700
 [8] 2750.622 2858.348 3014.704 3077.300 3144.520 3188.700 3202.320
[15] 3216.940 3307.296 3398.754 3485.434       NA       NA
```

Here, $m = 5 = 2k + 1$ so $k = 2$. Then, there are no values for the first two years or last two years because we don't have two observations on either side.

Visualisation

```
plot(elecsales, main = "Residential electricity sales", ylab = "GWh",
    xlab = "Year")
lines(ma(elecsales, 5), col = "red")
```

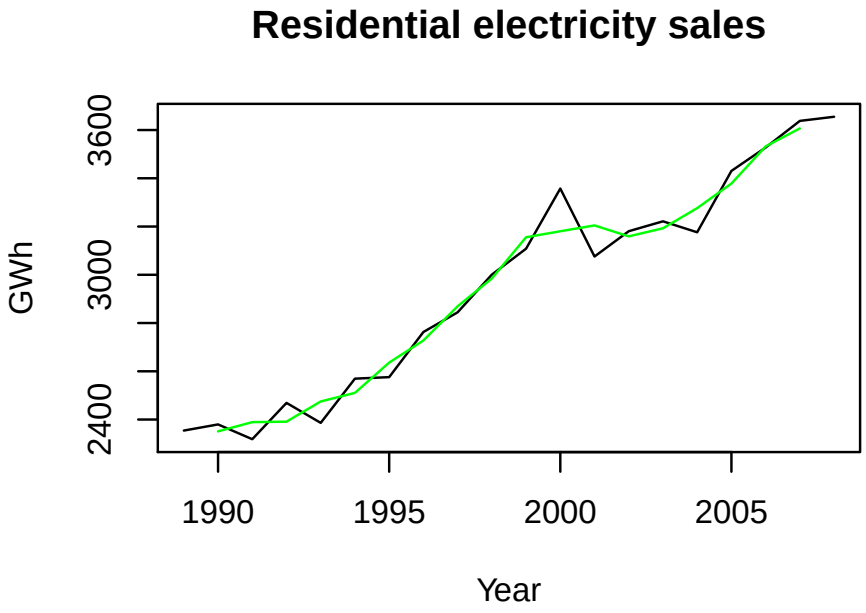


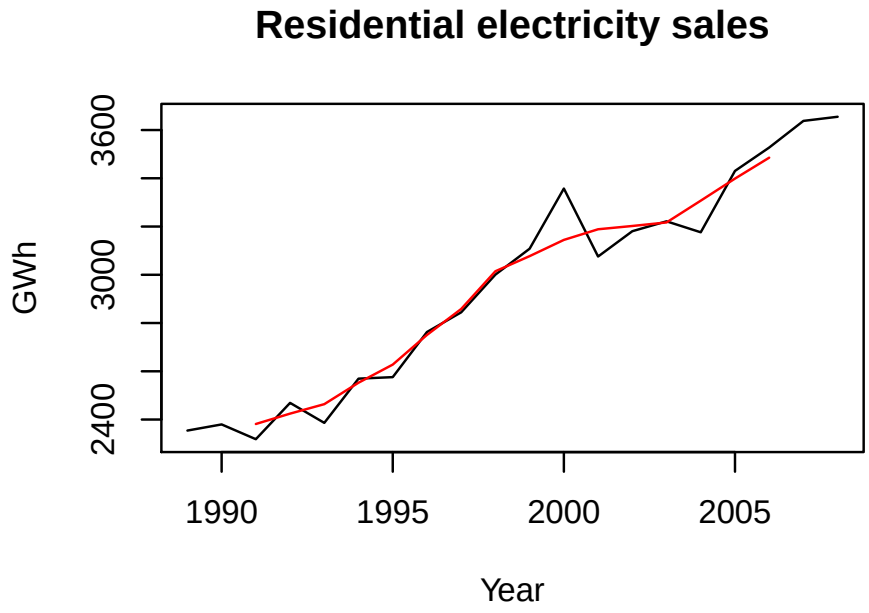
Order of moving average

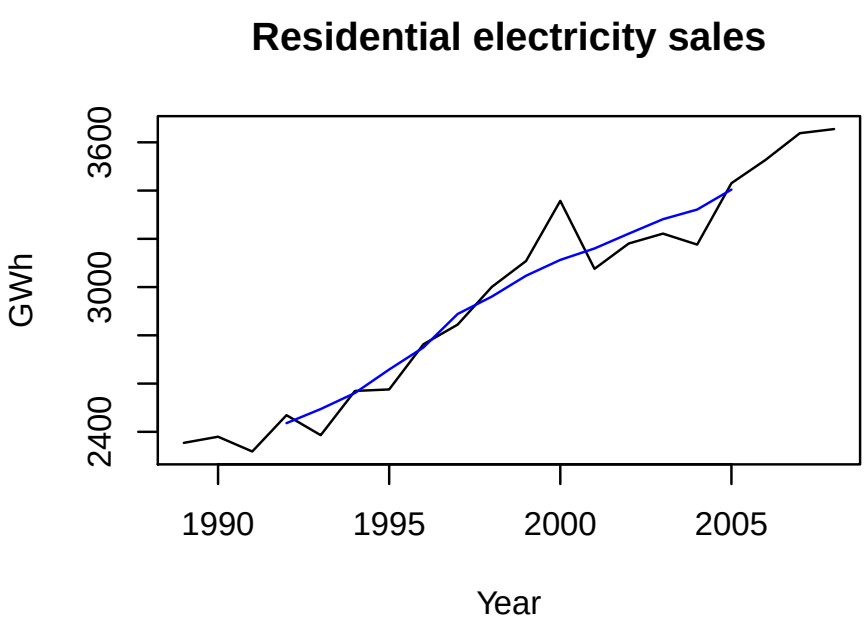
The order of the moving average determines the smoothness of the trend-cycle estimate. In general, a larger order means a smoother curve.

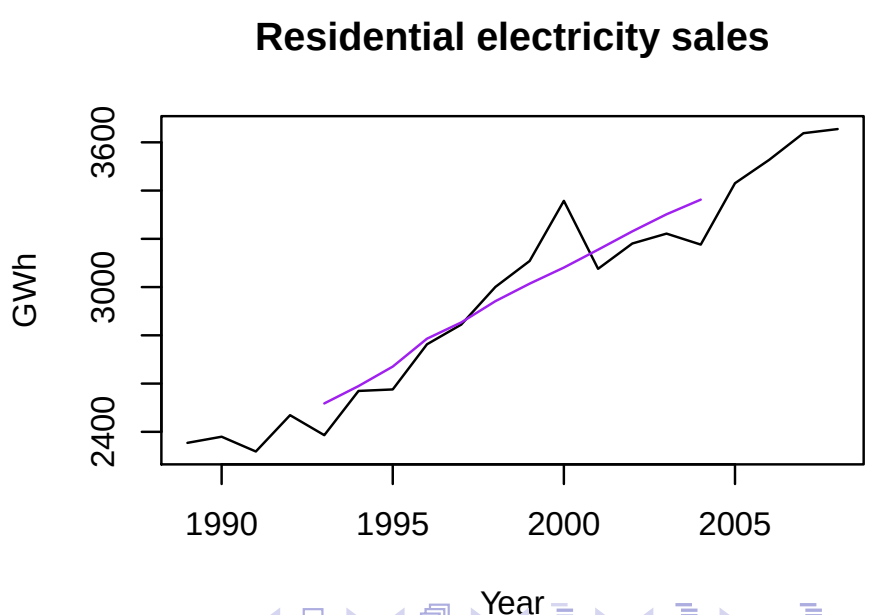
```
par(mfrow = c(2, 2))
plot(elecsales, main = "Residential electricity sales", ylab = "GWh",
    xlab = "Year")
lines(ma(elecsales, 3), col = "green")
plot(elecsales, main = "Residential electricity sales", ylab = "GWh",
    xlab = "Year")
lines(ma(elecsales, 5), col = "red")
plot(elecsales, main = "Residential electricity sales", ylab = "GWh",
    xlab = "Year")
lines(ma(elecsales, 7), col = "blue")
plot(elecsales, main = "Residential electricity sales", ylab = "GWh",
    xlab = "Year")
lines(ma(elecsales, 9), col = "purple")
```

Order of moving average









Moving averages of moving averages

It is possible to apply a moving average to a moving average. One reason for doing this is to make an even-order moving average symmetric.

For example, we might take a moving average of order 4, and then apply another moving average of order 2 to the results.

```
beer2 <- window(ausbeer, start = 1992)
ma4 <- ma(beer2, order = 4, centre = FALSE)
ma2x4 <- ma(beer2, order = 4, centre = TRUE)
year <- rep(1992:2008, each = 4)
quarter <- rep(c("Q1", "Q2", "Q3", "Q4"), times = 17)
result <- data.frame(Year = paste(year[-68], quarter[-68]), Data = as.numeric(beer2)
    `4-MA` = ma4, `.2x4-MA` = ma2x4)
```

Moving averages of moving averages

```
      Year Data  X4.MA X.2x4.MA
1  1992 Q1  443     NA       NA
2  1992 Q2  410 451.25       NA
3  1992 Q3  420 448.75  450.000
4  1992 Q4  532 451.50  450.125
5  1993 Q1  433 449.00  450.250
6  1993 Q2  421 444.00  446.500
7  1993 Q3  410 448.00  446.000
8  1993 Q4  512 438.00  443.000
9  1994 Q1  449 441.25  439.625
10 1994 Q2  381 446.00  443.625
```

- Notation 2×4-MA in the last column means a 4-MA followed by a 2-MA.
- The values in the last column are obtained by taking a moving average of order 2 of the values in the previous column.
- For example, the first two values in the 4−MA column are $451.2 = (443 + 410 + 420 + 532)/4$ and $448.8 = (410 + 420 + 532 + 433)/4$.
- The first value in the 2×4-MA column is the average of these two:$450.0 = (451.2 + 448.8)/2$.
- When a 2-MA follows a moving average of even order (such as 4), it is called a centered moving average of order 4.

Estimating the trend-cycle with seasonal data

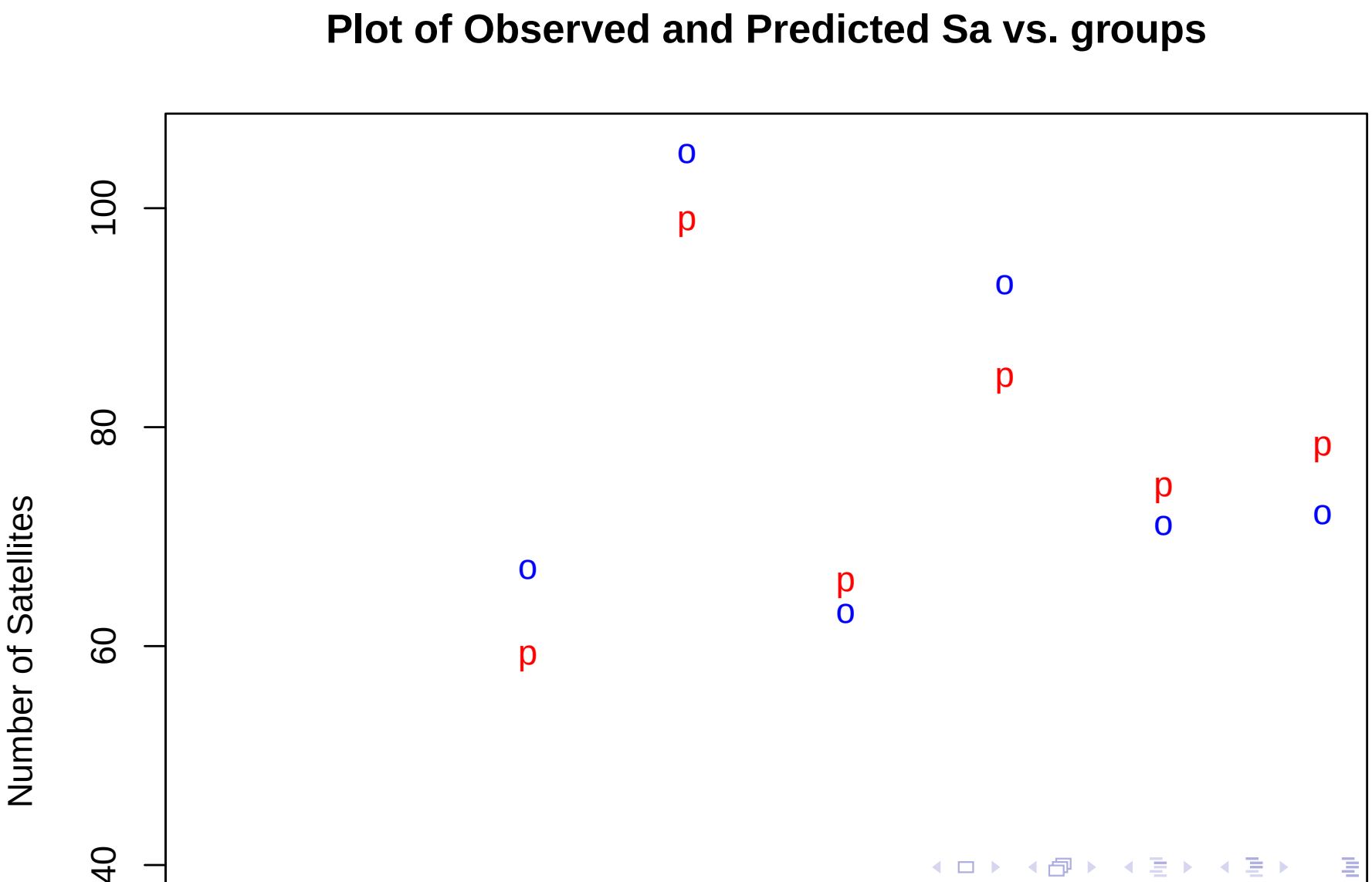
The most common use of centered moving averages is in estimating the trend-cycle from seasonal data. Consider the 2×4-MA:

$$\widehat{T}_t = \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}$$

- When applied to quarterly data, each quarter of the year is given equal weight as the first and last terms apply to the same quarter in consecutive years.
- Consequently, the seasonal variation will be averaged out and the resulting values of $\widehat{T}_t$ will have little or no seasonal variation remaining.
- So if the seasonal period is even and of order m, use a 2×4-MA to estimate the trend-cycle.
- If the seasonal period is odd and of order m, use a $m - MA$ to estimate the trend cycle
- In particular, a 2×12-MA can be used to estimate the trend-cycle of monthly data and a 7-MA can be used to estimate the trend-cycle of daily data.

Electrical equipment manufacturing: 2×12-MA

```
plot(elecequip, ylab = "New orders index", col = "gray", main = "Electrical equipment
lines(ma(elecequip, order = 12), col = "red")
```



Classical decomposition

There are two forms of classical decomposition: an additive decomposition and a multiplicative decomposition. These are described below for a time series with seasonal period m (e.g., $m = 4$ for quarterly data, $m = 12$ for monthly data, $m = 7$ for daily data with a weekly pattern).

In classical decomposition, we assume the seasonal component is constant from year to year. These m values are sometimes called the seasonal indices.

Additive decomposition

1. If m is an even number, compute the trend-cycle component using a $2 \times m$-MA to obtain $\widehat{T}$. If m is an odd number, compute the trend-cycle component using an m-MA to obtain $\widehat{T}$.

2. Calculate the detrended series: $y_t - \widehat{T}$.

3. To estimate the seasonal component for each month, simply average the detrended values for that month. These seasonal indexes are then adjusted to ensure that they add to zero. The seasonal component is obtained by stringing together all the seasonal indices for each year of data. This gives $\widehat{S}$.

4. The remainder component is calculated by subtracting the estimated seasonal and trend-cycle components: $\widehat{E} = y_t - \widehat{T}_t - \widehat{S}_t$.

Multiplicative decomposition

1. If m is an even number, compute the trend-cycle component using a $2 \times m$-MA to obtain $\widehat{T}$. If m is an odd number, compute the trend-cycle component using an m-MA to obtain $\widehat{T}$.

2. Calculate the detrended series: $y_t / \widehat{T}$.

3. To estimate the seasonal component for each month, simply average the detrended values for that month. These seasonal indexes are then adjusted to ensure that they add to zero. The seasonal component is obtained by stringing together all the seasonal indices for each year of data. This gives $\widehat{S}$.

4. The remainder component is calculated by subtracting the estimated seasonal and trend-cycle components: $\widehat{E} = y_t / (\widehat{T}_t \widehat{S}_t)$.

```
# x is the time series
fit <- decompose(x, type = "multiplicative")
plot(fit)
```

Comments on classical decomposition

While classical decomposition is still widely used, it is not always recommended. There are now several much better methods.

1. The estimate of the trend is unavailable for the first few and last few observations. For example, if $m = 12$, there is no trend estimate for the first six and last six observations. Consequently, there is also no estimate of the remainder component for the same time periods.

2. Classical decomposition methods assume that the seasonal component repeats from year to year. For many series, this is a reasonable assumption, but for some longer series it is not.

3. Occasionally, the value of the time series in a small number of periods may be particularly unusual. For example, monthly air passenger traffic may be affected by an industrial dispute making the traffic during the dispute very different from usual. The classical method is not robust to these kinds of unusual values.

4. Alternatives approaches ....

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

- The X-12-ARIMA method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

- The X-12-ARIMA method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition
- the trend estimate is available for all observations including the end points, and the seasonal component is allowed to vary slowly over time.

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

- The X-12-ARIMA method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition
- the trend estimate is available for all observations including the end points, and the seasonal component is allowed to vary slowly over time.
- It is also relatively robust to occasional unusual observations.

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

- The X-12-ARIMA method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition
- the trend estimate is available for all observations including the end points, and the seasonal component is allowed to vary slowly over time.
- It is also relatively robust to occasional unusual observations.
- X-12-ARIMA handles both additive and multiplicative decomposition, but only allows for quarterly and monthly data.

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

- The X-12-ARIMA method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition
- the trend estimate is available for all observations including the end points, and the seasonal component is allowed to vary slowly over time.
- It is also relatively robust to occasional unusual observations.
- X-12-ARIMA handles both additive and multiplicative decomposition, but only allows for quarterly and monthly data.
- There is no loss of observations at the start and end of the series.

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

- The X-12-ARIMA method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition
- the trend estimate is available for all observations including the end points, and the seasonal component is allowed to vary slowly over time.
- It is also relatively robust to occasional unusual observations.
- X-12-ARIMA handles both additive and multiplicative decomposition, but only allows for quarterly and monthly data.
- There is no loss of observations at the start and end of the series.
- X-12-ARIMA also has some sophisticated methods to handle trading day variation, holiday effects and the effects of known predictors, which are not covered here.

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

- The X-12-ARIMA method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition
- the trend estimate is available for all observations including the end points, and the seasonal component is allowed to vary slowly over time.
- It is also relatively robust to occasional unusual observations.
- X-12-ARIMA handles both additive and multiplicative decomposition, but only allows for quarterly and monthly data.
- There is no loss of observations at the start and end of the series.
- X-12-ARIMA also has some sophisticated methods to handle trading day variation, holiday effects and the effects of known predictors, which are not covered here.
- Complete discussion of the method: Ladiray and Quenneville (2001).

X-12-ARIMA decomposition

One of the most popular methods for decomposing quarterly and monthly data is X-12-ARIMA, which has its origins in methods developed by the US Bureau of the Census. Earlier versions of the method included X-11 and X-11-ARIMA.

- The X-12-ARIMA method is based on classical decomposition, but with many extra steps and features to overcome the drawbacks of classical decomposition
- the trend estimate is available for all observations including the end points, and the seasonal component is allowed to vary slowly over time.
- It is also relatively robust to occasional unusual observations.
- X-12-ARIMA handles both additive and multiplicative decomposition, but only allows for quarterly and monthly data.
- There is no loss of observations at the start and end of the series.
- X-12-ARIMA also has some sophisticated methods to handle trading day variation, holiday effects and the effects of known predictors, which are not covered here.
- Complete discussion of the method: Ladiray and Quenneville (2001).
- There is currently no `R` package for X-12-ARIMA decomposition. However, free software that implements the method is available from the US Census Bureau and an `R` interface to that software is provided by the `x12` package.

STL decomposition

STL is an acronym for Seasonal and Trend decomposition using Loess, while Loess is a method for estimating nonlinear relationships. The STL method was developed by Cleveland et al. (1990).

STL decomposition

STL is an acronym for Seasonal and Trend decomposition using Loess, while Loess is a method for estimating nonlinear relationships. The STL method was developed by Cleveland et al. (1990). STL has several advantages:

- Unlike X-12-ARIMA, STL will handle any type of seasonality, not only monthly and quarterly data.

STL decomposition

STL is an acronym for Seasonal and Trend decomposition using Loess, while Loess is a method for estimating nonlinear relationships. The STL method was developed by Cleveland et al. (1990). STL has several advantages:

- Unlike X-12-ARIMA, STL will handle any type of seasonality, not only monthly and quarterly data.
- The seasonal component is allowed to change over time, and the rate of change can be controlled by the user.

STL decomposition

STL is an acronym for Seasonal and Trend decomposition using Loess, while Loess is a method for estimating nonlinear relationships. The STL method was developed by Cleveland et al. (1990). STL has several advantages:

- Unlike X-12-ARIMA, STL will handle any type of seasonality, not only monthly and quarterly data.
- The seasonal component is allowed to change over time, and the rate of change can be controlled by the user.
- The smoothness of the trend-cycle can also be controlled by the user.

STL decomposition

STL is an acronym for Seasonal and Trend decomposition using Loess, while Loess is a method for estimating nonlinear relationships. The STL method was developed by Cleveland et al. (1990). STL has several advantages:

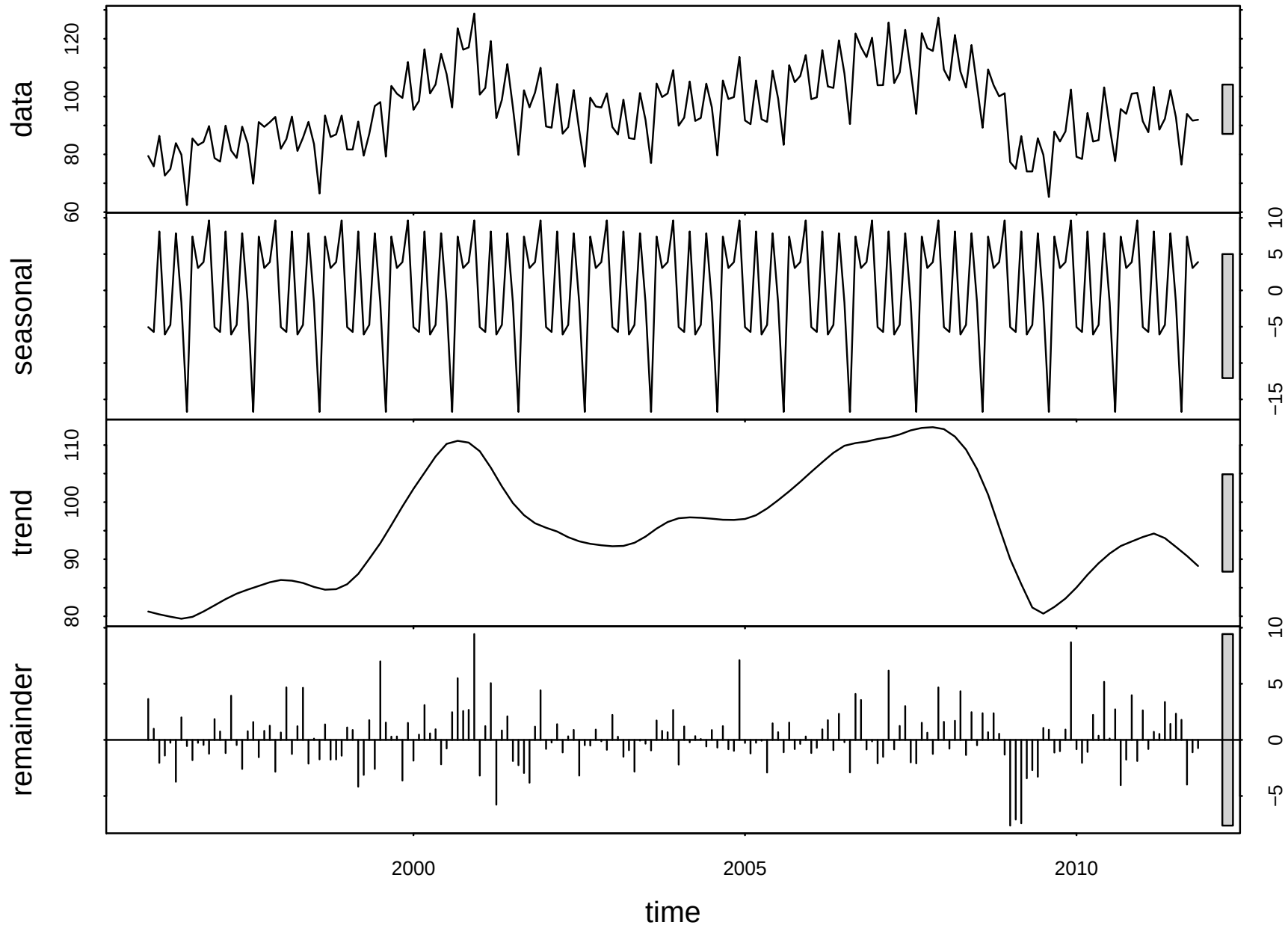
- Unlike X-12-ARIMA, STL will handle any type of seasonality, not only monthly and quarterly data.
- The seasonal component is allowed to change over time, and the rate of change can be controlled by the user.
- The smoothness of the trend-cycle can also be controlled by the user.
- It can be robust to outliers (i.e., the user can specify a robust decomposition). So occasional unusual observations will not affect the estimates of the trend-cycle and seasonal components.

R code

```
fit <- stl(elecequip, t.window = 15, s.window = "periodic", robust = TRUE)
plot(fit)
```

The two main parameters to be chosen when using STL are the trend window (t.window) and seasonal window (s.window). These control how rapidly the trend and seasonal components can change. Small values allow more rapid change. Setting the seasonal window to be infinite is equivalent to forcing the seasonal component to be periodic (i.e., identical across years).

STL outputs


data
seasonal
trend
remainder
60
80
100
120
−15
−5
0
5
10
80
90
100
110
−5
0
5
10
2000
2005
2010
time

Forecasting with decomposition

- Assuming an additive decomposition, the decomposed time series can be written as

$$y_t = \widehat{S}_t + \widehat{A}_t,$$

  where $\widehat{A}_t = \widehat{T}_t + \widehat{E}_t$ is the seasonally adjusted component.
- If a multiplicative decomposition has been used, we can write

$$y_t = \widehat{S}_t \widehat{A}_t,$$

  where $\widehat{A}_t = \widehat{T}_t \widehat{E}_t$.

Forecasting with decomposition

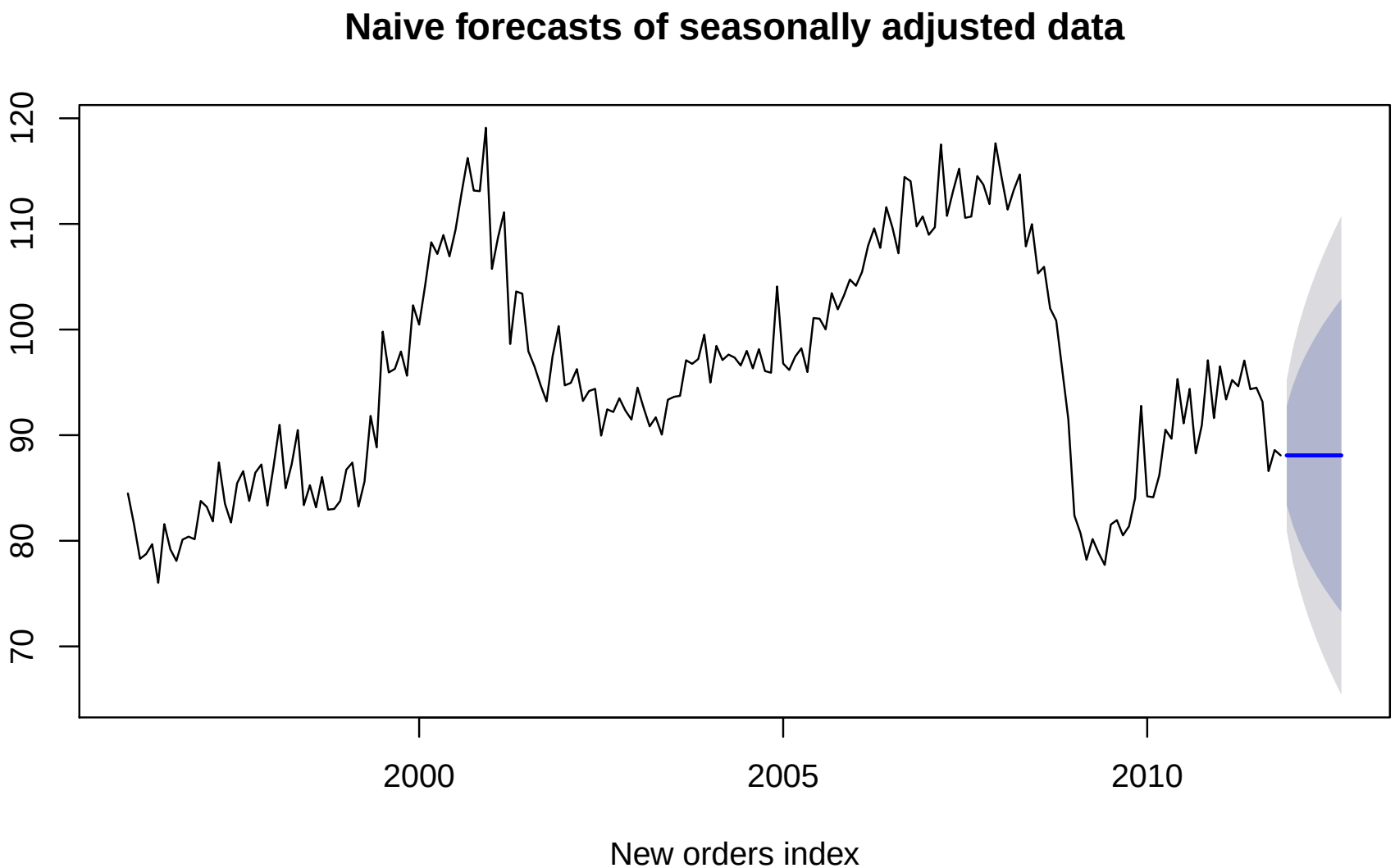
seasonal naive method

To forecast a decomposed time series, we separately forecast the seasonal component, $\widehat{S}_t$, and the seasonally adjusted component $\widehat{A}_t$. It is usually assumed that the seasonal component is unchanging, or changing extremely slowly, and so it is forecast by simply taking the last year of the estimated component.

To forecast the seasonally adjusted component, any non-seasonal forecasting method may be used. For example, a random walk with drift model, or Holt's method (discussed latter), or a non-seasonal ARIMA model (discussed later), may be used.

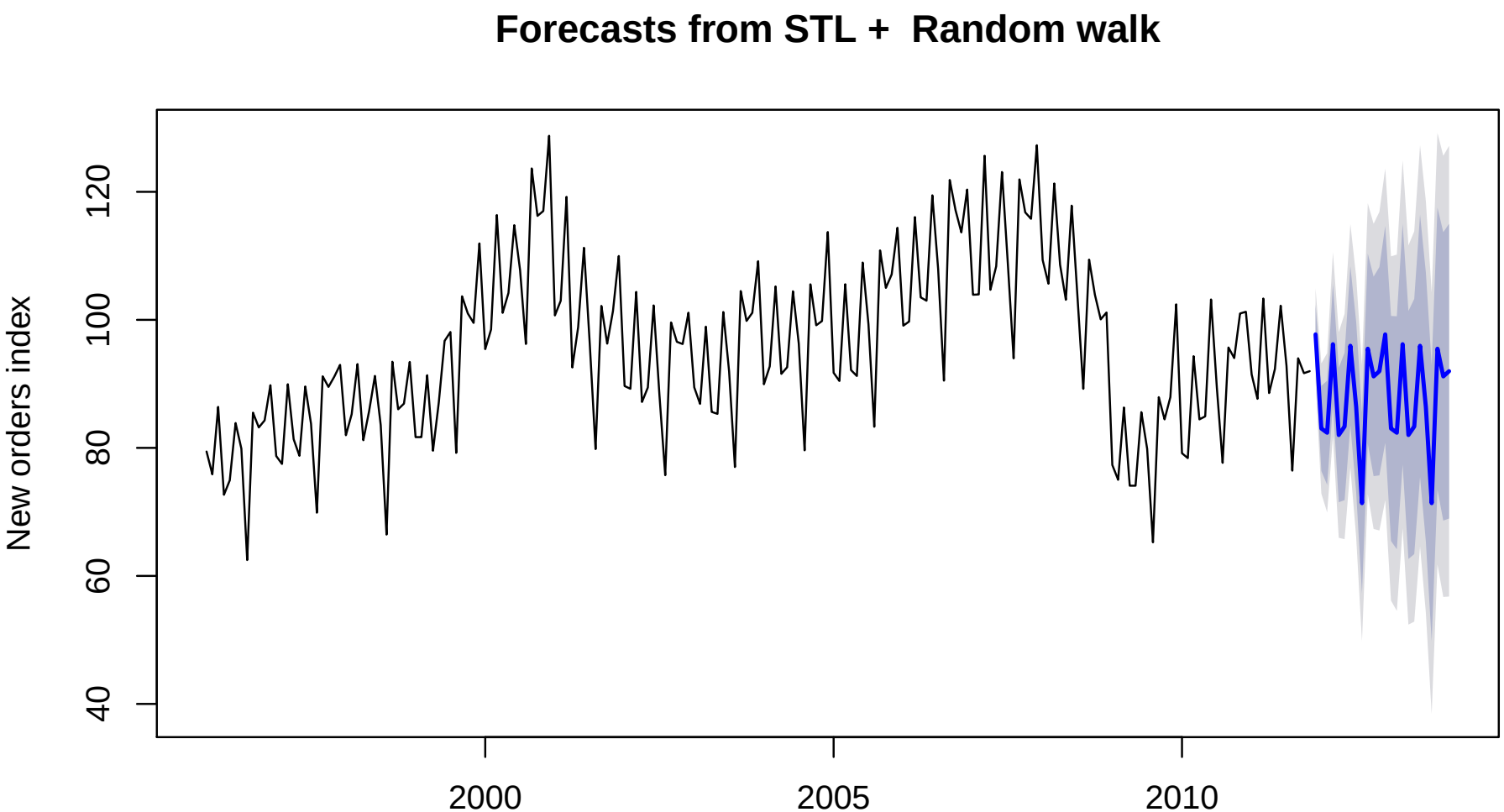
Electrical equipment manufacturing

```
fit <- stl(elecequip, t.window = 15, s.window = "periodic", robust = TRUE)
eeadj <- seasadj(fit)
plot(naive(eeadj), xlab = "New orders index", main = "Naive forecasts of seasonally a
```



Electrical equipment manufacturing

```
fcast <- forecast(fit, method = "naive")
plot(fcast, ylab = "New orders index")
```



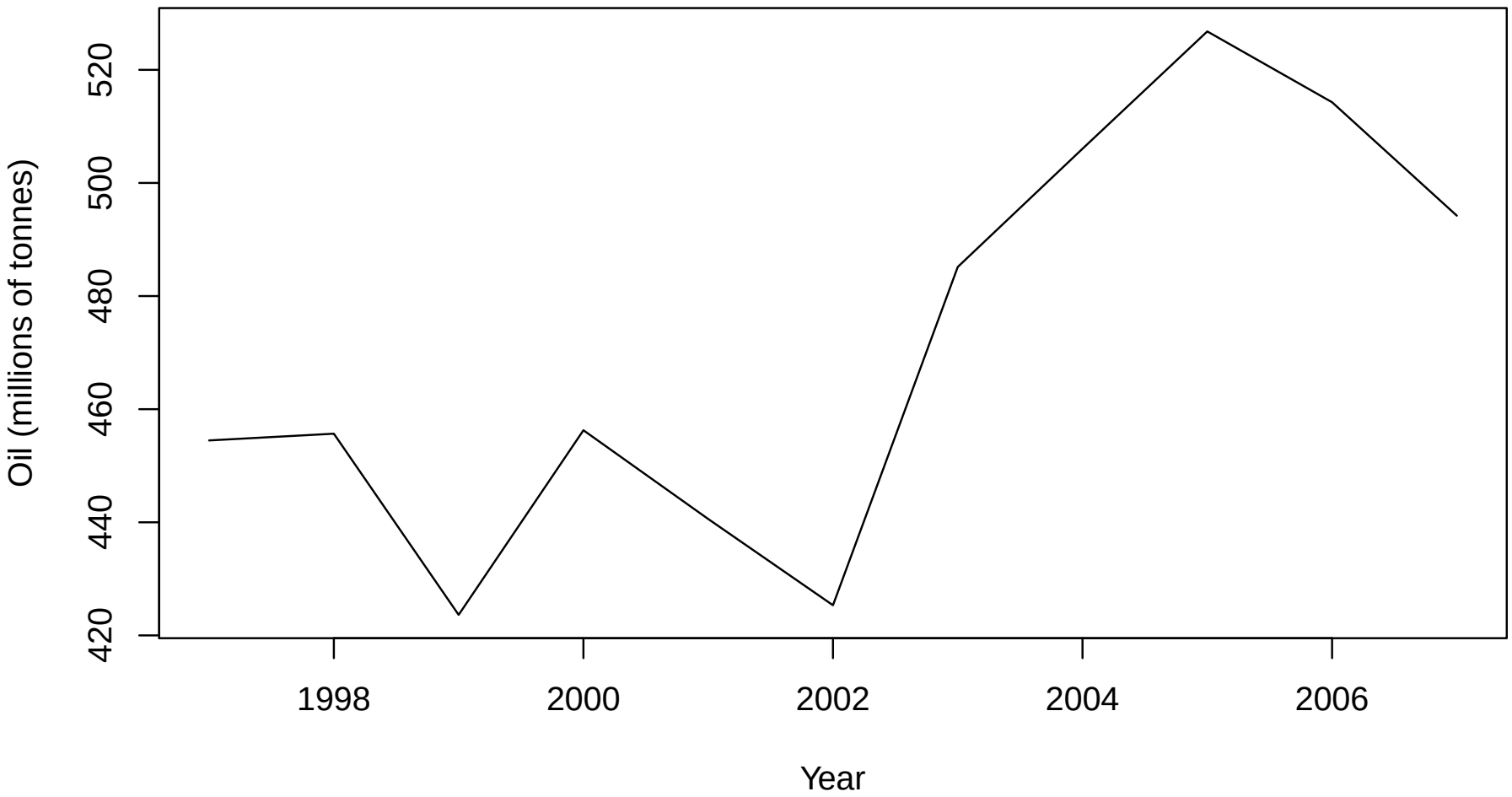
Exponential smoothing

- Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older.
- In other words, the more recent the observation the higher the associated weight.
- This framework generates reliable forecasts quickly and for a wide spectrum of time series which is a great advantage and of major importance to applications in industry.

Simple exponential smoothing

- This method is suitable for forecasting data with no trend or seasonal pattern, although the mean of the data may be changing slowly over time.
- The naive method assumes that the most current observation is the only important one and all previous observations provide no information for the future ($\widehat{y}_{T+h|T} = y_T$, for $h = 1, 2, \dots$). This can be thought of as a weighted average where all the weight is given to the last observation.
- The average method assumes that all observations are of equal importance and they are given equal weight when generating forecasts ($\widehat{y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} y_t$, for $h = 1, 2, \dots$).

Simple exponential smoothing



Simple exponential smoothing

We often want something between these two extremes. For example it may be sensible to attach larger weights to more recent observations than to observations from the distant past.

This is exactly the concept behind simple exponential smoothing.

Forecasts are calculated using weighted averages where the weights decrease exponentially as observations come from further in the past. The smallest weights are associated with the oldest observations:

$$\widehat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots,$$

where $0 \leqslant \alpha \leqslant 1$ is the smoothing parameter. The one-step-ahead forecast for time $T+1$ is a weighted average of all the observations in the series $y_1, \ldots, y_T$. The rate at which the weights decrease is controlled by the parameter α.

Simple exponential smoothing

The one-step-ahead forecast for time $T + 1$ is a weighted average of all the observations in the series $y_1, \ldots, y_T$. The rate at which the weights decrease is controlled by the parameter α.

Note that the sum of the weights even for a small α will be approximately equal to one.

For any α between 0 and 1, the weights attached to lagged observations decrease exponentially. If α is small (i.e., close to 0), more weight is given to observations from the more distant past. If α is large (i.e., close to 1), more weight is given to the more recent observations. At the extreme case where $\alpha = 1$, $\widehat{y}_{T+1|T} = y_T$ and

forecasts are equal to the naive forecasts.

Weighted average form

The forecast at time $t+1$ is equal to a weighted average between the most recent observation y_t and the most recent forecast $\widehat{y}_{t|t-1}$,

$$\widehat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\widehat{y}_{t|t-1}$$

for $t = 1, \ldots, T$, where $0 \leqslant \alpha \leqslant 1$ is the smoothing parameter. The process has to start somewhere, so we let the first forecast of y_1 be denoted by ℓ_0. Then

$$\begin{aligned}
\widehat{y}_{2|1} &= \alpha y_1 + (1-\alpha)\ell_0 \\
\widehat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\widehat{y}_{2|1} \\
\widehat{y}_{T+1|T} &= \alpha y_T + (1-\alpha)\widehat{y}_{T|T-1}
\end{aligned}$$

Weighted average form

Then substituting each equation into the following equation, we obtain

$$\begin{aligned}
\widehat{y}_{3|2} &= \alpha y_2 + (1-\alpha)\left[\alpha y_1 + (1-\alpha)\ell_0\right] \\
&= \alpha y_2 + \alpha(1-\alpha)y_1 + (1-\alpha)^2\ell_0 \\
\widehat{y}_{4|3} &= \alpha y_3 + (1-\alpha)[\alpha y_2 + \alpha(1-\alpha)y_1 + (1-\alpha)^2\ell_0] \\
&= \alpha y_3 + \alpha(1-\alpha)y_2 + \alpha(1-\alpha)^2 y_1 + (1-\alpha)^3\ell_0 \\
&\vdots
\end{aligned}$$

$$\widehat{y}_{T+1|T} = \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T \ell_0.$$

Component form

For simple exponential smoothing the only component included is the level, ℓ_t. (Other methods considered later in this chapter may also include a trend b_t and seasonal component s_t.)

Component form representations of exponential smoothing methods comprise a forecast equation and a smoothing equation for each of the components included in the method:

Forecast equation	$\widehat{y}_{t+1\|t} = \ell_t$
Smoothing equation	$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1},$

where ℓ_t is the level (or the smoothed value) of the series at time t.

The forecast equation shows that the forecasted value at time $t+1$ is the estimated level at time t. The smoothing equation for the level (usually referred to as the level equation) gives the estimated level of the series at each period t.

Applying the forecast equation for time T gives, $\widehat{y}_{T+1|T} = \ell_T$, the most recent estimated level.

If we replace ℓ_t by $\widehat{y}_{t+1|t}$ and ℓ_{t-1} by $\widehat{y}_{t|t-1}$ in the smoothing equation, we will recover the weighted average form of simple exponential smoothing.

Error correction form

The third form of simple exponential smoothing is obtained by re-arranging the level equation in the component form to get what we refer to as the error correction form

$$\begin{aligned}\ell_t &= \ell_{t-1} + \alpha(y_t - \ell_{t-1}) \\ &= \ell_{t-1} + \alpha e_t\end{aligned}$$

where $e_t = y_t - \ell_{t-1} = y_t - \widehat{y}_{t|t-1}$ for $t = 1, \ldots, T$. That is, e_t is the one-step

within-sample forecast error at time t. The within-sample forecast errors lead to

the adjustment/correction of the estimated level throughout the smoothing process for $t = 1, \ldots, T$.

For example, if the error at time t is negative, then $\widehat{y}_{t|t-1} > y_t$ and so the level at time $t-1$ has been over-estimated. The new level ℓ_t is then the previous level ℓ_{t-1} adjusted downwards.

The closer α is to one the rougher the estimate of the level (large adjustments take place).

The smaller the α the smoother the level (small adjustments take place).

Multi-horizon Forecasts

Simple exponential smoothing has a flat forecast function, and therefore for longer forecast horizons,

$$\widehat{y}_{T+h|T} = \widehat{y}_{T+1|T} = \ell_T, \qquad h = 2, 3, \ldots.$$

Remember these forecasts will only be suitable if the time series has no trend or seasonal component.

Initialisation

For simple exponential smoothing we need to specify an initial value for the level, ℓ_0. Hence ℓ_0 plays a role in all forecasts generated by the process. In general, the weight attached to ℓ_0 is small.

However, in the case that α is small and/or the time series is relatively short, the weight may be large enough to have a noticeable effect on the resulting forecasts. Therefore, selecting suitable initial values can be quite important.

A common approach is to set $\ell_0 = y_1$ (recall that $\ell_0 = \widehat{y}_{1|0}$).

An alternative approach is to use optimization to estimate the value of ℓ_0 rather than set it to some value. Even if optimization is used, selecting appropriate initial values can assist the speed and precision of the optimization process.

Optimization

For every exponential smoothing method we also need to choose the value for the smoothing parameters. For simple exponential smoothing, there is only one smoothing parameter (α).

There are cases where the smoothing parameters may be chosen in a subjective manne. The forecaster specifies the value of the smoothing parameters based on previous experience.

However, a more robust and objective way to obtain values for the unknown parameters included in any exponential smoothing method is to estimate them from the observed data.

Similarly to a regression model, the unknown parameters and the initial values for any exponential smoothing method can be estimated by minimizing the SSE.

The errors are specified as $e_t = y_t - \widehat{y}_{t|t-1}$ for $t = 1, \ldots, T$ (the one-step-ahead within-sample forecast errors).

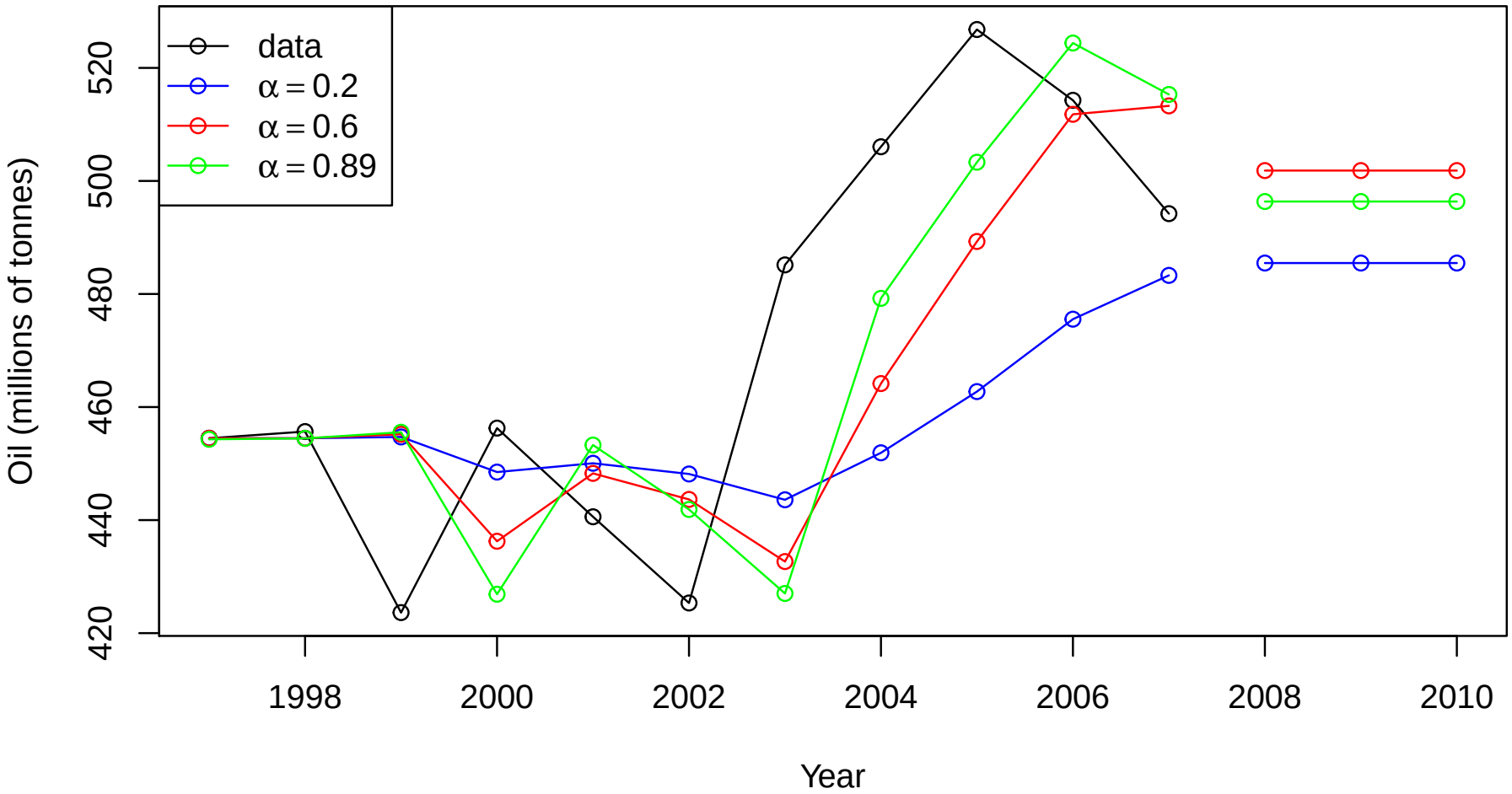
$$\text{SSE} = \sum_{t=1}^{T} (y_t - \widehat{y}_{t|t-1})^2 = \sum_{t=1}^{T} e_t^2.$$

This involves a non-linear minimization problem and we need to use an optimization tool to perform this.

Simple exponential smoothing

```
fit1 <- ses(oildata, alpha = 0.2, initial = "simple", h = 3)
fit2 <- ses(oildata, alpha = 0.6, initial = "simple", h = 3)
fit3 <- ses(oildata, h = 3)
plot(fit1, plot.conf = FALSE, ylab = "Oil (millions of tonnes)",
    xlab = "Year", main = "", fcol = "white", type = "o")
lines(fitted(fit1), col = "blue", type = "o")
lines(fitted(fit2), col = "red", type = "o")
lines(fitted(fit3), col = "green", type = "o")
lines(fit1$mean, col = "blue", type = "o")
lines(fit2$mean, col = "red", type = "o")
lines(fit3$mean, col = "green", type = "o")
legend("topleft", lty = 1, col = c(1, "blue", "red", "green"),
    c("data", expression(alpha == 0.2), expression(alpha == 0.6),
        expression(alpha == 0.89)), pch = 1)
```

Simple exponential smoothing



Holt's linear trend method

Extended simple exponential smoothing to allow forecasting of data with a trend. This method involves a forecast equation and two smoothing equations (one for the level and one for the trend):

$$\begin{aligned}
\text{Forecast equation} \quad & \widehat{y}_{t+h|t} = \ell_t + hb_t \\
\text{Level equation} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
\text{Trend equation} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}
\end{aligned}$$

where ℓ_t denotes an estimate of the level of the series at time t, b_t denotes an estimate of the trend (slope) of the series at time t, α is the smoothing parameter for the level, $0 \leqslant \alpha \leqslant 1$ and β^* is the smoothing parameter for the trend, $0 \leqslant \beta^* \leqslant 1$ (we denote this as β^* instead of β.

As with simple exponential smoothing, the level equation here shows that ℓ_t is a weighted average of observation y_t and the within-sample one-step-ahead forecast for time t, here given by $\ell_{t-1} + b_{t-1}$.

The trend equation shows that b_t is a weighted average of the estimated trend at time t based on $\ell_t - \ell_{t-1}$ and b_{t-1}, the previous estimate of the trend.

The forecast function is no longer flat but trending. The h-step-ahead forecast is equal to the last estimated level plus h times the last estimated trend value. Hence the forecasts are a linear function of h.

The error correction form of the level and the trend equations show the adjustments in terms of the within-sample one-step forecast errors:

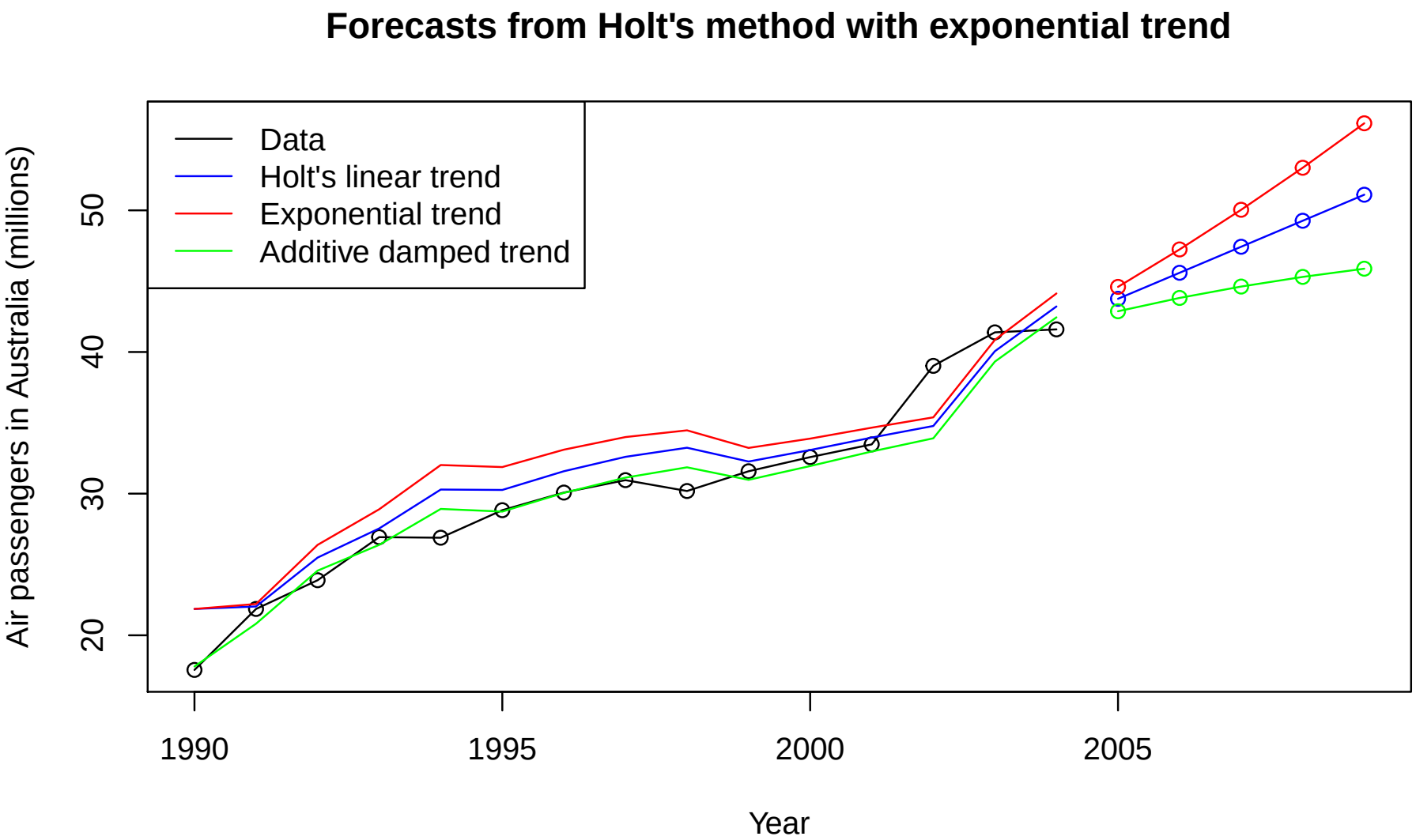
$$\ell_t = \ell_{t-1} + b_{t-1} + \alpha e_t$$
$$b_t = b_{t-1} + \alpha\beta^* e_t$$

where $e_t = y_t - (\ell_{t-1} + b_{t-1}) = y_t - \widehat{y}_{t|t-1}$.

Holt's linear trend method

```
air <- window(ausair, start = 1990, end = 2004)
fit1 <- holt(air, alpha = 0.8, beta = 0.2, initial = "simple",
    h = 5)
fit2 <- holt(air, alpha = 0.8, beta = 0.2, initial = "simple",
    exponential = TRUE, h = 5)
# Results for first model:
fit1$model$state
fitted(fit1)
fit1$mean
fit3 <- holt(air, alpha = 0.8, beta = 0.2, damped = TRUE, initial = "simple",
    h = 5)
plot(fit2, type = "o", ylab = "Air passengers in Australia (millions)",
    xlab = "Year", fcol = "white", plot.conf = FALSE)
lines(fitted(fit1), col = "blue")
lines(fitted(fit2), col = "red")
lines(fitted(fit3), col = "green")
lines(fit1$mean, col = "blue", type = "o")
lines(fit2$mean, col = "red", type = "o")
lines(fit3$mean, col = "green", type = "o")
legend("topleft", lty = 1, col = c("black", "blue", "red", "green"),
    c("Data", "Holt's linear trend", "Exponential trend", "Additive damped trend"))
```

Holt's linear trend method



Exponential trend method

A variation from Holt's linear trend method is achieved by allowing the level and the slope to be multiplied rather than added:

$$\begin{aligned}\widehat{y}_{t+h|t} &= \ell_t b_t^h \\ \ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} b_{t-1}) \\ b_t &= \beta^* \frac{\ell_t}{\ell_{t-1}} + (1-\beta^*) b_{t-1}\end{aligned}$$

where b_t now represents an estimated growth rate (in relative terms rather than absolute) which is multiplied rather than added to the estimated level.

The trend in the forecast function is now exponential rather than linear, so that the forecasts project a constant growth rate rather than a constant slope.

Damped trend methods

The error correction form is

$$\ell_t = \ell_{t-1} b_{t-1} + \alpha e_t$$
$$b_t = b_{t-1} + \alpha\beta^* \frac{e_t}{\ell_{t-1}}$$

where $e_t = y_t - (\ell_{t-1} b_{t-1}) = y_t - \widehat{y}_{t|t-1}$.

Damped trend methods

The forecasts generated by Holt's linear method display a constant trend (increasing or decreasing) indefinitely into the future. Even more extreme are the forecasts generated by the exponential trend method which include exponential growth or decline. Empirical evidence indicates that these methods tend to

over-forecast, especially for longer forecast horizons. Motivated by this, a *dampening* parameter is introduced so that the trend approaches a flat line some time in the future.

Additive damped trend

In conjunction with the smoothing parameters α and β^* (with values between 0 and 1 as in Holt's method), this method also includes a damping parameter $0 < \phi < 1$:

$$\begin{aligned}\widehat{y}_{t+h|t} &= \ell_t + (\phi + \phi^2 + \cdots + \phi^h)b_t \\ \ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}) \\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)\phi b_{t-1}.\end{aligned}$$

If $\phi = 1$ the method is identical to HoltâĂŹs linear method. For values between 0 and 1, ϕ dampens the trend so that it approaches a constant some time in the future. In fact the forecasts converge to $\ell_T + \phi b_T/(1-\phi)$ as $h \to \infty$ for any value

$0 < \phi < 1$. The effect of this is that short-run forecasts are trended while long-run forecasts are constant.

Multiplicative damped trend

Motivated by the improvements in forecasting performance seen in the additive damped trend case, this method introduces a damping parameter to the exponential trend method:

$$\begin{aligned}
\widehat{y}_{t+h|t} &= \ell_t b_t^{(\phi+\phi^2+\cdots+\phi^h)} \\
\ell_t &= \alpha y_t + (1-\alpha)\ell_{t-1} b_{t-1}^{\phi} \\
b_t &= \beta^* \frac{\ell_t}{\ell_{t-1}} + (1-\beta^*) b_{t-1}^{\phi}.
\end{aligned}$$

This method will produce even more conservative forecasts than the additive damped trend method when compared to Holt's linear method.

The error correction form of the additive damped trend smoothing equations is

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha e_t$$
$$b_t = \phi b_{t-1} + \alpha\beta^* e_t.$$

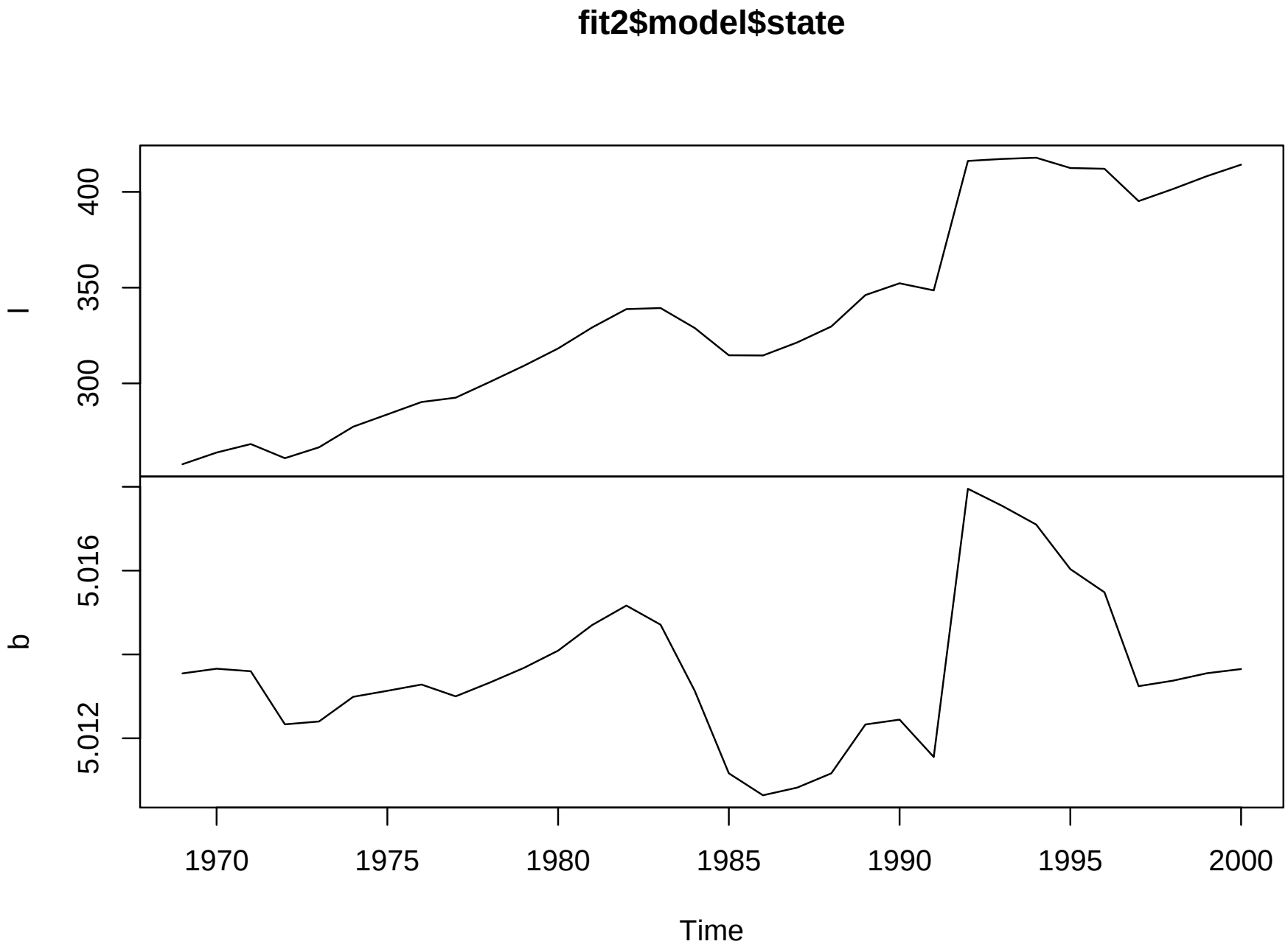
The error correction form of the multiplicative damped trend smoothing equations is

$$\ell_t = \ell_{t-1} b_{t-1}^{\phi} + \alpha e_t$$
$$b_t = b_{t-1}^{\phi} + \alpha\beta^* \frac{e_t}{\ell_{t-1}}.$$

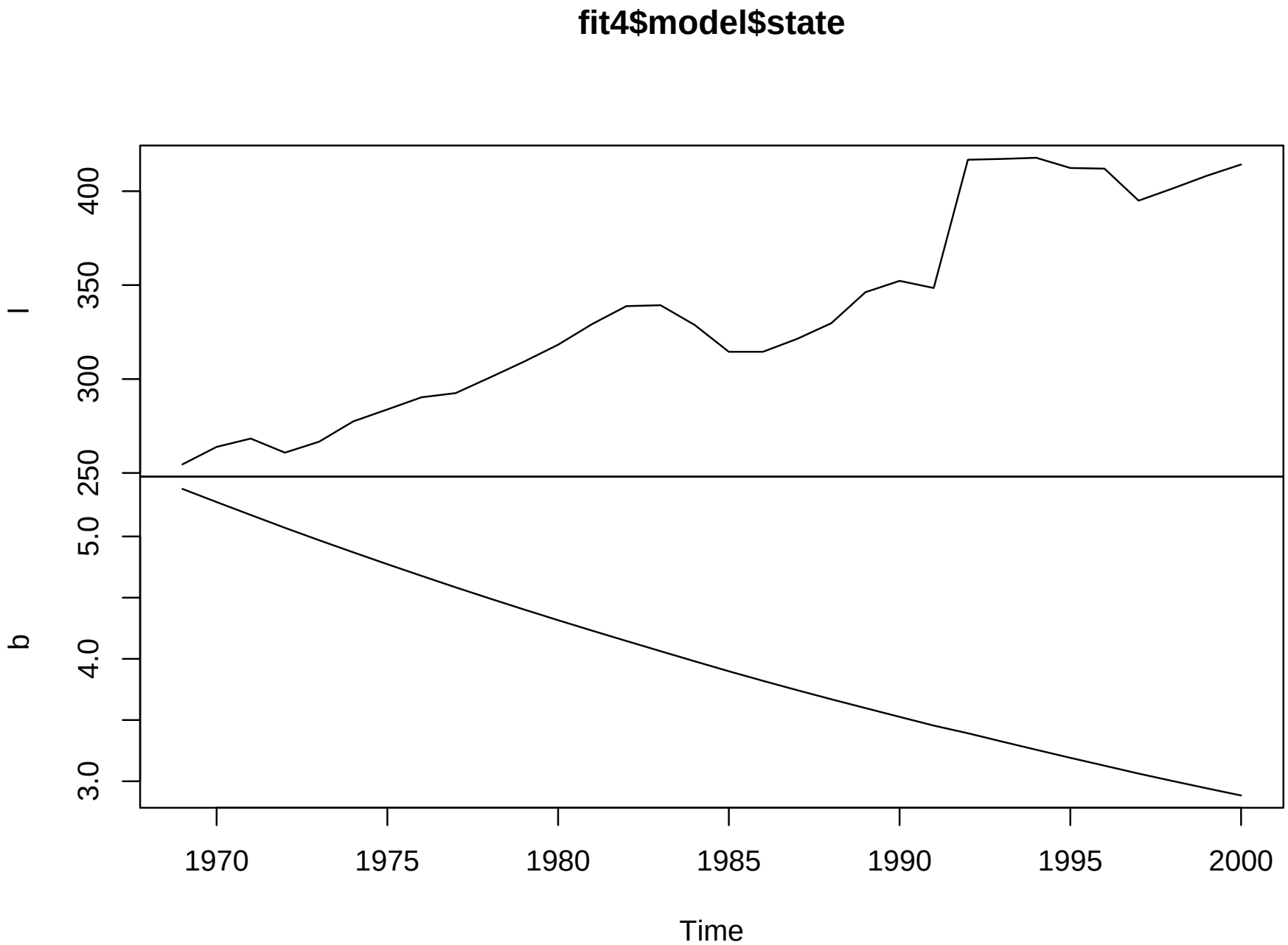
Example: Sheep in Asia

```
livestock2 <- window(livestock, start = 1970, end = 2000)
fit1 <- ses(livestock2)
fit2 <- holt(livestock2)
fit3 <- holt(livestock2, exponential = TRUE)
fit4 <- holt(livestock2, damped = TRUE)
fit5 <- holt(livestock2, exponential = TRUE, damped = TRUE)
# Results for first model: fit1$model accuracy(fit1) #
# training set accuracy(fit1,livestock) # test set
# plot(fit2$model$state) plot(fit4$model$state)
```

Example: Sheep in Asia



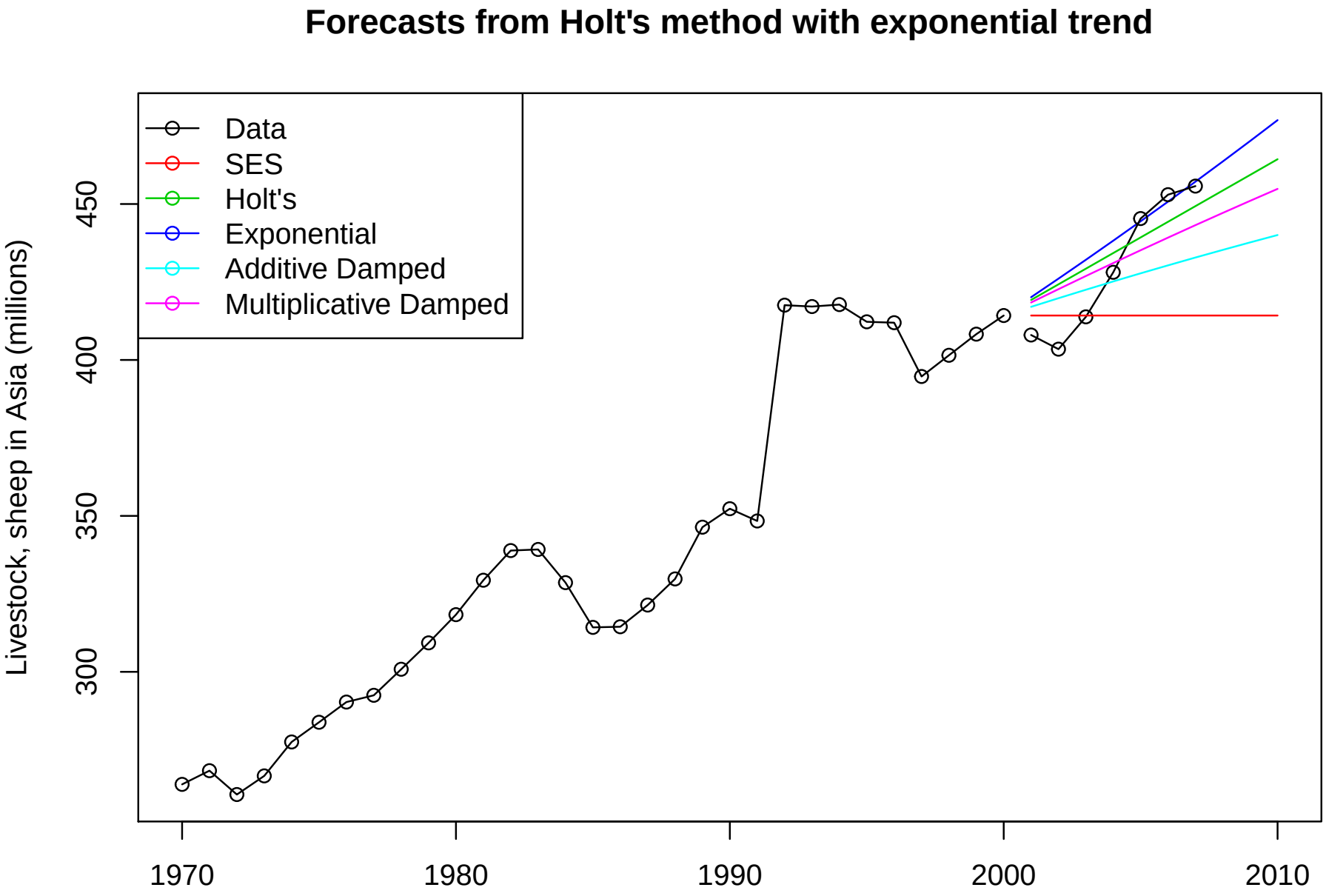
Example: Sheep in Asia



Forecats: Sheep in Asia

```
plot(fit3, type = "o", ylab = "Livestock, sheep in Asia (millions)",
    flwd = 1, plot.conf = FALSE)
lines(window(livestock, start = 2001), type = "o")
lines(fit1$mean, col = 2)
lines(fit2$mean, col = 3)
lines(fit4$mean, col = 5)
lines(fit5$mean, col = 6)
legend("topleft", lty = 1, pch = 1, col = 1:6, c("Data", "SES",
    "Holt's", "Exponential", "Additive Damped", "Multiplicative Damped"))
```

Forecats: Sheep in Asia



Holt-Winters seasonal method

The Holt-Winters seasonal method consists of the forecast equation and three smoothing equations: one for the level ℓ_t, one for trend b_t, and one for the seasonal component denoted by s_t, with smoothing parameters α, β^* and γ.

We use m to denote the period of the seasonality, i.e., the number of seasons in a year.

There are two variations to this method that differ in the nature of the seasonal component. The additive method is preferred when the seasonal variations are roughly constant through the series, while the multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series.

Holt-Winters additive method

The component form for the additive method is:

$$\begin{aligned}
\widehat{y}_{t+h|t} &= \ell_t + hb_t + s_{t-m+h_m^+} \\
\ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},
\end{aligned}$$

where $h_m^+ = \lfloor (h-1) \mod m \rfloor + 1$, which ensures that the estimates of the seasonal indices used for forecasting come from the final year of the sample. (The notation $\lfloor u \rfloor$ means the largest integer not greater than u.)

The level equation shows a weighted average between the seasonally adjusted observation $(y_t - s_{t-m})$ and the non-seasonal forecast $(\ell_{t-1} + b_{t-1})$ for time t. The trend equation is identical to Holt's linear method.

The seasonal equation shows a weighted average between the current seasonal index, $(y_t - \ell_{t-1} - b_{t-1})$, and the seasonal index of the same season last year (i.e., m time periods ago).

The error correction form of the smoothing equations is:

$$\begin{aligned}
\ell_t &= \ell_{t-1} + b_{t-1} + \alpha e_t \\
b_t &= b_{t-1} + \alpha\beta^* e_t \\
s_t &= s_{t-m} + \gamma e_t.
\end{aligned}$$

where $e_t = y_t - (\ell_{t-1} + b_{t-1} + s_{t-m}) = y_t - \widehat{y}_{t|t-1}$ are the one-step training forecast errors.

Holt-Winters multiplicative method

The component form for the multiplicative method is:

$$\widehat{y}_{t+h|t} = (\ell_t + hb_t)s_{t-m+h_m^+}.$$
$$\ell_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})$$
$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$$
$$s_t = \gamma\frac{y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m}$$

and the error correction representation is:

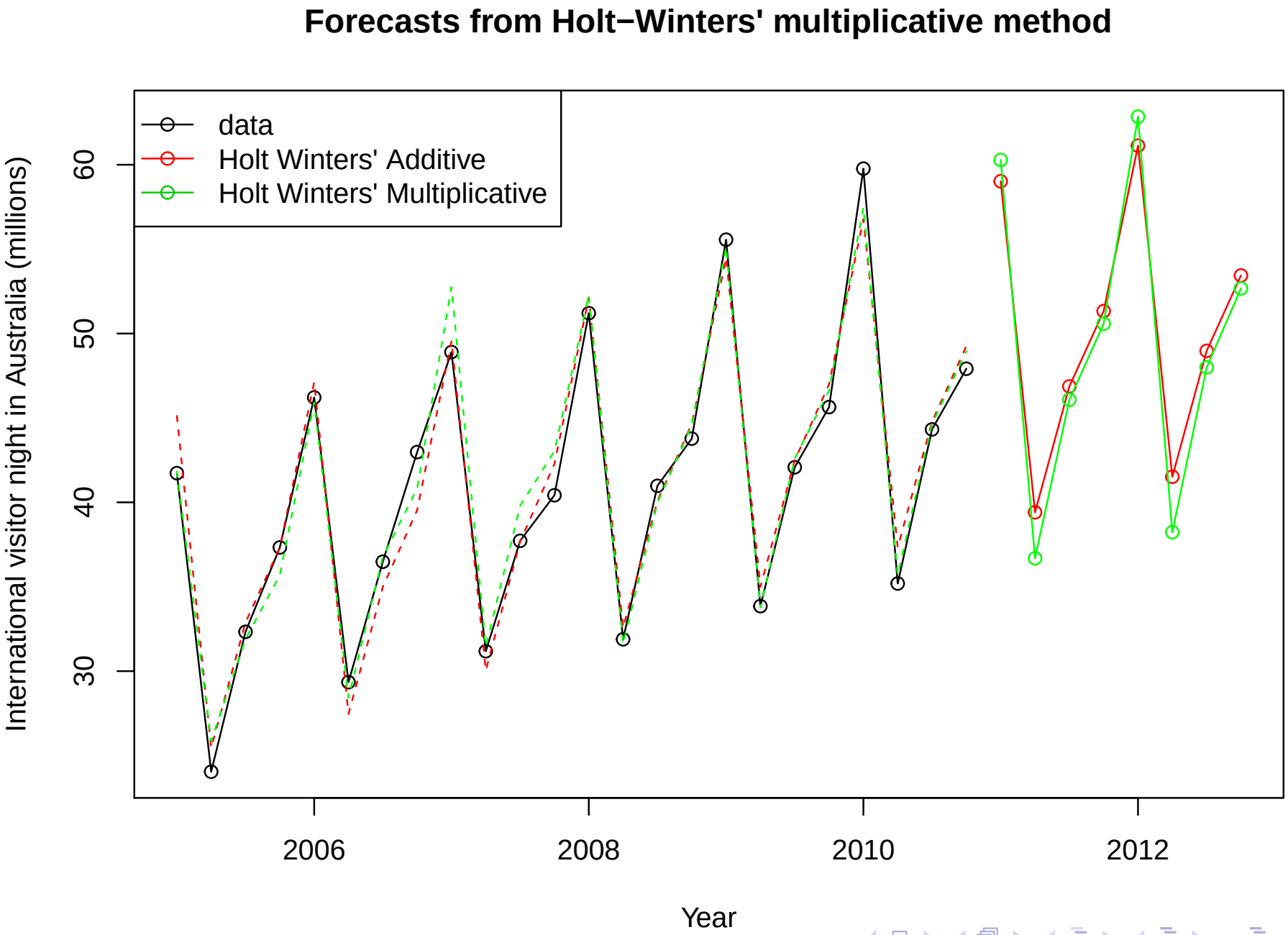
$$\ell_t = \ell_{t-1} + b_{t-1} + \alpha\frac{e_t}{s_{t-m}}$$
$$b_t = b_{t-1} + \alpha\beta^*\frac{e_t}{s_{t-m}}$$
$$s_t = s_t + \gamma\frac{e_t}{(\ell_{t-1} + b_{t-1})}$$

where $e_t = y_t - (\ell_{t-1} + b_{t-1})s_{t-m}$.

International tourist visitor nights in Australia

```
aust <- window(austourists, start = 2005)
fit1 <- hw(aust, seasonal = "additive")
fit2 <- hw(aust, seasonal = "multiplicative")
plot(fit2, ylab = "International visitor night in Australia (millions)",
    plot.conf = FALSE, type = "o", fcol = "white", xlab = "Year")
lines(fitted(fit1), col = "red", lty = 2)
lines(fitted(fit2), col = "green", lty = 2)
lines(fit1$mean, type = "o", col = "red")
lines(fit2$mean, type = "o", col = "green")
legend("topleft", lty = 1, pch = 1, col = 1:3, c("data", "Holt Winters' Additive",
    "Holt Winters' Multiplicative"))
```

International tourist visitor nights in Australia



Holt-Winters method with additive and multiplicative seasonal components.

```
states <- cbind(fit1$model$states[, 1:3], fit2$model$states[,
    1:3])
colnames(states) <- c("level", "slope", "seasonal", "level",
    "slope", "seasonal")
plot(states, xlab = "Year")
fit1$model$state[, 1:3]
fitted(fit1)
fit1$mean
```

Holt-Winters method with additive and multiplicative seasonal components.