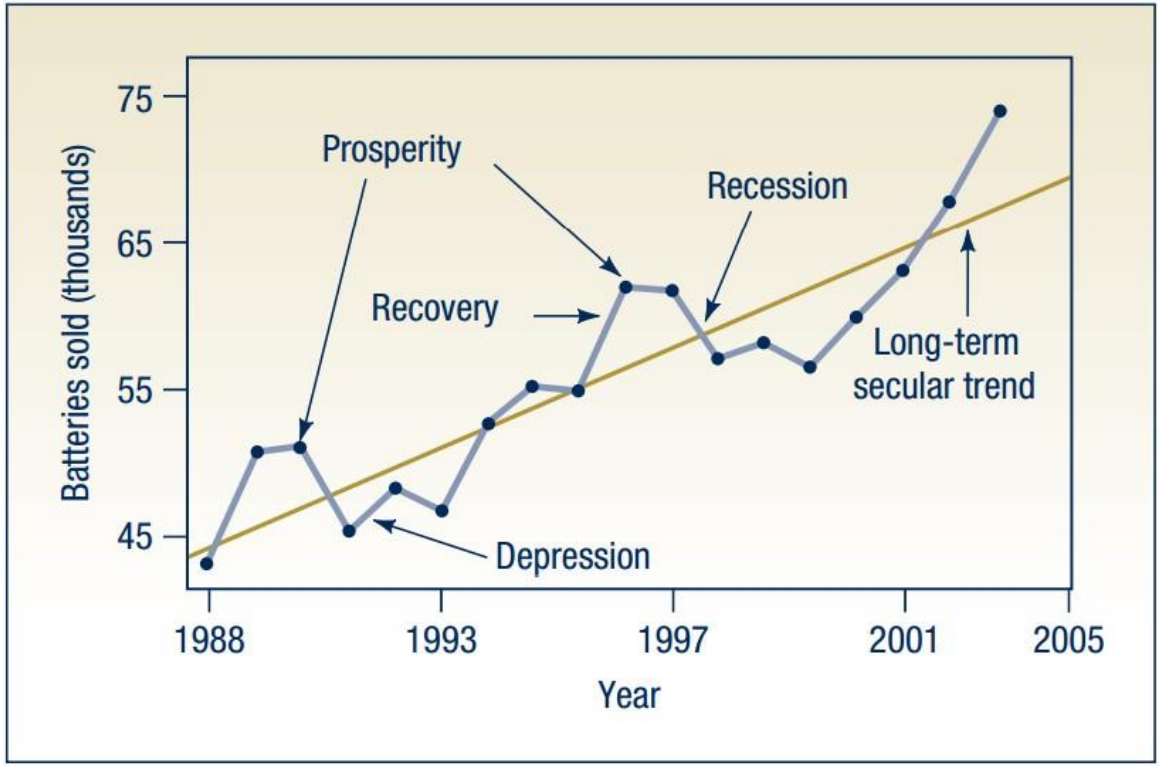
Time Series Forecasting in R

Quick Review

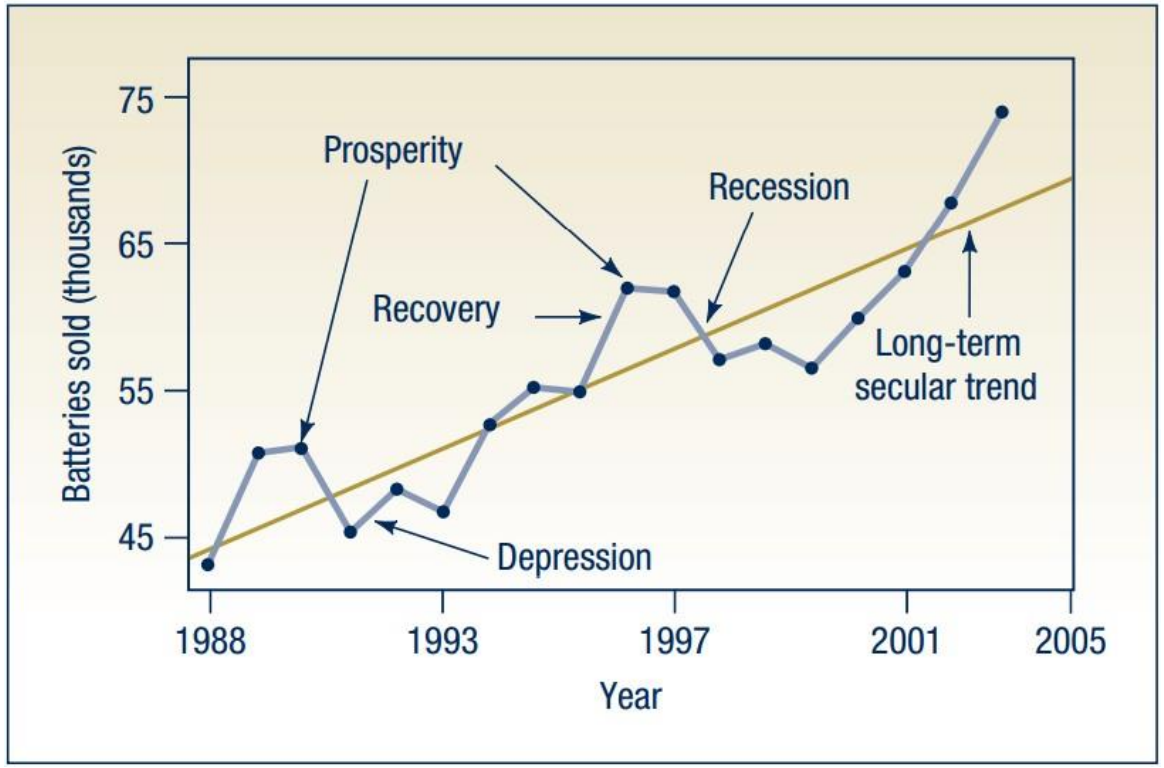
- A time series is a collection of observations made sequentially in time.



- Battery Sales by National Battery Sales, Inc., 1988–2005

Short Review

- Forecasting is making a statement about the future



- Forecast for 2006?
- How about for 2007?
- For 2008?
- Then for 2017?

- Battery Sales by National Battery Sales, Inc., 1988–2005

Quantitative

- Time-series Forecasts
  - Uses historical data assuming that the future is simply a repetition of the past

  - Let y_{t+1} be the forecast using the k past periods
  - Y = revenues, sales, expenses, losses, costs, employees hired, etc.

Quantitative

- The pattern or behavior of the data in a time series has several components.
- Theoretically, any time series can be decomposed into:
  - Trend
  - Cyclical
  - Seasonal
  - Irregular
- However, this decomposition is often not straight- forward because these factors interact.

Outline for this Session

- Moving Averages
- Exponential Smoothing Methods
- Decomposition in R

Methods of Smoothing Time Series

- Smoothing a time series: to eliminate some of short-term fluctuations or effect due to random variation
- Smoothing also can be done to remove seasonal fluctuations, i.e., to de-seasonalize a time series.
  - Arithmetic Moving Average
  - Exponential Smoothing Methods
  - Holt-Winters method for Exponential Smoothing
- These models are deterministic in that no reference is made to the sources or nature of the underlying randomness in the series.
- The models involves extrapolation techniques.

Averaging Methods

- Simple Averages - quick, inexpensive
- Moving Average method consists of computing an average of the most recent n data values for the series and using this average for forecasting the value of the time series for the next period.
- Recall Simple Moving Average (SMA)

$$\hat{y}_{t+1} = \frac{y_t + y_{t-1} + \cdots + y_{t-n}}{n}$$

Moving Averages

- Moving averages are useful if one can assume item to be forecast will stay steady over time.
- Series of arithmetic means – used only for smoothing, provides overall impression of data over time
  - The smaller the number, the more weight given to recent periods. This is desirable when there are sudden shifts in the level of the series.
  - The greater the number, less weight is given to more recent periods and the greater the smoothing effect.

Moving Averages

(WMA)

- Weighted Moving Average - place more weight on recent observations. Sum of the weights needs to equal 1.
- Used when trend is present
  - Older data usually less important

$$\hat{y}_{t+1} = \frac{w_1 y_t + w_2 y_{t-1} + \cdots + w_n y_{t-n}}{n}$$

Births

- An example is a data set of the number of births per month in New York city, from January 1946 to December 1959
- Forecast using Simple Moving Average with periods 2, 5 and 10

$$\hat{y}_{t+1} = \frac{w_1 y_t + w_2 y_{t-1} + \cdots + w_n y_{t-n}}{n}$$

Preliminary: ts

- Class ts
- Used to create “time series objects”
- Represents data which has been sampled at equally spaced points in time
- By default, frequencies can be 4, 7, and 12
- A weekly, monthly and quarterly series

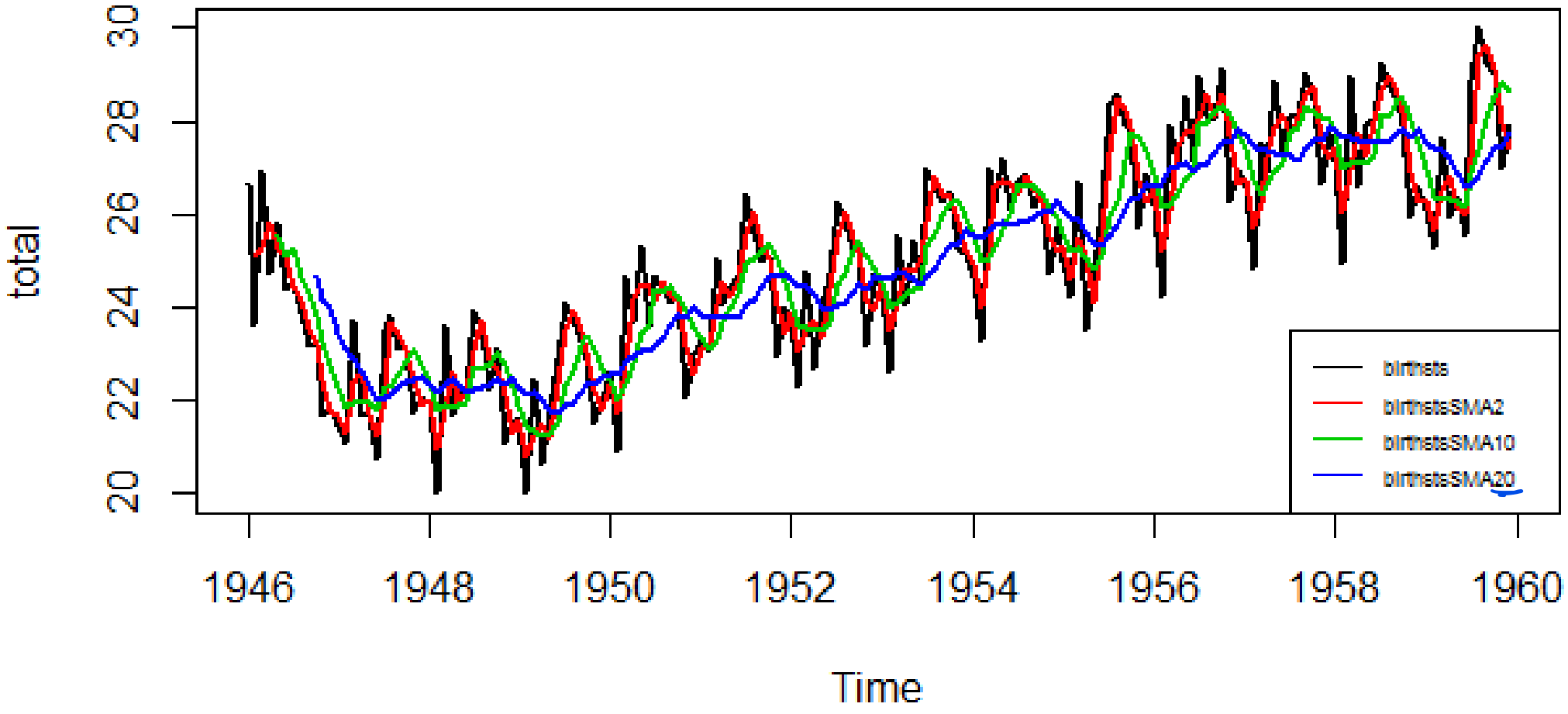
Preliminary: ts

- `install.packages("TTR")`
- `install.packages("forecast")`
- `library("TTR")`
- `a <- ts(1:20, frequency=12, start=c(2011,3))`
- `print(a)`
- `str(a)`
- `attributes(a)`

R Code

```
library("TTR")
births = read.csv("births.csv")
birthsts = ts(births[,2], frequency=12, start=c(1946,1))
birthstsSMA2 = SMA(birthsts,n=2)
birthstsSMA5 = SMA(birthsts,n=5)
birthstsSMA10 = SMA(birthsts,n=10)
total = cbind(birthsts,birthstsSMA2,birthstsSMA5,birthstsSMA10)
plot(total, plot.type="single",col = 1:ncol(total), lwd = c(2, 2, 2,2))
legend("bottomright", colnames(total), col=1:ncol(total), lty = c(1, 1, 1,1), cex=.5, y.intersp = 1)
```

Births Dataset

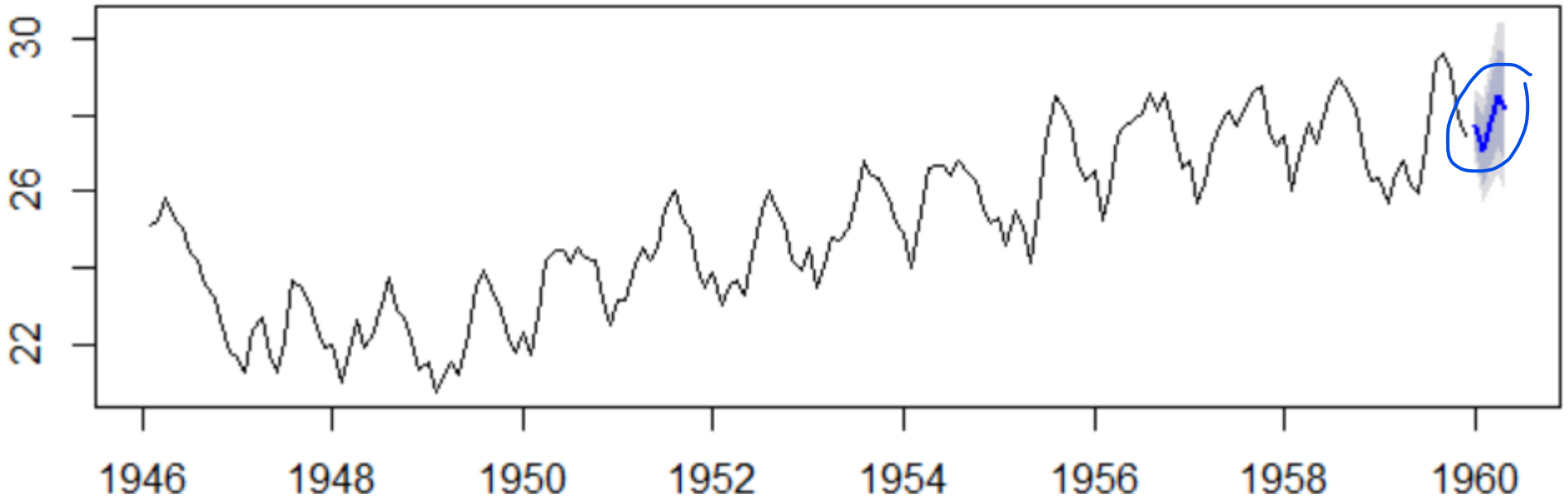


R Code: Forecast using SMA

- `#To forecast using the same model`
- `library("forecast")`
- `birthstsSMA2`
- `birthstsSMA2forecast = forecast(birthstsSMA2, h=5)`
- `birthstsSMA2forecast`
- `plot(birthstsSMA2forecast)`

- *CAUTION: SMA function warns that it will assume an Exponential Smoothing State (ETS) if time period to be forecasted is greater than 1.*

Births Dataset



R Code: Forecast Accuracy

- `accuracy(birthstsSMA2, birthsts)`

forecast

actual

```
> accuracy(birthstsSMA2, birthsts)
                     ME      RMSE       MAE          MPE     MAPE       ACF1 Theil's U
Test set 0.003694611 0.7526019 0.5976826 -0.07872341 2.399529 -0.5282228       0.5
```

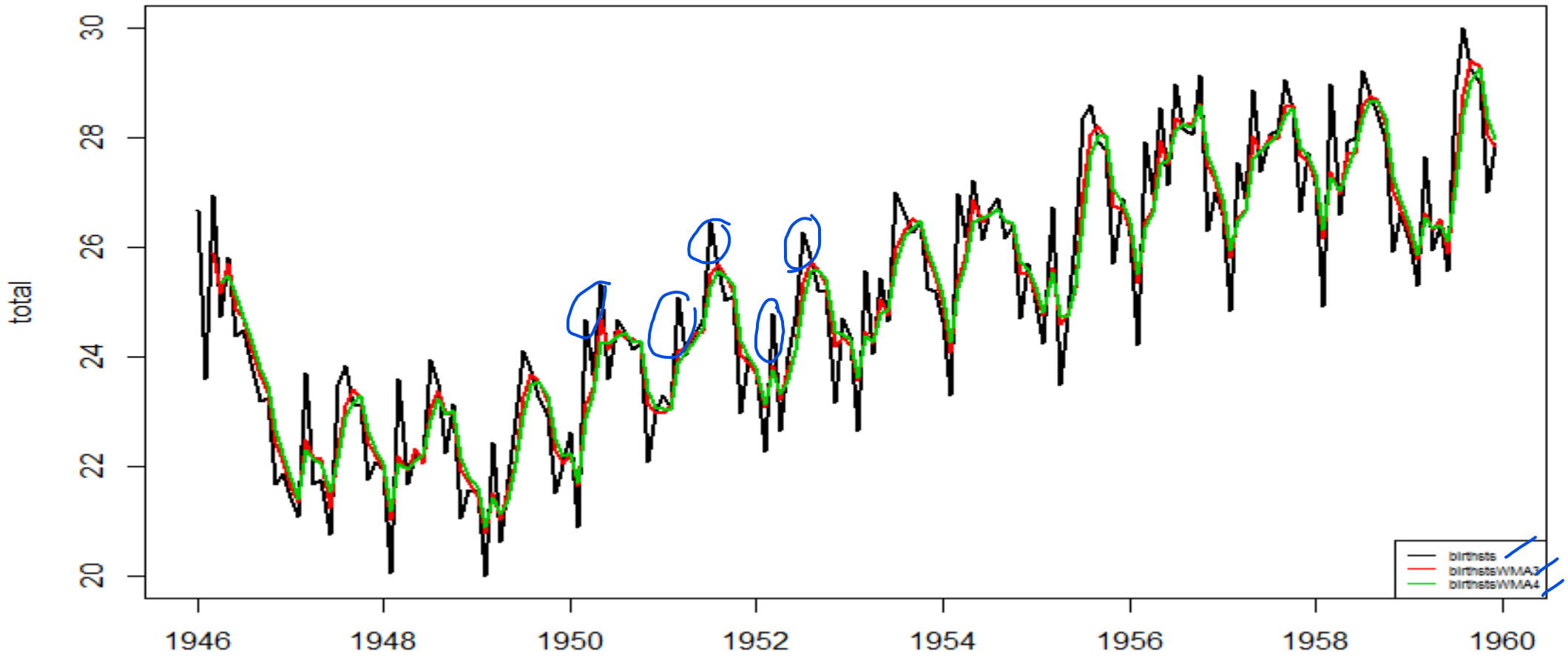
Weighted Moving Average

- Over the years, technicians have found problems with the simple moving average.
- One problem lies in the time frame of the moving average (MA). Analysts believe in assigning more weights to the more recent data.
- A weighted average is any average that has multiplying factors to give different weights to data at different positions in the sample window.

R Code: WMA

- **xx <- c(.2,.3,.5)** → weights
- birthstsWMA3 = WMA(birthsts, n=3, wts=xx)
- birthstsWMA3
- yy <- c(.1,.2,.3,.4) → weight
- birthstsWMA4 = WMA(birthsts, n=4, wts=yy)
- birthstsWMA4
- total = cbind(birthsts,birthstsWMA3,birthstsWMA4)
- plot(total, plot.type="single",col = 1:ncol(total), lwd = c(2, 2, 2))
- legend("bottomright", colnames(total), col=1:ncol(total), lty = c(1, 1, 1), cex=.5, y.intersp = 1)

WMA Plot


total
30
28
26
24
22
20
1946
1948
1950
1952
1954
1956
1958
1960
births
birthsWMA3
birthsWMA4

Extra on Linear Weighting

- WMA is not really using a manually set weighting system
- Weights = 1,2,3,4,5
- 5/15, 4/15, 3/15, 2/15, 1/15
- Therefore WMA is 83(5/15) + 81(4/15) + 79(3/15) + 79(2/15) + 77(1/15) = 80.7

Day	Price
1	77
2	79
3	79
4	81
5 (Current)	83

Notes on Moving Averages

- MA models do not provide information about forecast confidence.
- We can not calculate standard errors.
- We can not explain the stochastic component of the time series. This stochastic component creates the error in our forecast.
- Lag factor – the longer moving average, the more the lag
  - Tells us behavioral change of the time series data (e.g., delay)

Exponential Smoothing Methods

- Single Exponential Smoothing (Averaging)
  - Used for a series without a trend and a seasonal component.
- Double Exponential Smoothing
  - Double Exponential Smoothing is for a series with a trend but without a seasonal component.
- Winter's Model
  - Winter's model is for a series with a trend and seasonal component.

Single Exponential Smoothing

- Averaging (smoothing) past values of a series in a decreasing (exponential) manner.
- The observations are weighted with more weight being given to the more recent observations

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_t$$

- New forecast = $\alpha \times$(old observation) + (1- α) $\times$ old forecast.
- The equation can be rewritten as:

$$\hat{y}_{t+1} = \hat{y}_t + \alpha(y_t - \hat{y}_t)$$

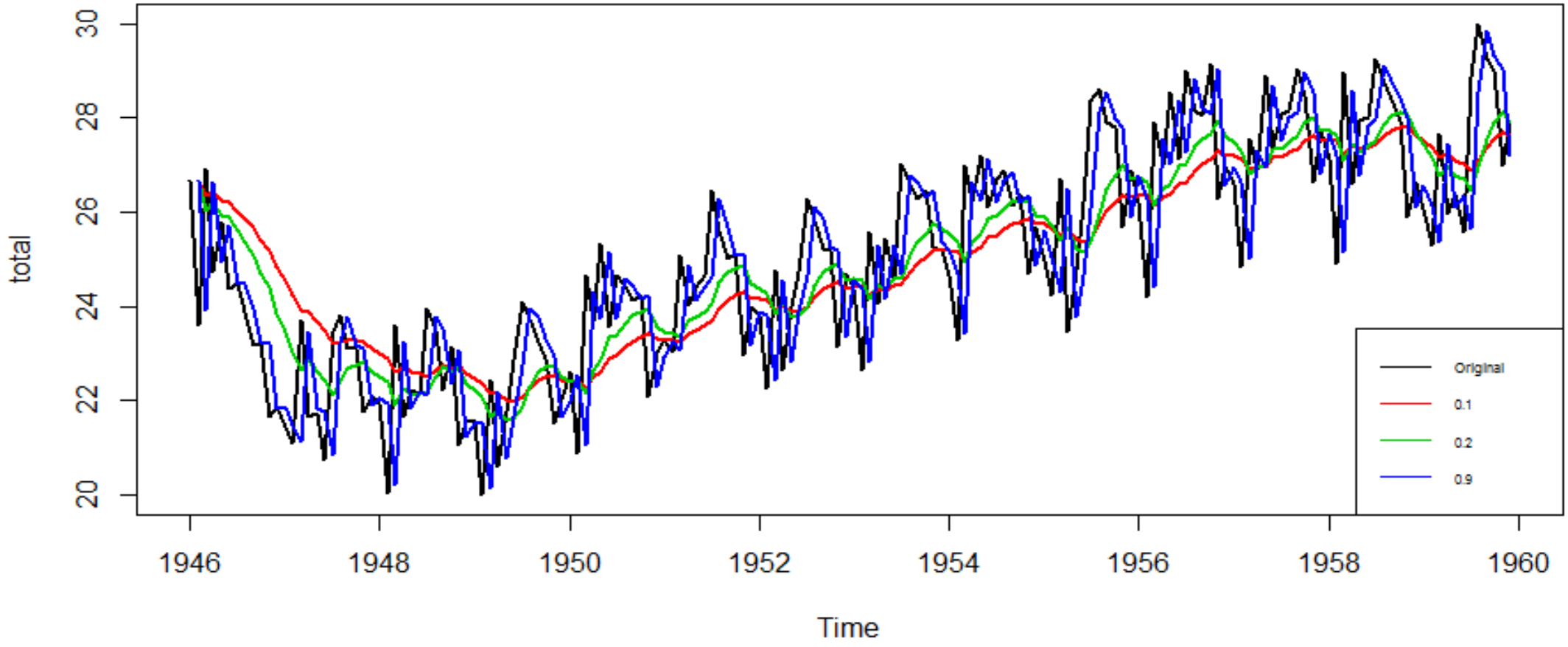
Single Exponential Smoothing

- We need a smoothing constant α, an initial forecast, and an actual value.
- The smoothing constant serves as the weighting factor.
  - When α is close to 1, the new forecast will include a substantial adjustment for any error that occurred in the preceding forecast.
  - When α is close to 0, the new forecast is very similar to the old forecast.
- The smoothing constant α is not an arbitrary choice but generally falls between 0.1 and 0.4.

Single Exponential Smoothing: R Code

- `birthstssesa01 = HoltWinters(birthsts,alpha=0.1, beta=FALSE, gamma=FALSE)`
- `birthstssesa02 = HoltWinters(birthsts,alpha=0.2, beta=FALSE, gamma=FALSE)`
- `birthstssesa09 = HoltWinters(birthsts,alpha=0.9, beta=FALSE, gamma=FALSE)`
- `total = cbind(birthsts,birthstssesa01$fitted[,1],birthstssesa02$fitted[,1],birthstssesa09$fitted[,1])`
- `plot(total, plot.type="single", col = 1:ncol(total), lwd = c(2, 2, 2,2))`
- `legend("bottomright", c("Original","ses0.1","ses0.2","ses0.9"), col=1:ncol(total), lty = c(1, 1), cex=.5, y.intersp = 1)`

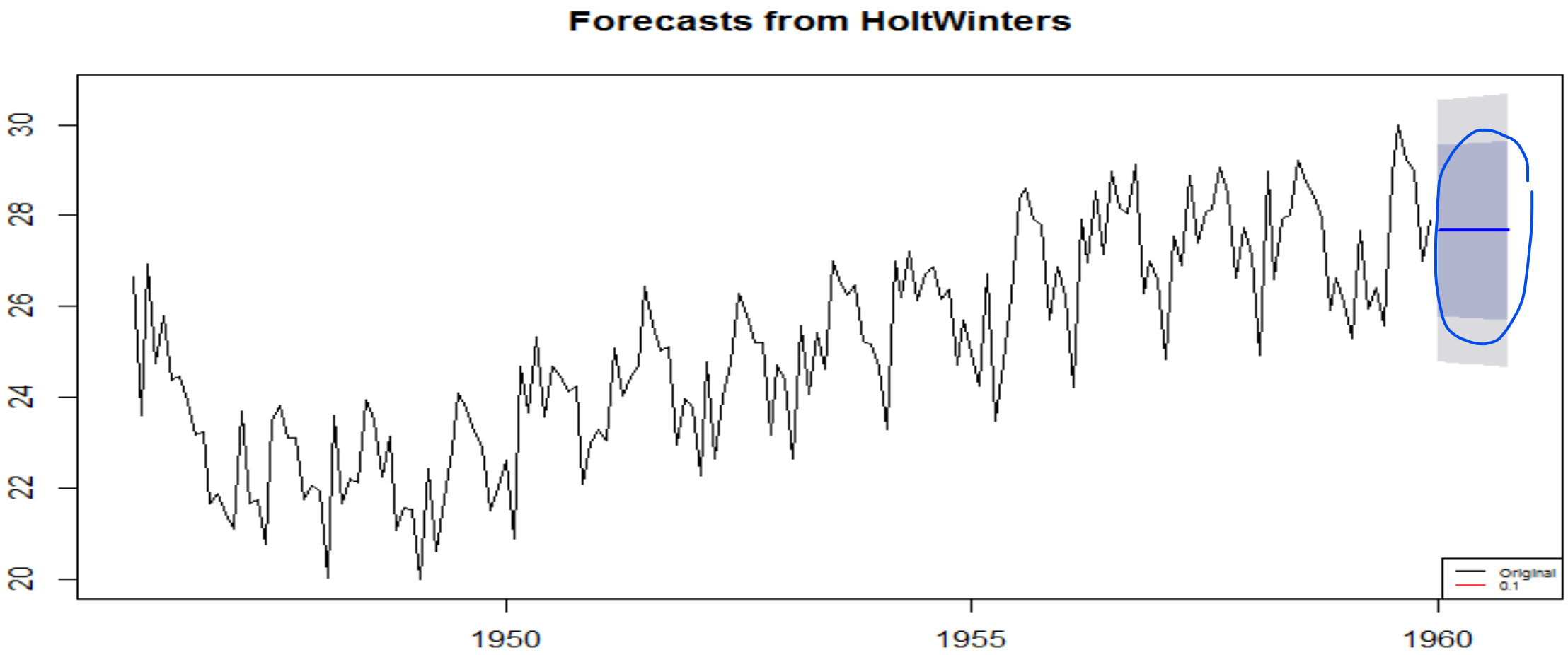
Choice of Alpha, α



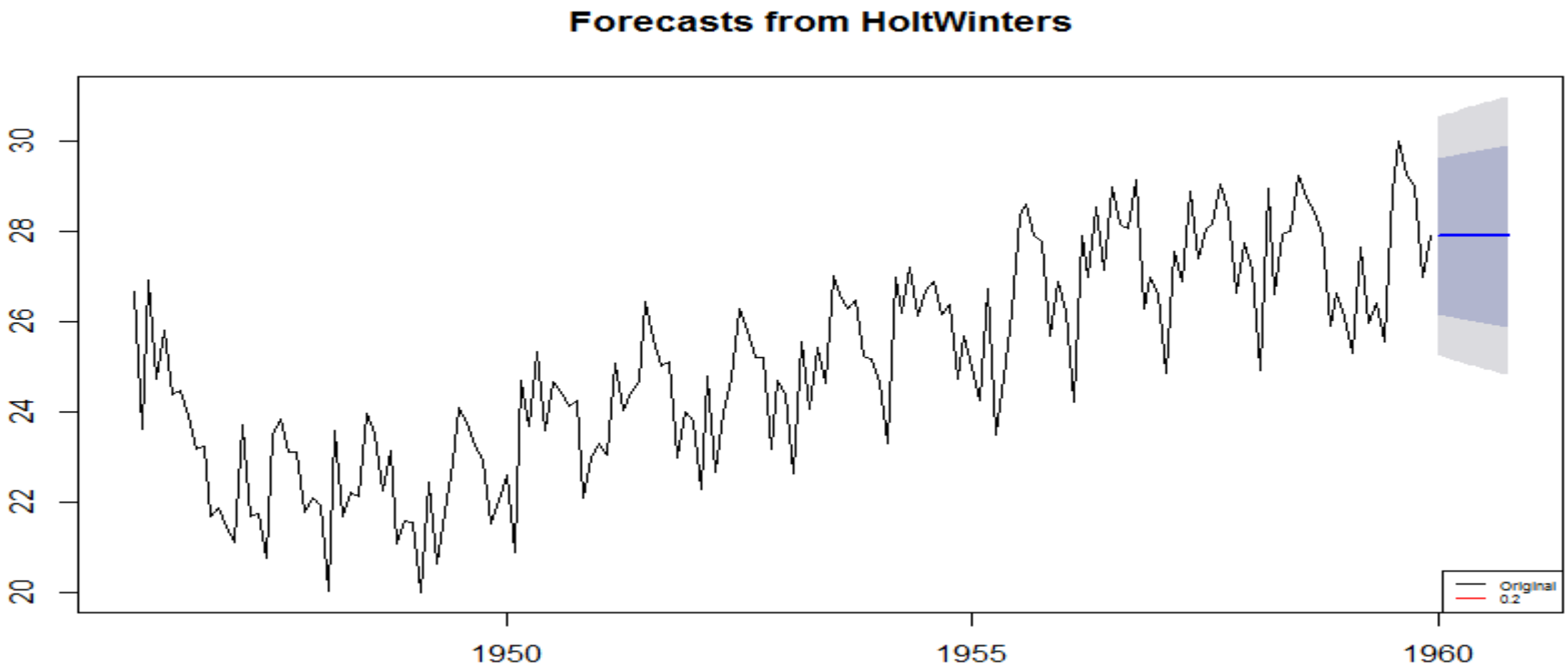
Single Exponential Smoothing Forecast: R Code

- `birthstssesforecast01 = forecast(birthstssesa01, h=10)`
- `birthstssesforecast02 = forecast(birthstssesa02, h=10)`
- `birthstssesforecast09 = forecast(birthstssesa09, h=10)`
- `plot(birthstssesforecast01)`
- `plot(birthstssesforecast02)`
- `plot(birthstssesforecast09)`

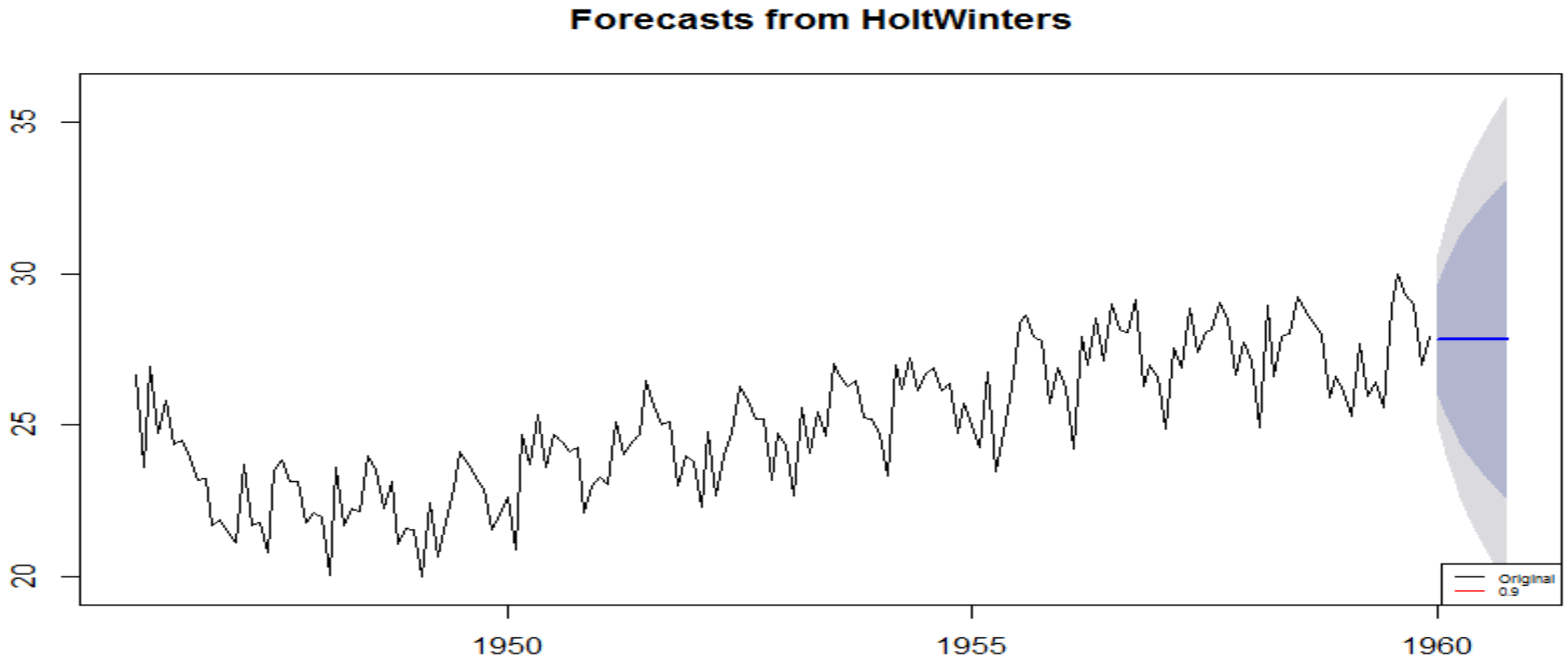
Forecast


Forecasts from HoltWinters
30
28
26
24
22
20
1950
1955
1960
Original
0.1

Forecast


Forecasts from HoltWinters
30
28
26
24
22
20
1950
1955
1960
Original
0.2

Forecast


Forecasts from HoltWinters
35
30
25
20
1950
1955
1960
Original
0.9

Single Exponential Smoothing Forecast: R Code

- `accuracy(birthstssesforecast01)`
- `accuracy(birthstssesforecast02)`
- `accuracy(birthstssesforecast09)`

Single Exponential Smoothing Forecast: R Code

```
> accuracy(birthstssesforecast01)
                      ME     RMSE      MAE        MPE     MAPE     MASE       ACF1
Training set 0.05971923 1.46493 1.219158 0.1068986 4.907687 1.291966 0.4175016
> accuracy(birthstssesforecast02)
                      ME     RMSE      MAE        MPE     MAPE     MASE      ACF1
Training set 0.03688771 1.346141 1.118519 0.1116814 4.506428 1.185318 0.242625
> accuracy(birthstssesforecast09)
                       ME     RMSE      MAE        MPE     MAPE     MASE        ACF1
Training set 0.007744566 1.434287 1.124833 0.1983202 4.568417 1.192009 -0.4568299
```

Holt's Method: Double Exponential Smoothing

- In some situations, the observed data are trending and contain information that allows the anticipation of future upward (or downward) movement.
- In that case, a linear trend forecast function is needed.
- Holt's smoothing method allows for evolving local linear trend in a time series and can be used to forecast.
- When there is a trend, an estimate of the current slope and the current level is required.

Holt's Method: Double Exponential Smoothing

- Holt's method uses two coefficients.
  - α is the smoothing constant for the *level*
  - β is the *trend* smoothing constant - used to remove random error
- Advantage of Holt's method: it provides flexibility in selecting the rates at which the level and trend are tracked.

Equations in Holt's Method

- The exponentially smoothed series, or the current level estimate

$$\hat{y}_t = \alpha y_t + (1 - \alpha)(\hat{y}_{t-1} + T_{t-1})$$

- The trend estimate:

$$T_t = \beta(\hat{y}_t - \hat{y}_{t-1}) + (1 - \beta)T_{t-1}$$

- Forecast *m* periods into the future:

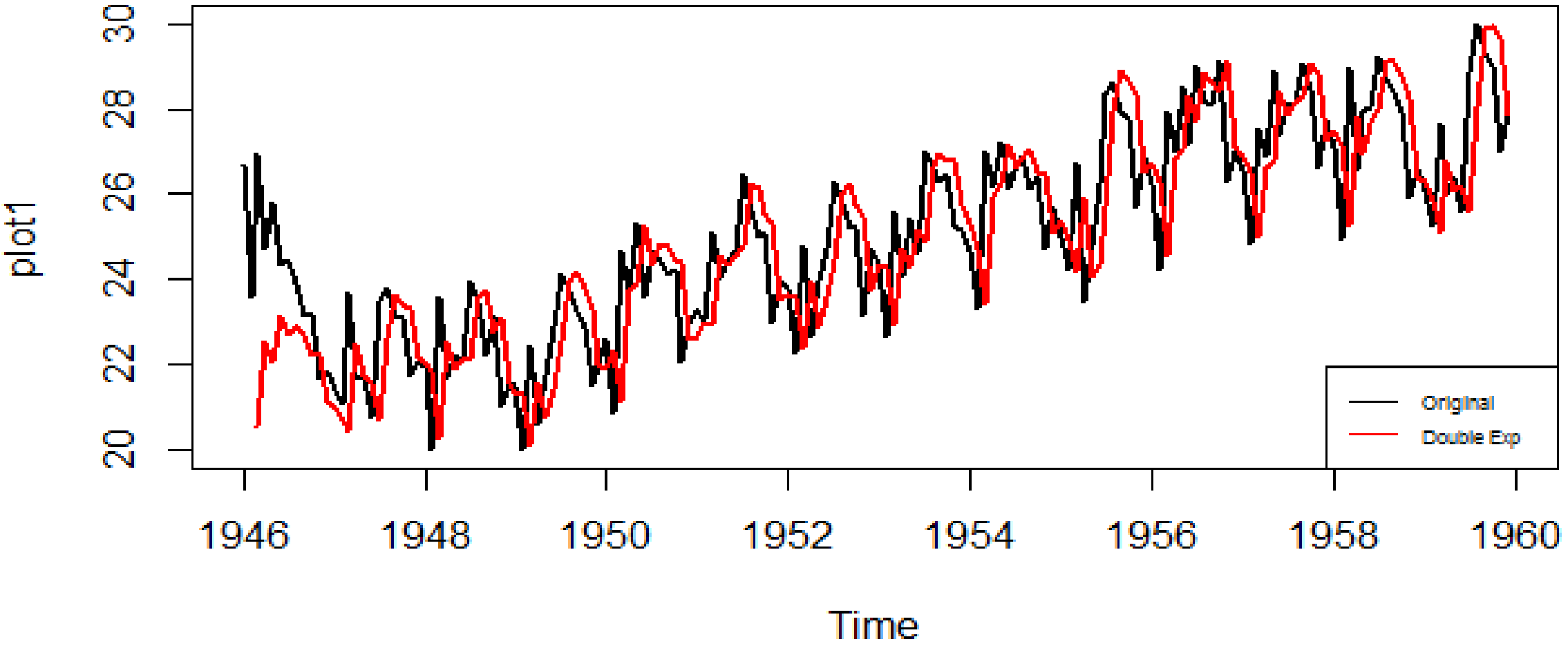
$$\hat{y}_{t+m} = \hat{y}_t + mT_t$$

  - Where:
    - T_t = trend estimate
    - y_{t+m} = forecast for *p* periods into the future.
    - α = smoothing constant for the level
    - β = smoothing constant for trend estimate

R Code: Holt's Method - Double Exponential Smoothing

- `birthstsdep = HoltWinters(birthsts, gamma=FALSE)`
- `total = cbind(birthsts,birthstsdep$fitted[,1])`
- `plot(total,plot.type="single", col = 1:ncol(total), lwd = c(2, 2))`
- `legend("bottomright", c("Original","Double Exp"), col=1:ncol(total), lty = c(1, 1), cex=.5,y.intersp = 1)`

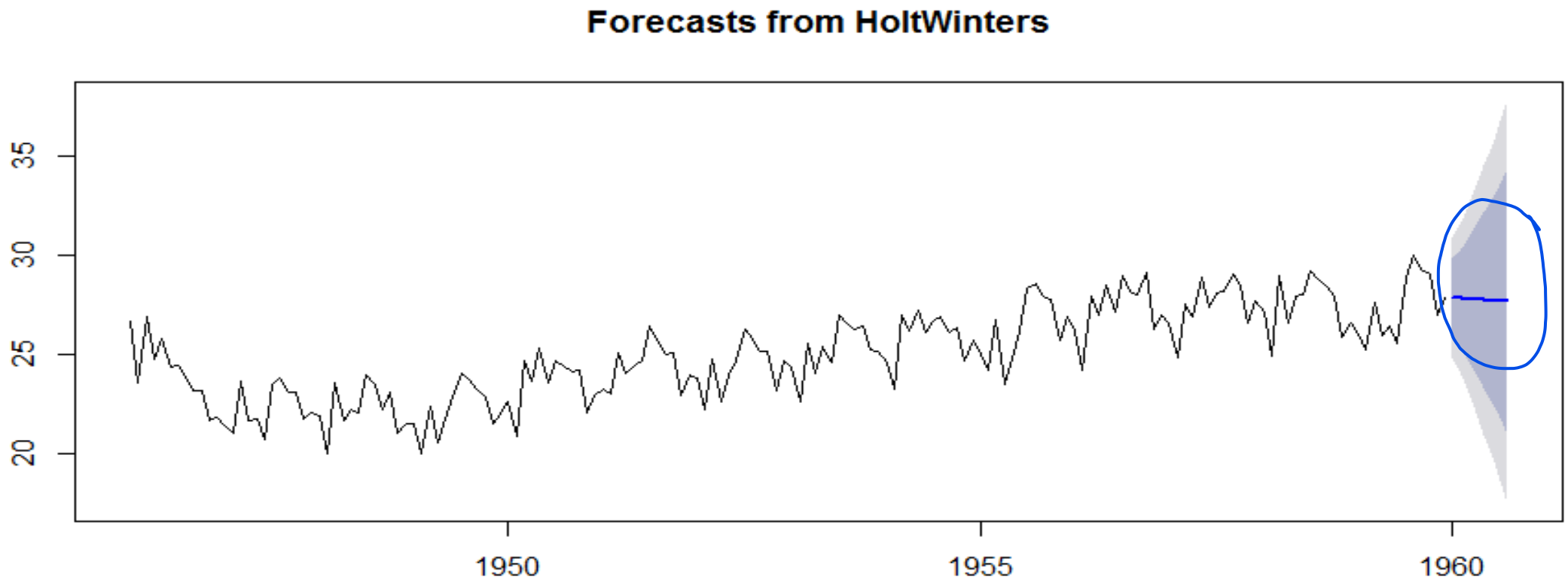
Holt's Method



R Code: Forecast - Holt's Method

- `birthstsdepforecast = forecast(birthstsdep , h=8)`
- `plot(birthstsdepforecast)`

Forecast: Holt's Method



Winter's Method

- Winters' method is an easy way to account for seasonality when data have a seasonal pattern.
- It extends Holt's Method to include an estimate for seasonality.
  - α is the smoothing constant for the level
  - β is the trend smoothing constant - used to remove random error
  - γ smoothing constant for seasonality
- This formula removes seasonal effects. The forecast is modified by multiplying by a seasonal index.

Equations in Winter's Method

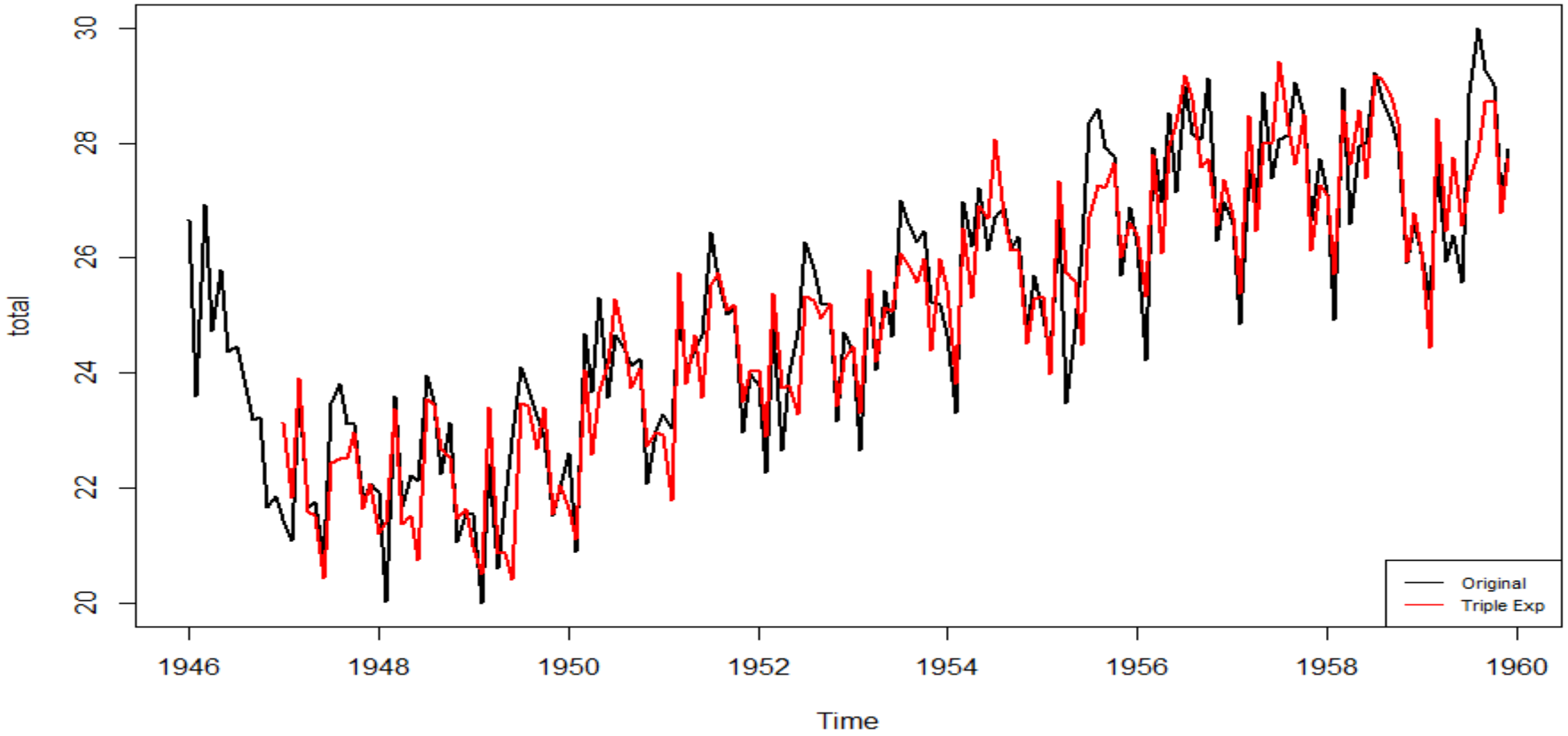
- The exponentially smoothed series, or the current level estimate:
  - $\hat{y}_t = \frac{\alpha y_t}{S_{t-s}} + (1-\alpha)(\hat{y}_{t-1} + T_{t-1})$
- The trend estimate:
  - $T_t = \beta(\hat{y}_t - \hat{y}_{t-1}) + (1-\beta)T_{t-1}$
- The seasonality estimate:
  - $St = \frac{\gamma y_t}{\hat{y}_t} + (1-\gamma)S_{t-s}$
- Forecast m periods into the future:
  - $\hat{y}_{t+m} = (\hat{y}_t + mTt)\, S_{t-s+m}$

γ = smoothing constant for seasonality estimate
m = periods to be forecast into the future
s = length of season

R Code: Winter's Method

- birthststes= HoltWinters(birthsts)
- total = cbind(birthsts,birthststes$fitted[,1])
- plot(total, plot.type="single", col = 1:ncol(total), lwd= c(2, 2))
- legend("bottomright", c("Original","TripleExp"), col=1:ncol(total), lty= c(1, 1), cex=.7, y.intersp= 1)

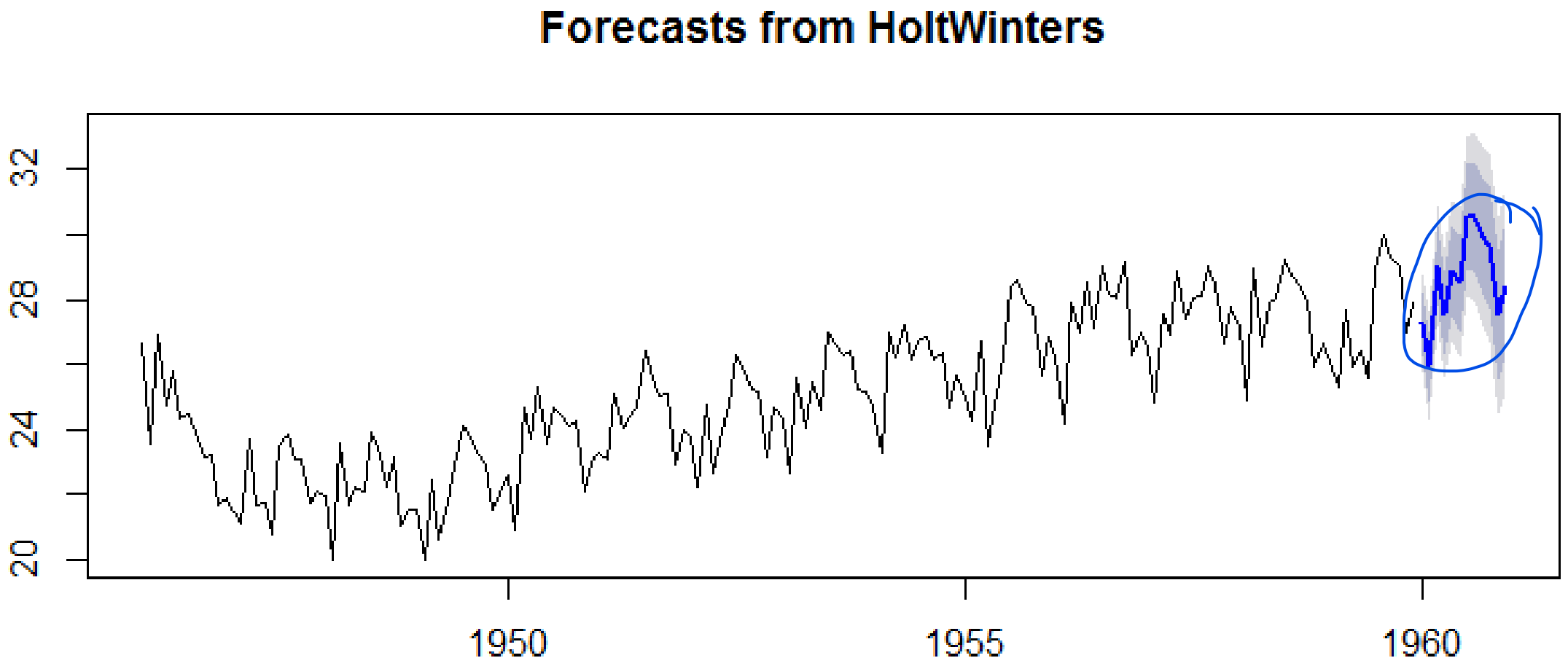
Forecast: Winter's Method



Forecast: Winter's Method

- `library("forecast")`
- `birthststes= HoltWinters(birthsts)`
- `birthstesforecast = forecast(birthststes , h=12)`
- `plot(birthstesforecast)`

Forecast: Winter's Method



Comparing DEP and TEP for births

- `accuracy(birthstsdep$fitted, birthsts)`
- `accuracy(birthststes$fitted, birthsts)`

```
> accuracy(birthstsdep$fitted, birthsts)
                    ME     RMSE      MAE       MPE     MAPE      ACF1
Test set 0.1571495 1.640062 1.249048 0.4435553 4.982137 0.2791452
          Theil's U
Test set   1.044106
> accuracy(birthststes$fitted, birthsts)
                     ME      RMSE       MAE       MPE     MAPE      ACF1
Test set 0.08838484 0.7635132 0.5910602 0.307559 2.361872 0.3134715
          Theil's U
Test set 0.4901279
```

Decomposition

- Decomposition is a procedure to identify the component factors of a time series.
- How the components relate to the original series: a model that expresses the time series variable Y in terms of the components T (trend), C (cycle), S (seasonal) and I (irregular).
- It is difficult to deal with cyclical component of a time series. To keep things simple we assume that any cycle in the data is part of the trend.

Decomposition

- Two models: Additive components model and multiplicative components model
- Additive model: $y_t = T_t + S_t + I_t + Ct$
- Multiplicative model: $y_t = T_t \times S_t \times I_t \times C_t$
- Decomposition finds the estimates of these four components

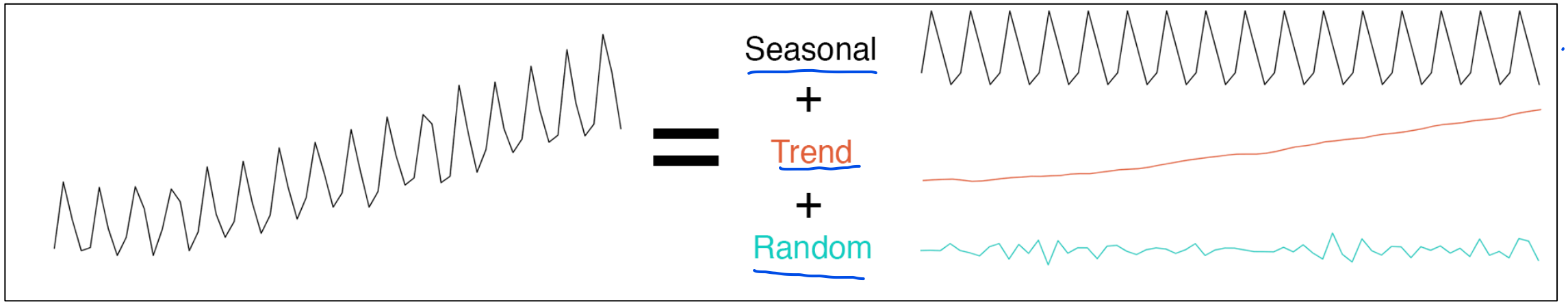
Decomposition

- There are a lot of different ways to decompose your data
  - A lot of different functions in R
  - Several manual procedures
  - Use of moving averages (e.g., centered MA)
  - De-trend or de-seasonalizing comes first
  - Use of seasonal relatives/indices

Steps in Decomposition Method

1. Plot the time series data
2. Identify time series model – Additive or Multiplicative
3. Detect the trend
4. De-trend the time series
5. Estimate the seasonality
6. Examine Irregular and Random variations
7. Reconstruct original time series

Decomposition



Steps in Decomposition Method

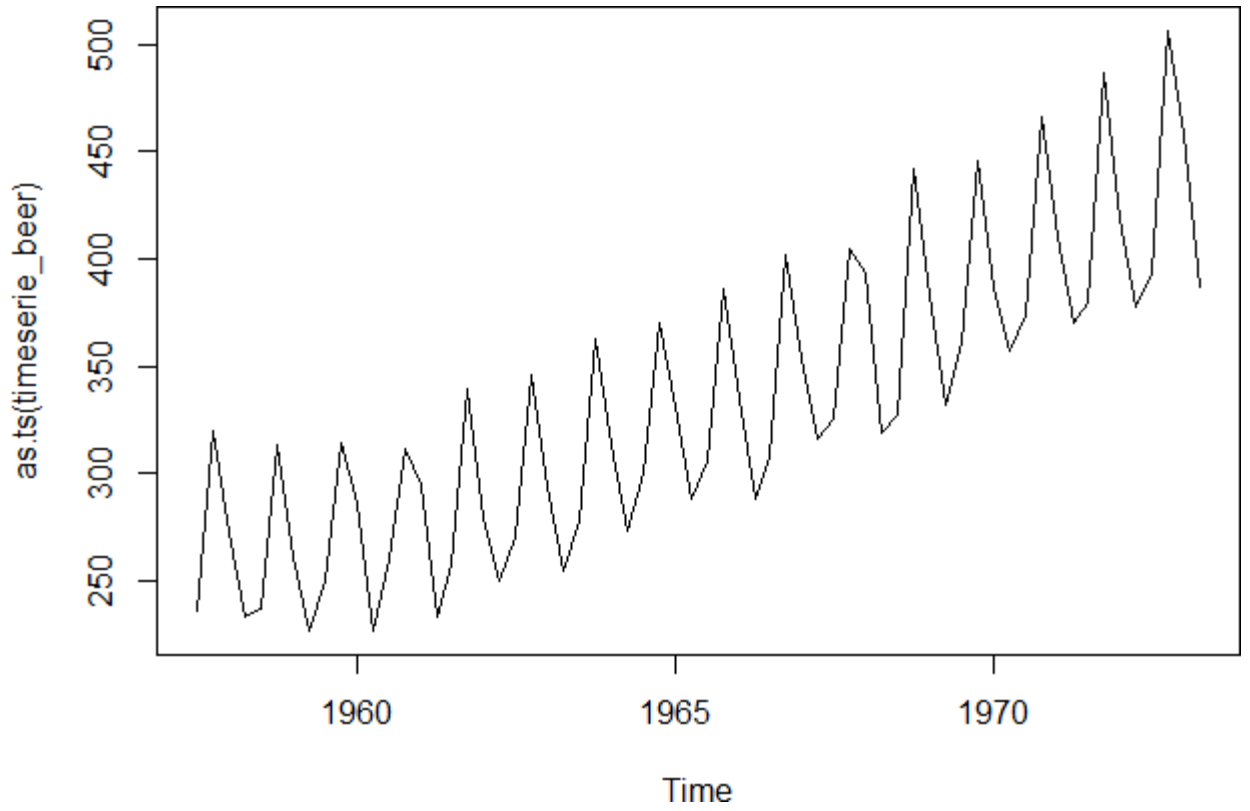
1. Plot the time series data
2. Identify time series model – Additive or Multiplicative
3. Detect the trend
4. De-trend the time series
5. Estimate the seasonality
6. Examine Irregular and Random variations
7. Reconstruct original time series

Steps in Decomposition Method

1. Plot the time series data
2. Identify time series model – Additive or Multiplicative

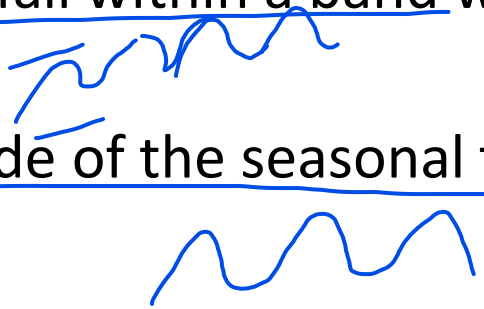
- `install.packages("fpp")`
- `library(fpp)`
- `data(ausbeer)`
- `class(ausbeer)`
- `ausbeer`
- `timeserie_beer = tail(head(ausbeer, 70),64)`
- `plot(as.ts(timeserie_beer))`

Steps in Decomposition Method



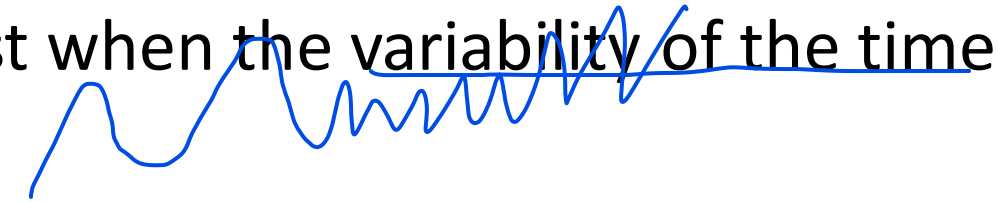
Additive and Multiplicative Models

- The additive model works best when the time series has roughly the same variability through the length of the series.
  - All the values of the series fall within a band with constant width centered on the trend.
  - Appropriate if the magnitude of the seasonal fluctuation does not vary with the level of the series



Additive and Multiplicative Models

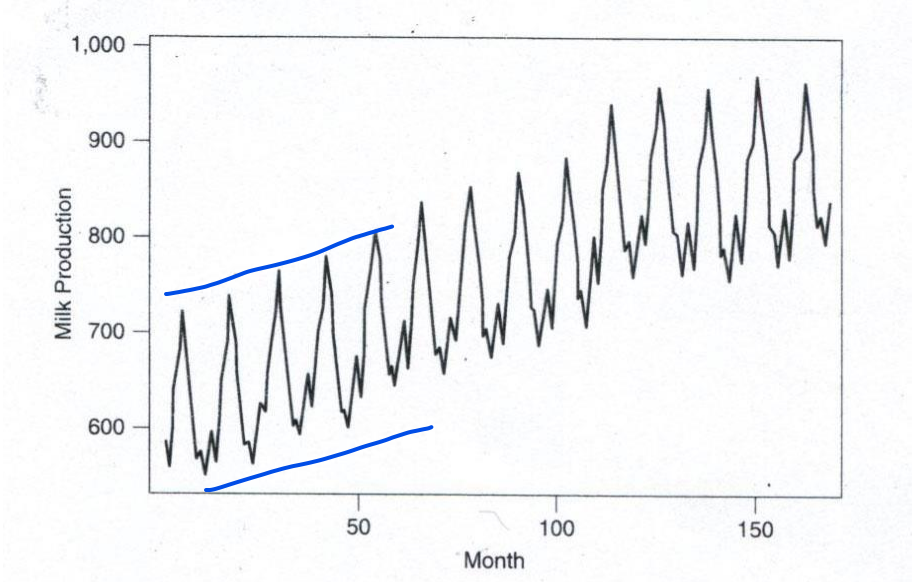
- The multiplicative model works best when the variability of the time series increased with the level.
  - Variance of the series varies as the trend progresses
  - More prevalent with economic series since most seasonal variation increases with the level of the series



Note on Cycles

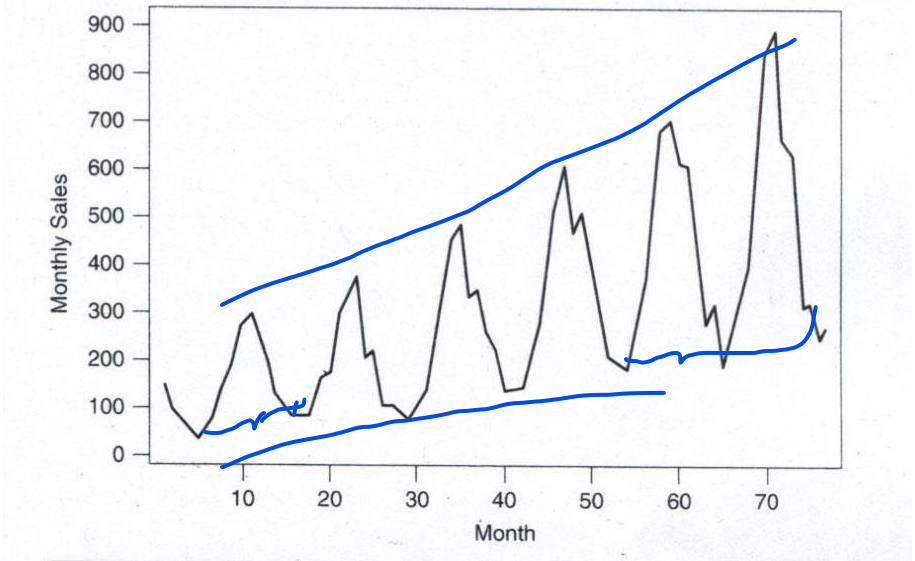
- Cycles are often difficult to identify with a short time series
- Thus, in this module, cycle components shall be assumed to be combined with trend components
- Models will be reduced to the following:
  - Additive model: $y_t = TC_t + S_t + I_t$
  - Multiplicative model: $y_t = TC_t \times S_t \times I_t$

Additive and Multiplicative Models



(a) A time series with constant variability

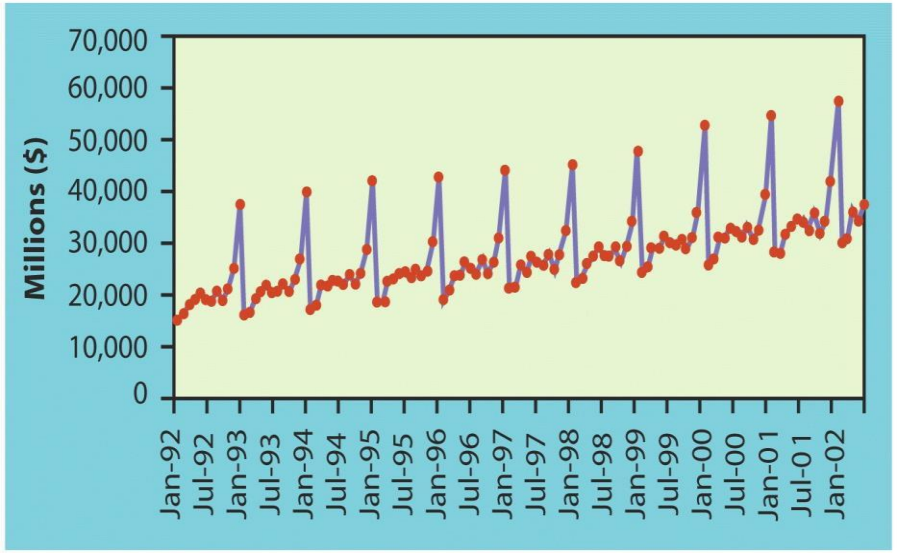
additive



(b) A time series with variability increasing with level

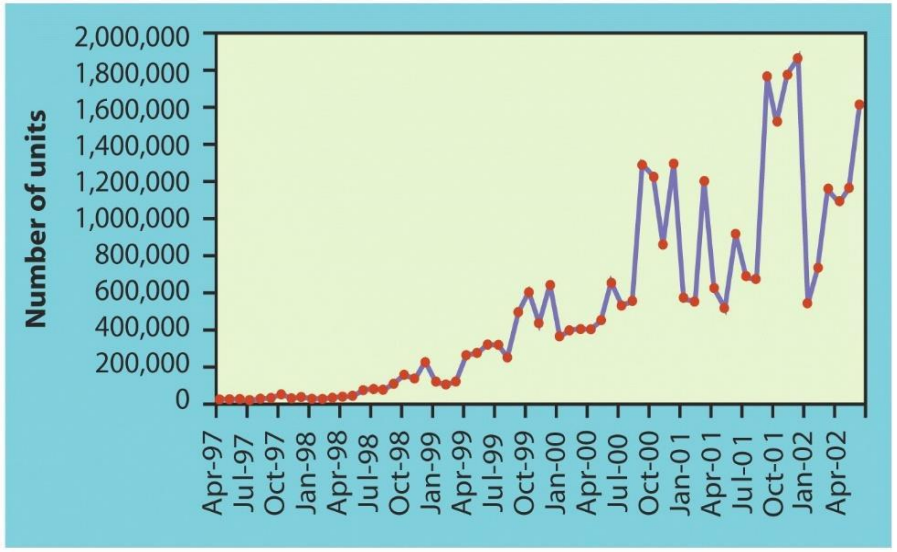
multiplicative

Additive and Multiplicative Models



(a) A time series with constant variability

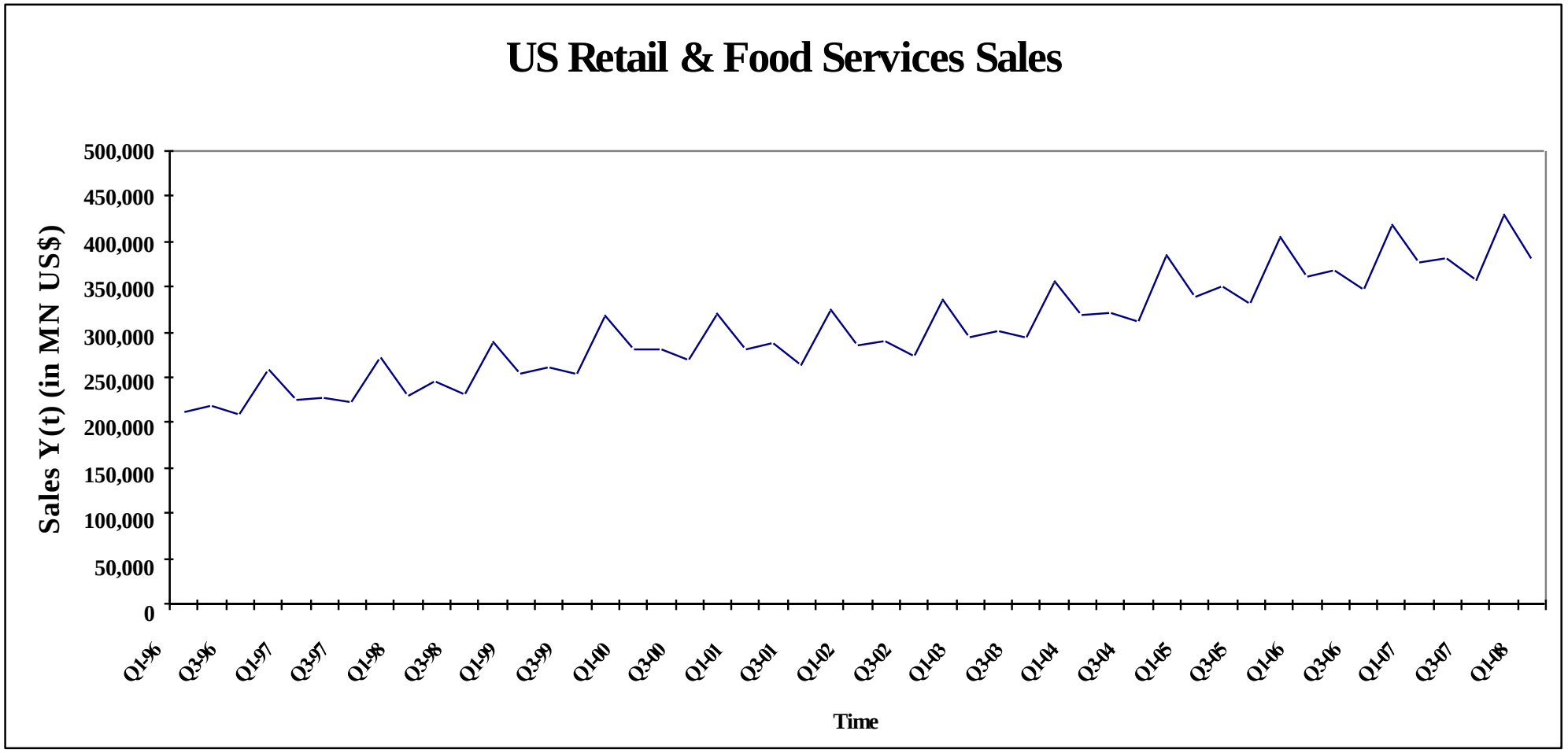
additive



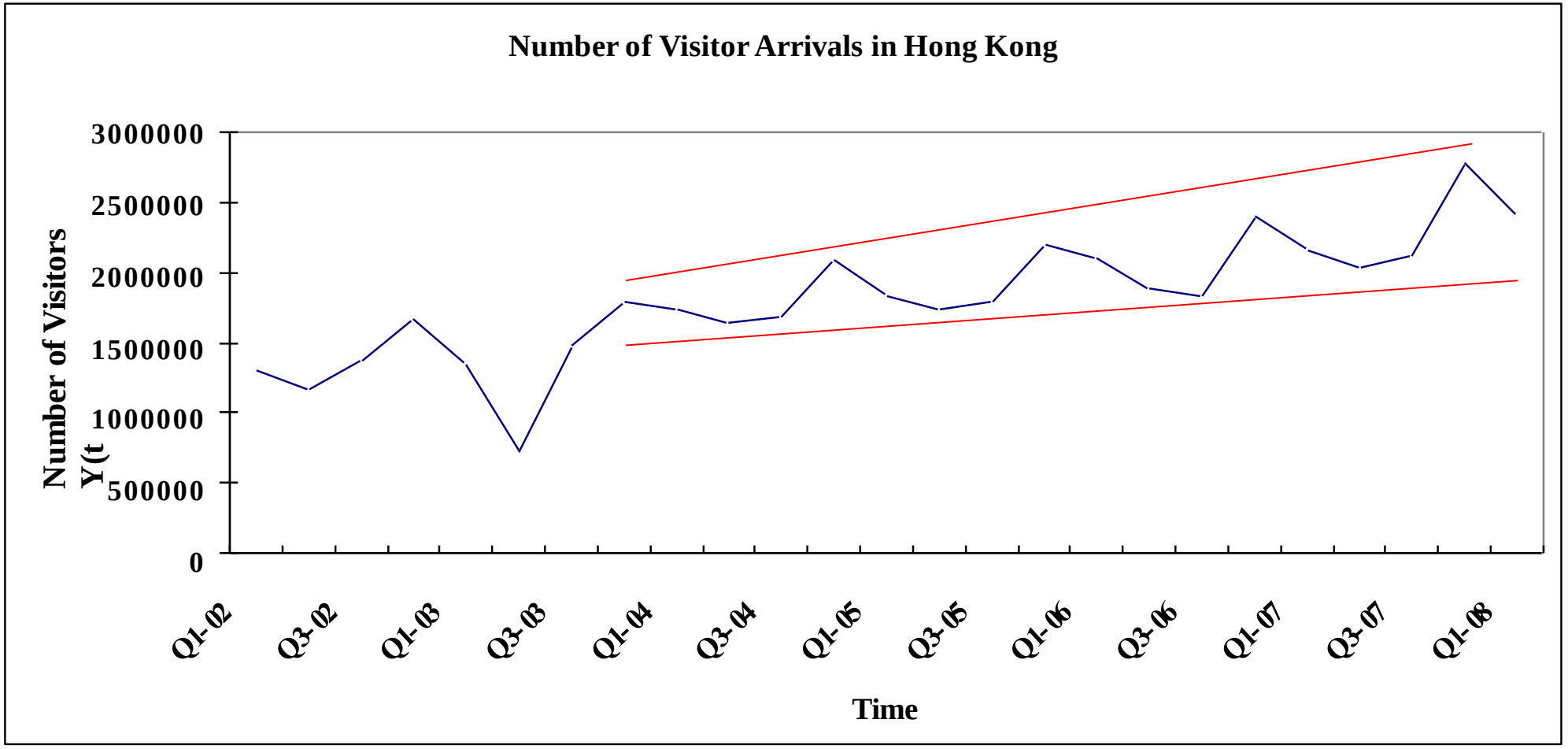
(b) A time series with variability increasing with level

multiplicative

US Retail and Food Services Sales from 1996 Q1 to 2008 Q1



Quarterly Number of Visitor Arrivals in Hong Kong from 2002 Q1 to 2008 Q1



Steps in Decomposition Method

1. Plot the time series data
2. Identify time series model – Additive or Multiplicative
3. Detect the trend
4. De-trend the time series
5. Estimate the seasonality
6. Examine Irregular and Random variations
7. Reconstruct original time series

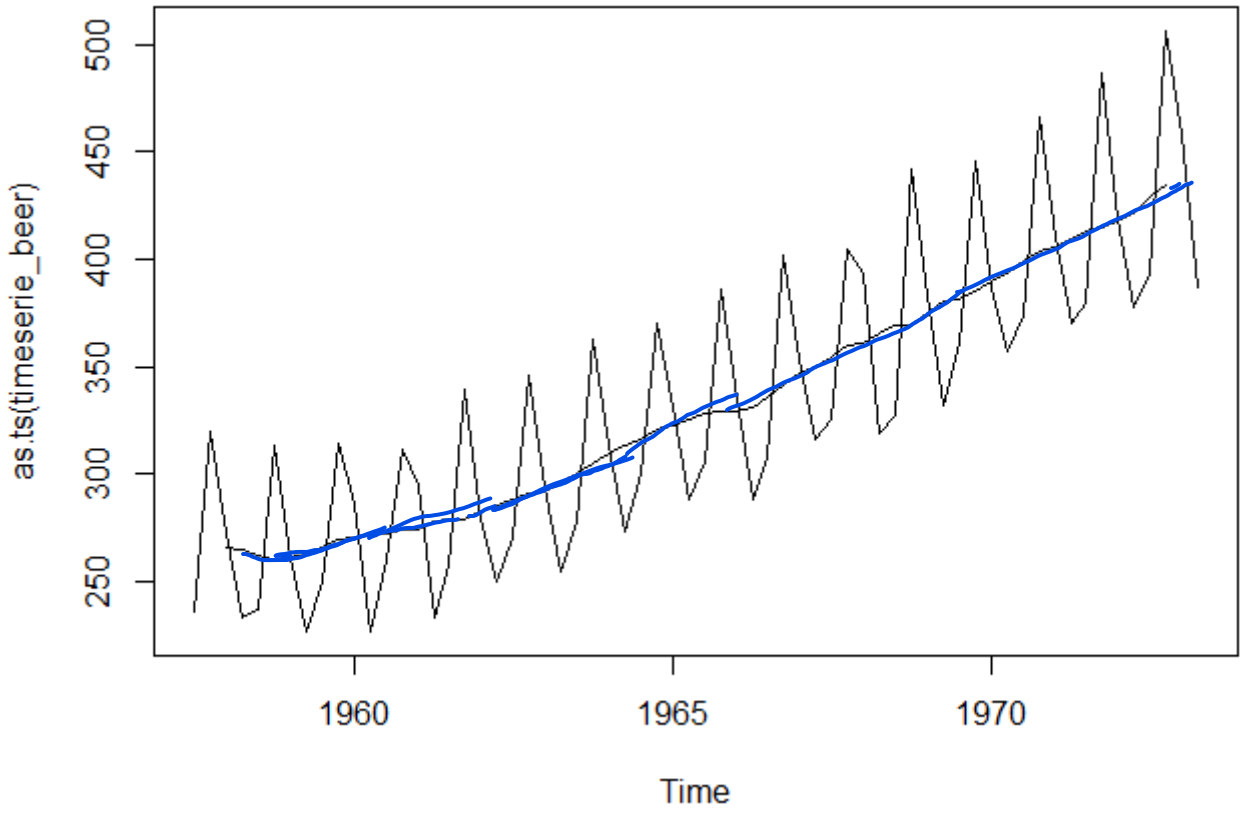
Steps in Decomposition Method

3. Detect the trend
4. De-trend the time series

moving average

```
#Here, MA used as smoother/trend detection
library(forecast)
trend_beer = ma(timeserie_beer, order=4, centre=T)
plot(as.ts(timeserie_beer))
lines(trend_beer)
plot(as.ts(trend_beer))
trend_beer
```

Steps in Decomposition Method




```
> trend_beer
       Qtr1    Qtr2    Qtr3    Qtr4
1957                      NA      NA
1958 265.375 264.625 262.375 260.250
1959 261.125 262.875 266.125 269.250
1960 270.500 271.375 272.125 274.000
1961 274.375 277.500 279.000 279.125
1962 282.875 285.375 288.125 290.625
1963 292.250 295.375 299.875 304.500
1964 309.500 313.125 316.250 320.375
1965 323.000 325.750 328.250 328.750
1966 329.000 331.250 335.500 341.250
1967 346.875 349.375 354.750 360.125
1968 360.750 365.625 369.000 369.375
1969 375.250 380.000 381.000 384.625
1970 389.375 393.500 398.875 403.375
1971 405.625 408.875 412.625 414.750
1972 417.500 421.625 428.875 434.875
1973      NA      NA
```

Trend Component

- Trend-Cycle Estimation
  - Can be done by using smoothing methods or moving averages
  - Idea is that observations which are nearby in time are also likely to be close in value
  - The long-term trend is estimated from the de-seasonalized data for the variable to be forecasted

Trend Component

- Trend-Cycle Estimation
  - The average of the points near an observation will provide a reasonable estimate of the trend-cycle at that observation
  - The average eliminate some of the randomness in the data, and leaves a smooth trend-cycle component
  - Use of method of least squares

Trend Equations Using Simple Linear Regression

- Trend can be described by a moving average trend
- Local regression is a way of fitting a much more flexible trend-cycle curve to the data
- Simple Linear trend:

$$T_t = \beta_0 + \beta_1 t$$

- Can be extended to quadratic models:

$$T_t = \beta_0 + \beta_1 t + \beta_2 t^2$$

- Or Exponential:

$$T_t = \beta_0 \beta_1^{\,t}$$

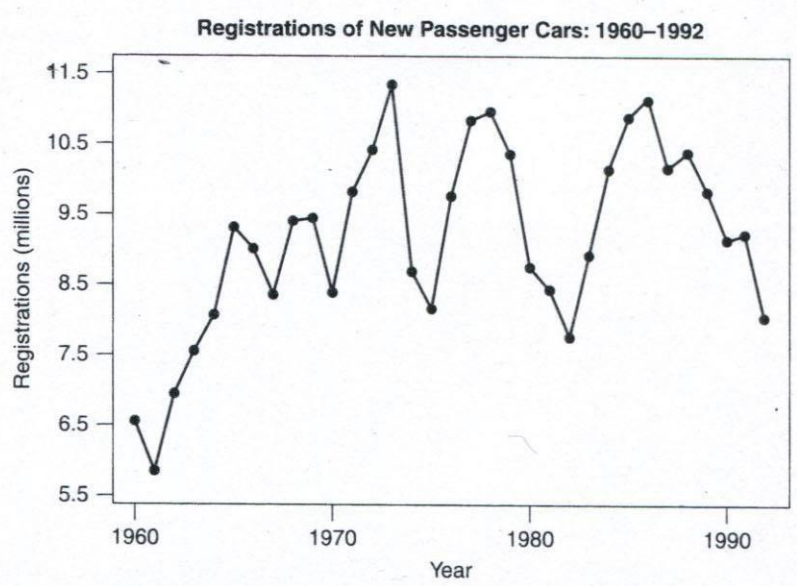
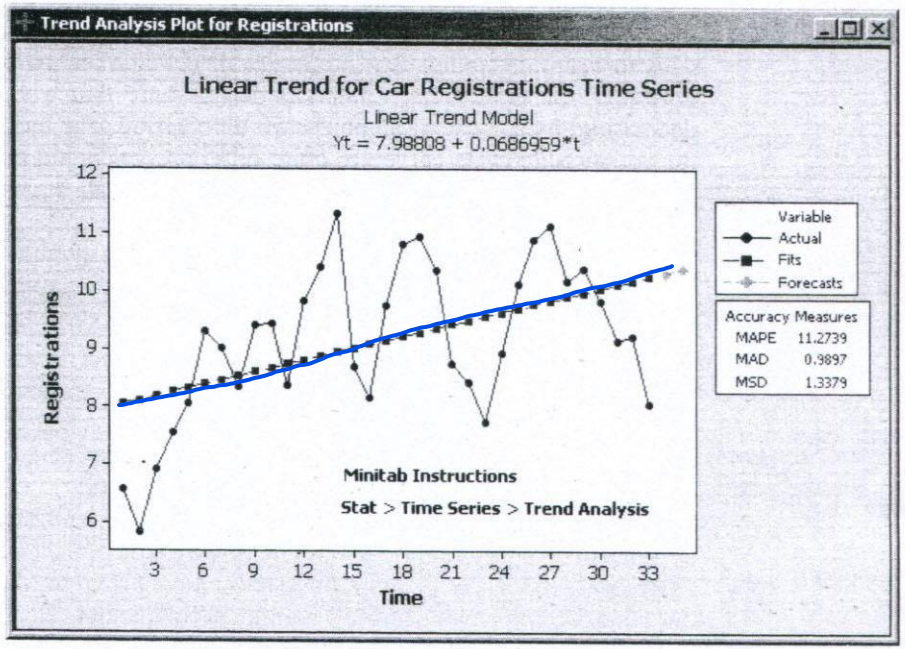
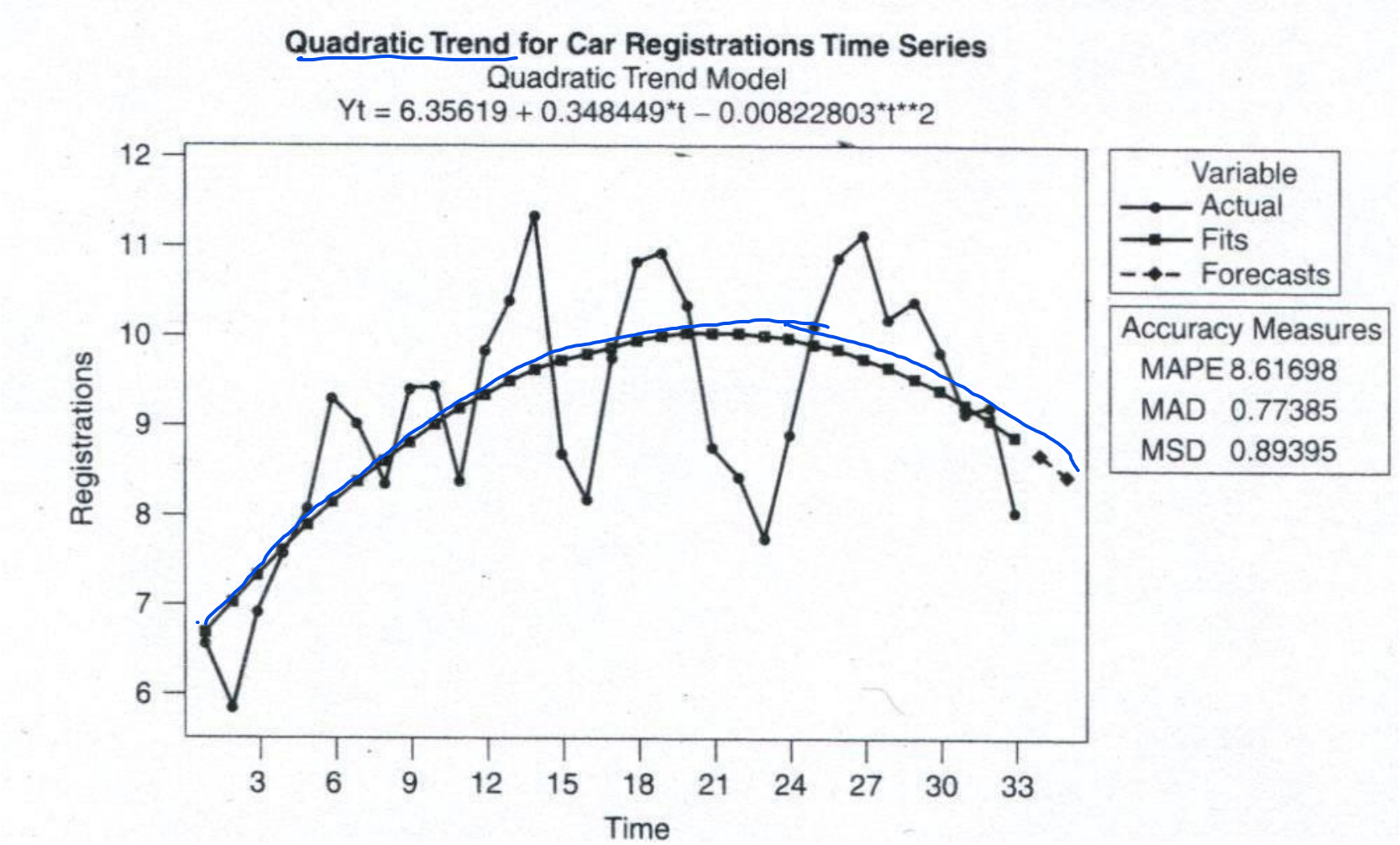


FIGURE 5-2 Car Registrations Time Series for Example 5.1



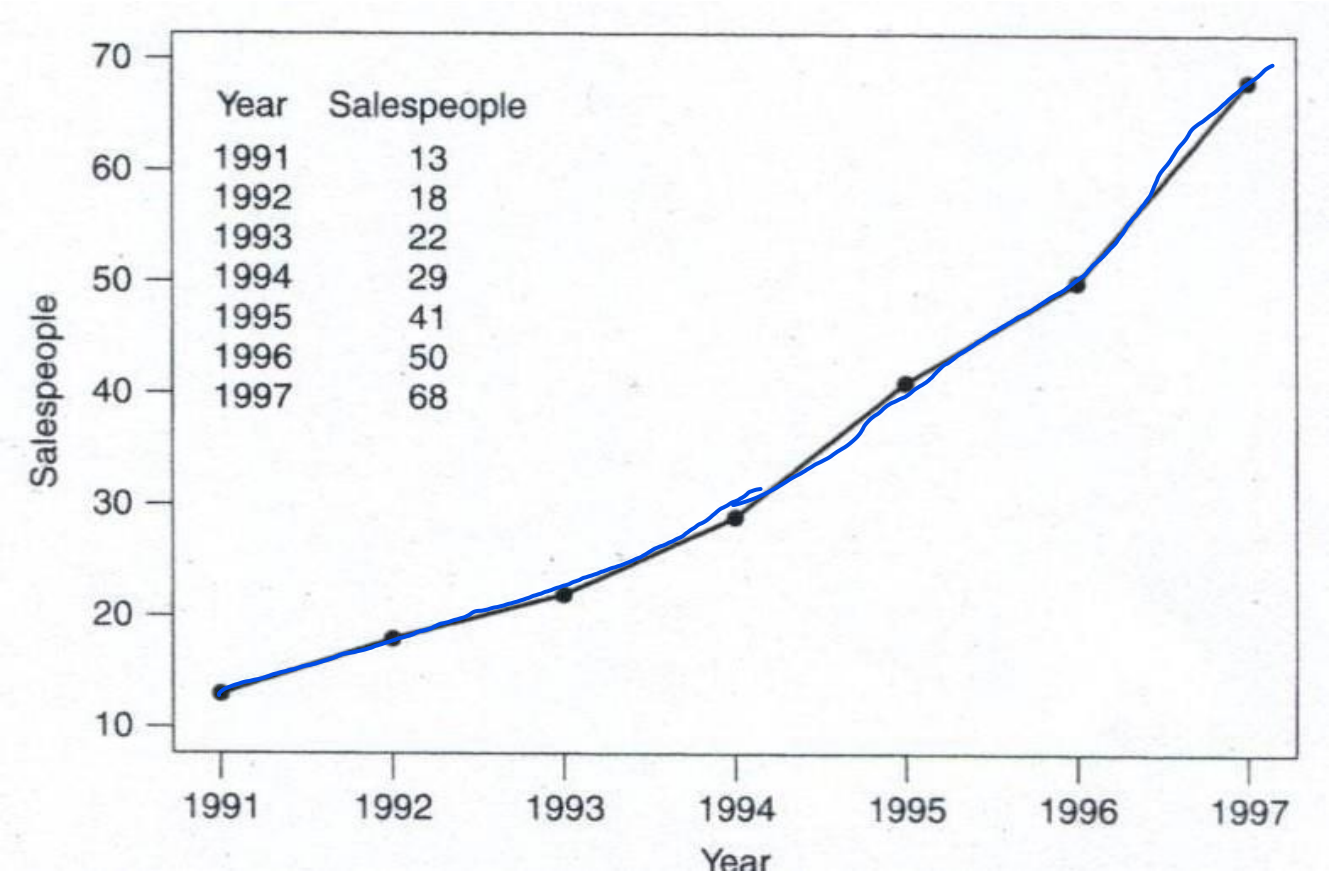
Trend Equations Using Simple Linear Regression: Trend Line for the Car Registrations Time Series

Trend Equations Using Quadratic Regression: Trend Line for the Car Registrations Time Series

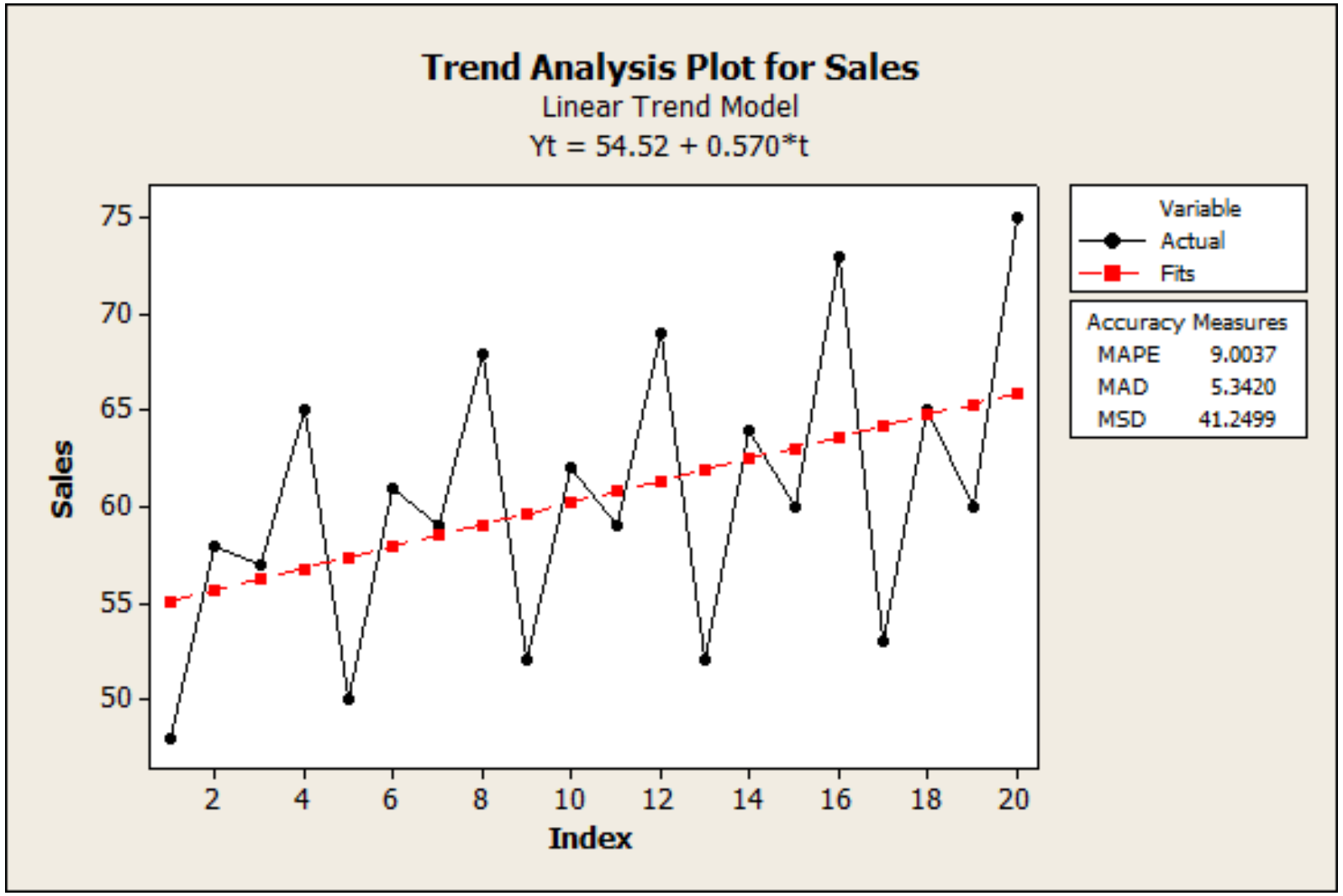


Exponential Trend

- The increase in the number of salespeople is not constant. It appears as if increasingly larger numbers of people are being added in the later years.
- An exponential trend curve fit to the sales people data has the equation:
  - $T'_t = 10.016(1.313)^t$

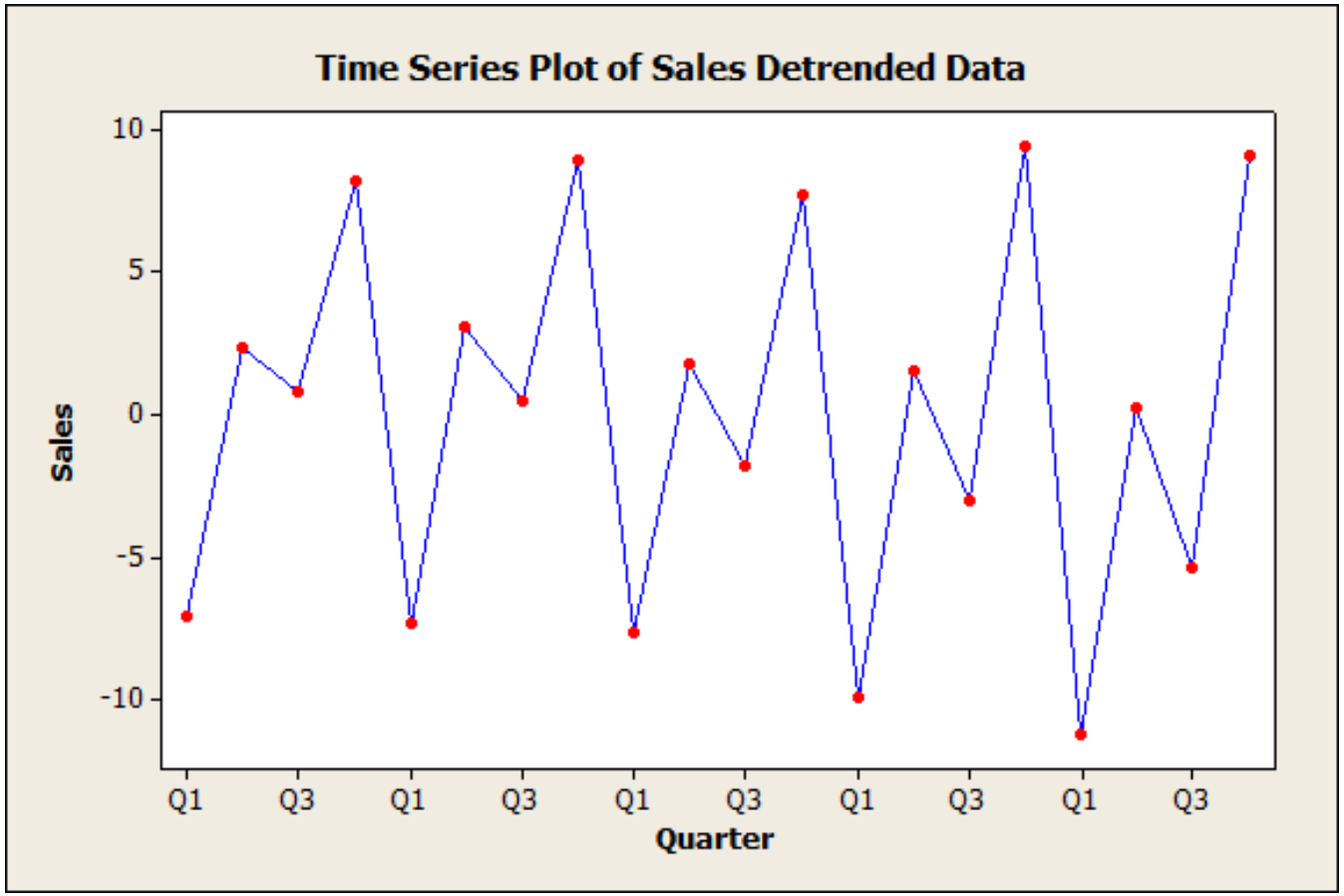


Use Trend Analysis in Minitab®



Sample De-trended Sales Data (Additive Model)

- $y_t - TC_t = S_t + I_t$



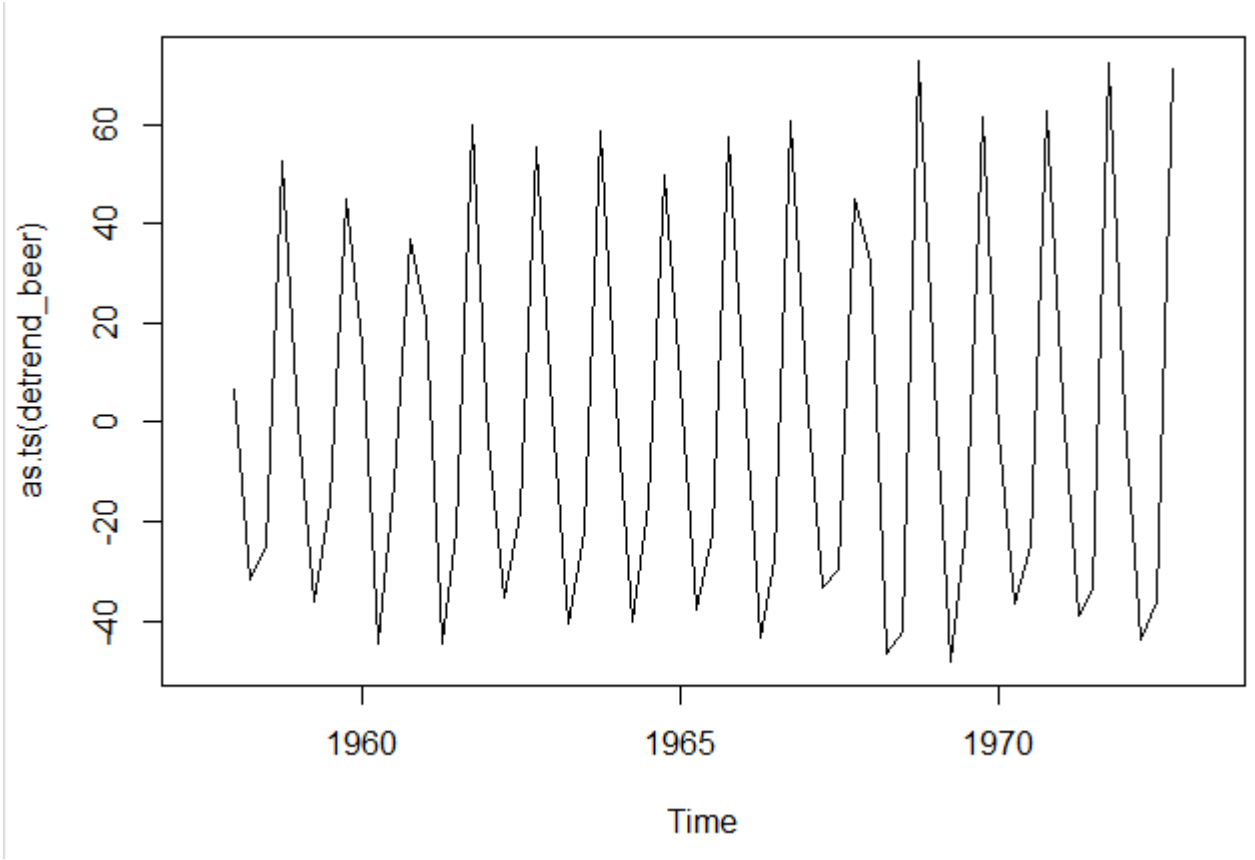
Steps in Decomposition Method

3. Detect the trend
4. De-trend the time series

MA

- `detrend_beer = timeserie_beer - trend_beer`
- `plot(as.ts(detrend_beer))`

Steps in Decomposition Method




```
> detrend_beer
        Qtr1    Qtr2    Qtr3    Qtr4
1957                      NA      NA
1958   6.625 -31.625 -25.375  52.750
1959  -0.125 -35.875 -16.125  44.750
1960  15.500 -44.375 -12.125  37.000
1961  20.625 -44.500 -22.000  59.875
1962  -3.875 -35.375 -18.125  55.375
1963   1.750 -40.375 -21.875  58.500
1964   3.500 -40.125 -16.250  49.625
1965   8.000 -37.750 -22.250  57.250
1966   6.000 -43.250 -27.500  60.750
1967   6.125 -33.375 -29.750  44.875
1968  32.250 -46.625 -42.000  72.625
1969   7.750 -48.000 -20.000  61.375
1970  -2.375 -36.500 -24.875  62.625
1971   4.375 -38.875 -33.625  72.250
1972   1.500 -43.625 -35.875  71.125
1973      NA      NA
```

Seasonality

- Several methods for measuring seasonal variation
- The basic idea:
  - First, estimate and remove the trend from the original series and then smooth out the irregular component. This leaves data containing only seasonal variation.
  - The seasonal values are collected and summarized to produce a number for each observed interval of the year (week, month, quarter, and so on)

Identification of Seasonal Components

- The identification of seasonal component in a time series differs from trend analysis in two ways:
  - The trend is determined directly from the original data, but the seasonal component is determined indirectly after eliminating the other components from the data.
  - The trend is represented by one best-fitting curve, but a separate seasonal value has to be computed for each observed interval.

Seasonal Adjustment

- A useful by-product of decomposition is that it provides an easy way to calculate seasonally adjusted data.
- For additive decomposition, the seasonally adjusted data are computed by subtracting the seasonal component.

$$y_t - S_t = TC_t + I_t$$

- The average of the detrended value for a given month (for monthly data) or given quarter (for quarterly data) will be the seasonal index for the corresponding month or quarter.

Seasonal Adjustment

- For Multiplicative decomposition, the seasonally adjusted data are computed by dividing the original observation by the seasonal component.

$$\frac{y}{S_t} = TC_t \times I_t$$

De-seasonalizing Data

- The process of de-seasonalizing the data has useful results:
  - The seasonalized data allow us to see better the underlying pattern in the data.
  - It provides us with measures of the extent of seasonality in the form of seasonal indexes.
  - It provides us with a tool in projecting what one quarter's (or month's) observation may portend for the entire year.

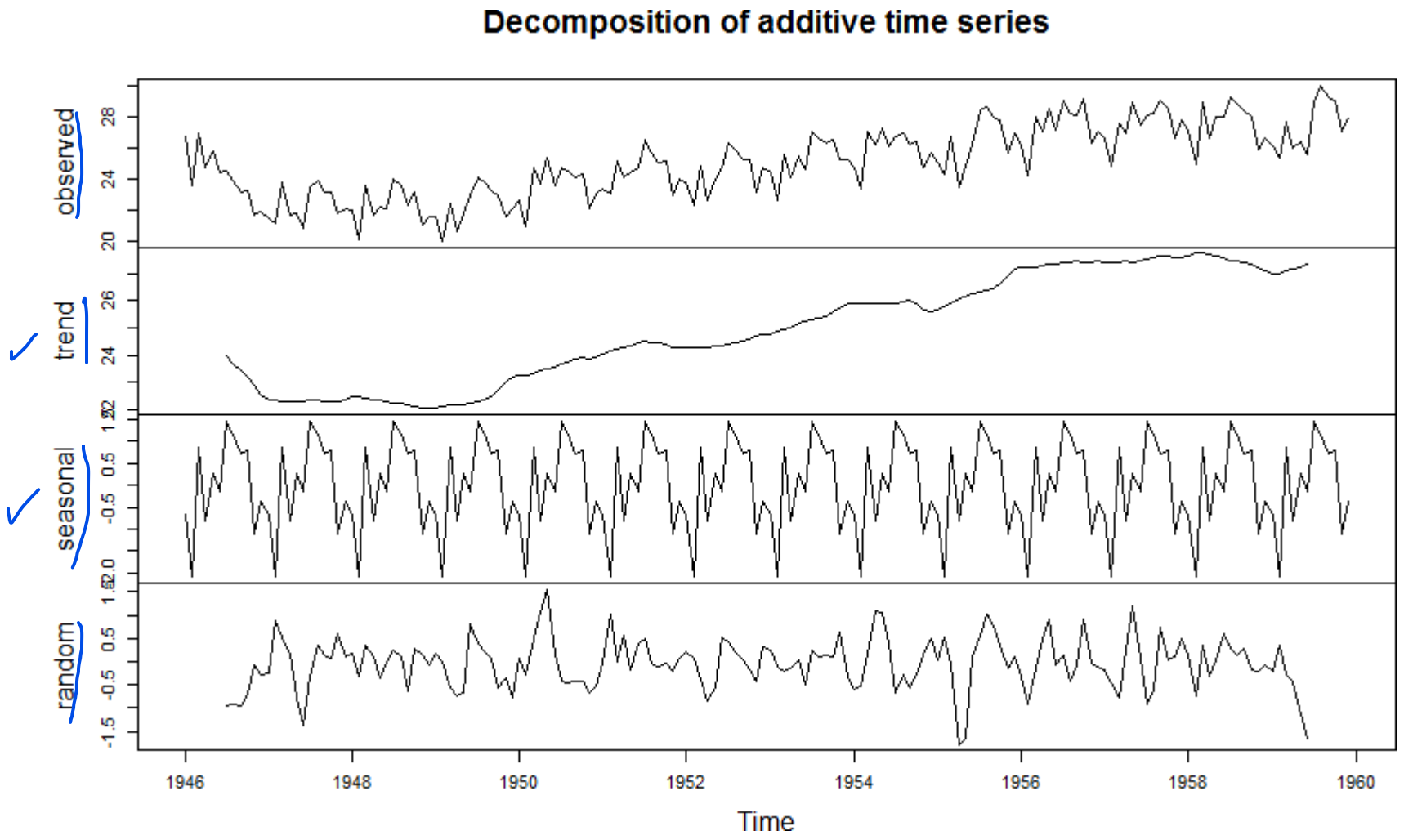
Other Ways to Decompose Trend and Seasonality

- There are more known functions in R that can be used to decompose time series data
- `decompose()`
- `tslm()`
- `stl()`

R Code: Decomposition 1

- `birthstimeseriescomponents <- decompose(birthsts)`
- `plot(birthstimeseriescomponents)`
- `birthstimeseriescomponents`

R Code: Decomposition 1



R Code: Decomposition 1

$seasonal

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
1946	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1947	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1948	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1949	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1950	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1951	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1952	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1953	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1954	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1955	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1956	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1957	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1958	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444
1959	-0.6771947	-2.0829607	0.8625232	-0.8016787	0.2516514	-0.1532556	1.4560457	1.1645938	0.6916162	0.7752444

$trend

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1946	NA	NA	NA	NA	NA	NA	23.98433	23.66213	23.42333	23.16112	22.86425	22.54521
1947	22.35350	22.30871	22.30258	22.29479	22.29354	22.30562	22.33483	22.31167	22.26279	22.25796	22.27767	22.35400
1948	22.43038	22.43667	22.38721	22.35242	22.32458	22.27458	22.23754	22.21988	22.16983	22.07721	22.01396	22.02604
1949	22.06375	22.08033	22.13317	22.16604	22.17542	22.21342	22.27625	22.35750	22.48862	22.70992	22.98563	23.16346
1950	23.21663	23.26967	23.33492	23.42679	23.50638	23.57017	23.63888	23.75713	23.86354	23.89533	23.87342	23.88150
1951	24.00083	24.12350	24.20917	24.28208	24.35450	24.43242	24.49496	24.48379	24.43879	24.36829	24.29192	24.27642
1952	24.27204	24.27300	24.28942	24.30129	24.31325	24.35175	24.40558	24.44475	24.49325	24.58517	24.70429	24.76017
1953	24.78646	24.84992	24.92692	25.02362	25.16308	25.26963	25.30154	25.34125	25.42779	25.57588	25.73904	25.87513
1954	25.92446	25.92317	25.92967	25.92137	25.89567	25.89458	25.92963	25.98246	26.01054	25.88617	25.67087	25.57312
1955	25.64612	25.78679	25.93192	26.06388	26.16329	26.25388	26.35471	26.40496	26.45379	26.64933	26.95183	27.14683
1956	27.21104	27.21900	27.20700	27.26925	27.35050	27.37983	27.39975	27.44150	27.45229	27.43354	27.44488	27.46996
1957	27.44221	27.40283	27.44300	27.45717	27.44429	27.48975	27.54354	27.56933	27.63167	27.67804	27.62579	27.61212
1958	27.68642	27.76067	27.75963	27.71037	27.65783	27.58125	27.49075	27.46183	27.42262	27.34175	27.25129	27.08558
1959	26.96858	27.00512	27.09250	27.17263	27.26208	27.36033	NA	NA	NA	NA	NA	NA

$random

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
1946	NA	NA	NA	NA	NA	NA	-0.963379006	-0.925718750	-0.939949519
1947	-0.237305288	0.863252404	0.543893429	0.175887019	-0.793193109	-1.391369391	-0.311879006	0.347739583	0.150592147
1948	0.183819712	-0.318705929	0.340268429	0.121262019	-0.354234776	0.001672276	0.256412660	0.119531250	-0.623449519

R Code: Decomposition 1

- `birthstimeseriescomponents <- decompose(birthsts)`
- `plot(birthstimeseriescomponents)`
- `birthstimeseriescomponents`

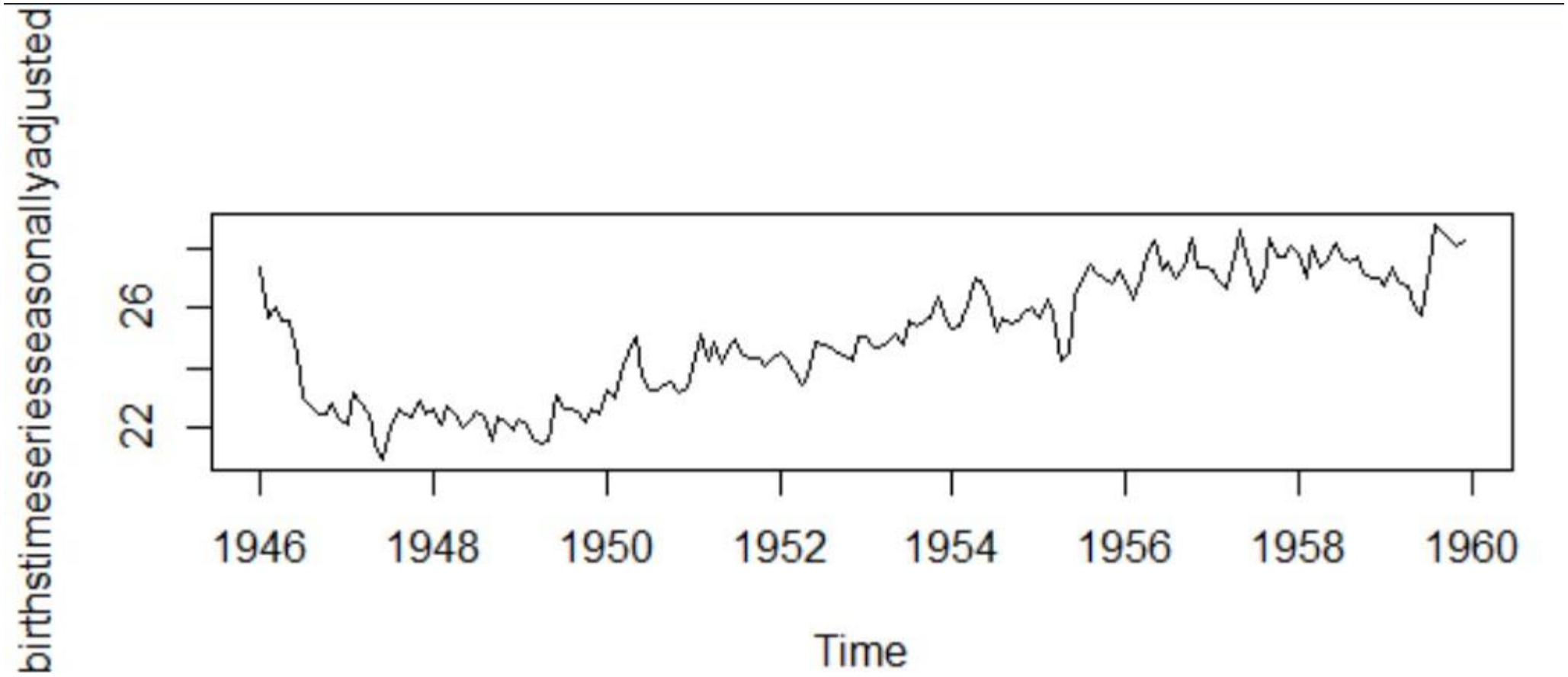
Seasonal Adjustment in R

- Seasonal time series that can be described using an additive model, adjust by estimating the seasonal component and subtracting it from the original time series
- Thus, removing the seasonal variation from the original series containing only trend and irregular components

R Code: Seasonal Adjustment Decomposition 1

- `birthstimeseriescomponents <- decompose(birthsts)`
- `birthstimeseriesseasonallyadjusted <- birthsts - birthstimeseriescomponents$seasonal`
- `birthstimeseriesseasonallyadjusted`
- `plot(birthstimeseriesseasonallyadjusted)`

R Code: Seasonal Adjustment Decomposition 1

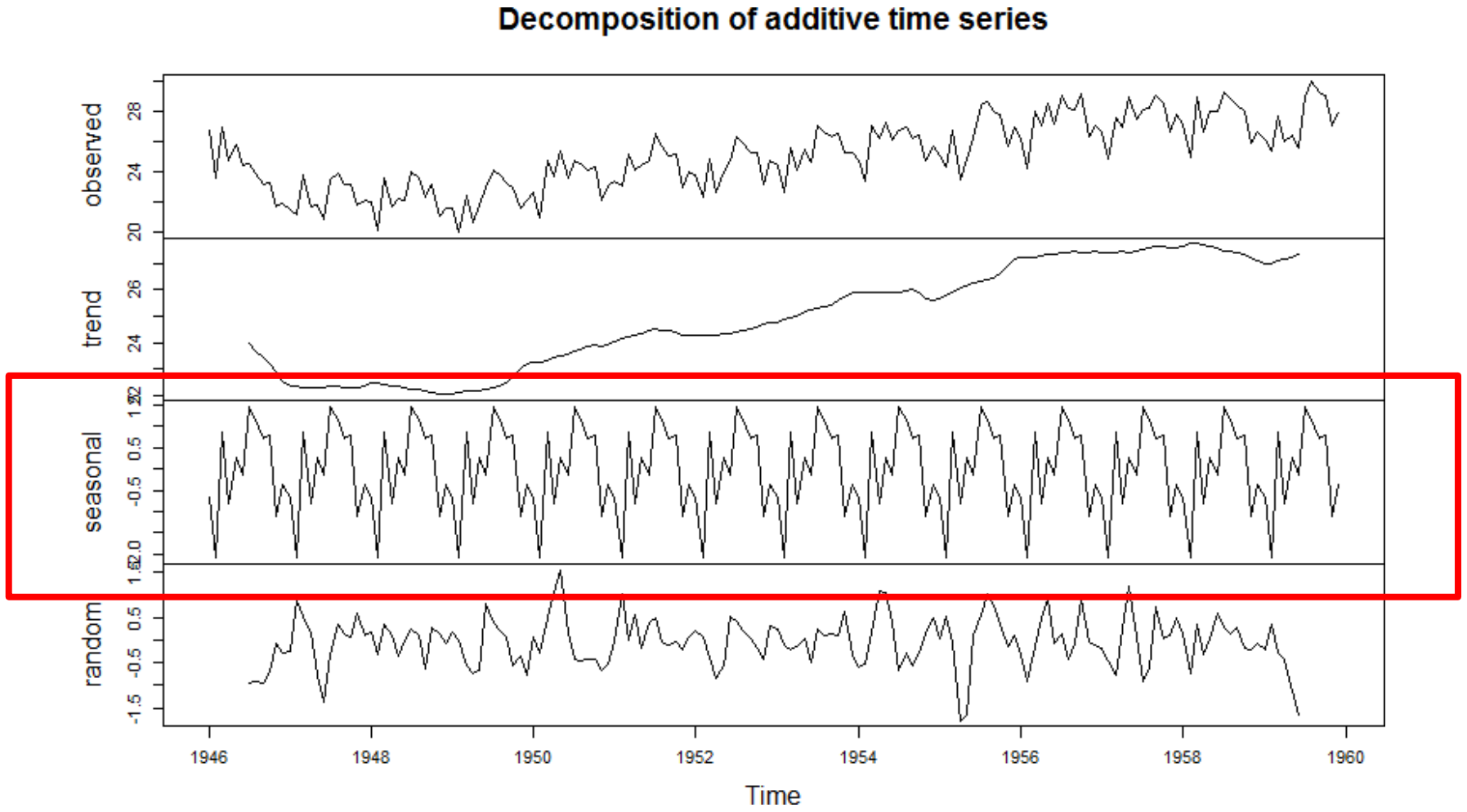


Notes on Seasonality

- There maybe different ways to get the seasonal components in a decomposition method
- However, values will converge in similar plots

Notes on Seasonality

- Decomposing using decompose in R



Notes on Seasonality

- Decomposing using “decompose” in R

```
$seasonal
            Jan        Feb       Mar        Apr       May        Jun       Jul       Aug       Sep       Oct
1946 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1947 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1948 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1949 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1950 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1951 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1952 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1953 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1954 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1955 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1956 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1957 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1958 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
1959 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444
```

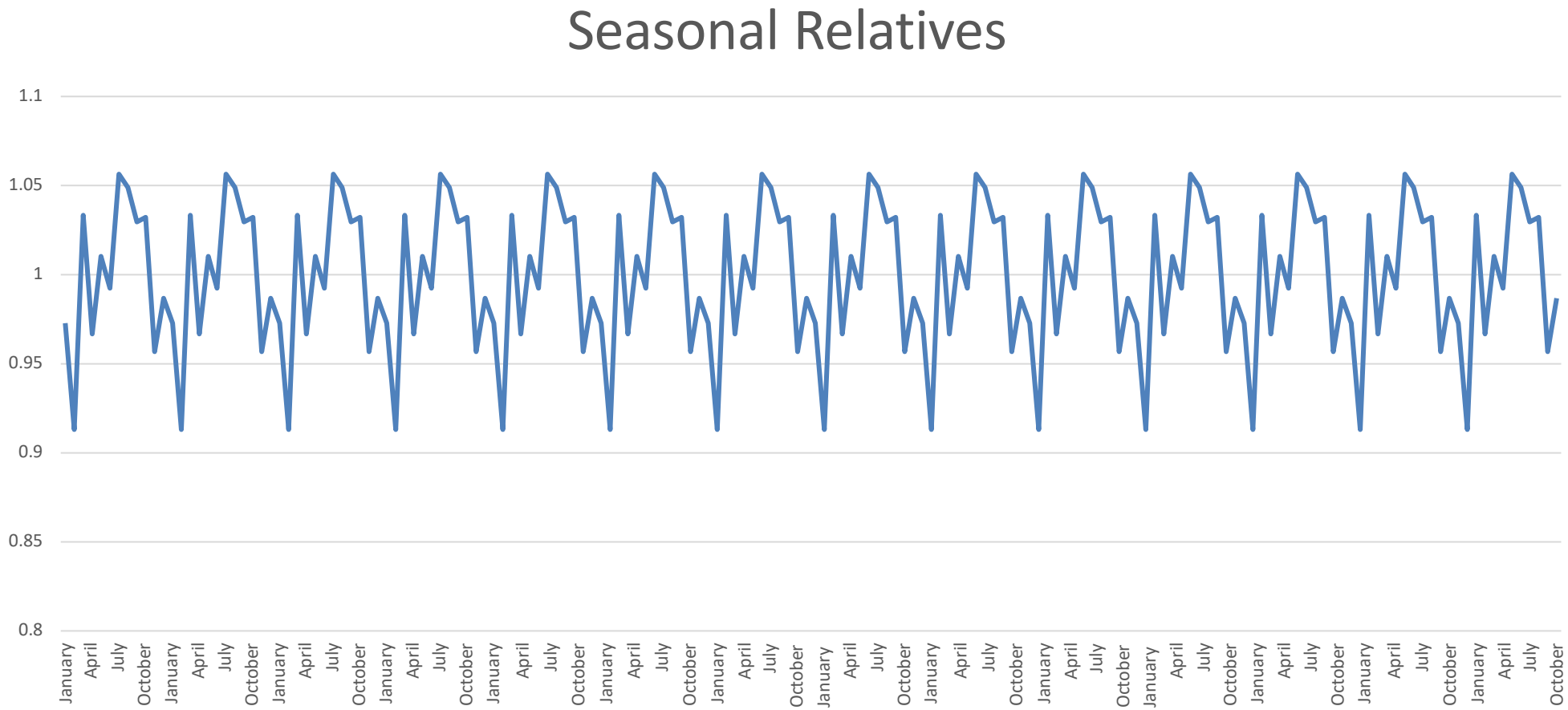
Notes on Seasonality

- Decomposing using seasonal relatives from yesterday

Month	SR
January	0.972838
February	0.913256
March	1.033349
April	0.966846
May	1.010235
June	0.992468
July	1.056422
August	1.048989
September	1.029643
October	1.032263
November	0.95687
December	0.986822

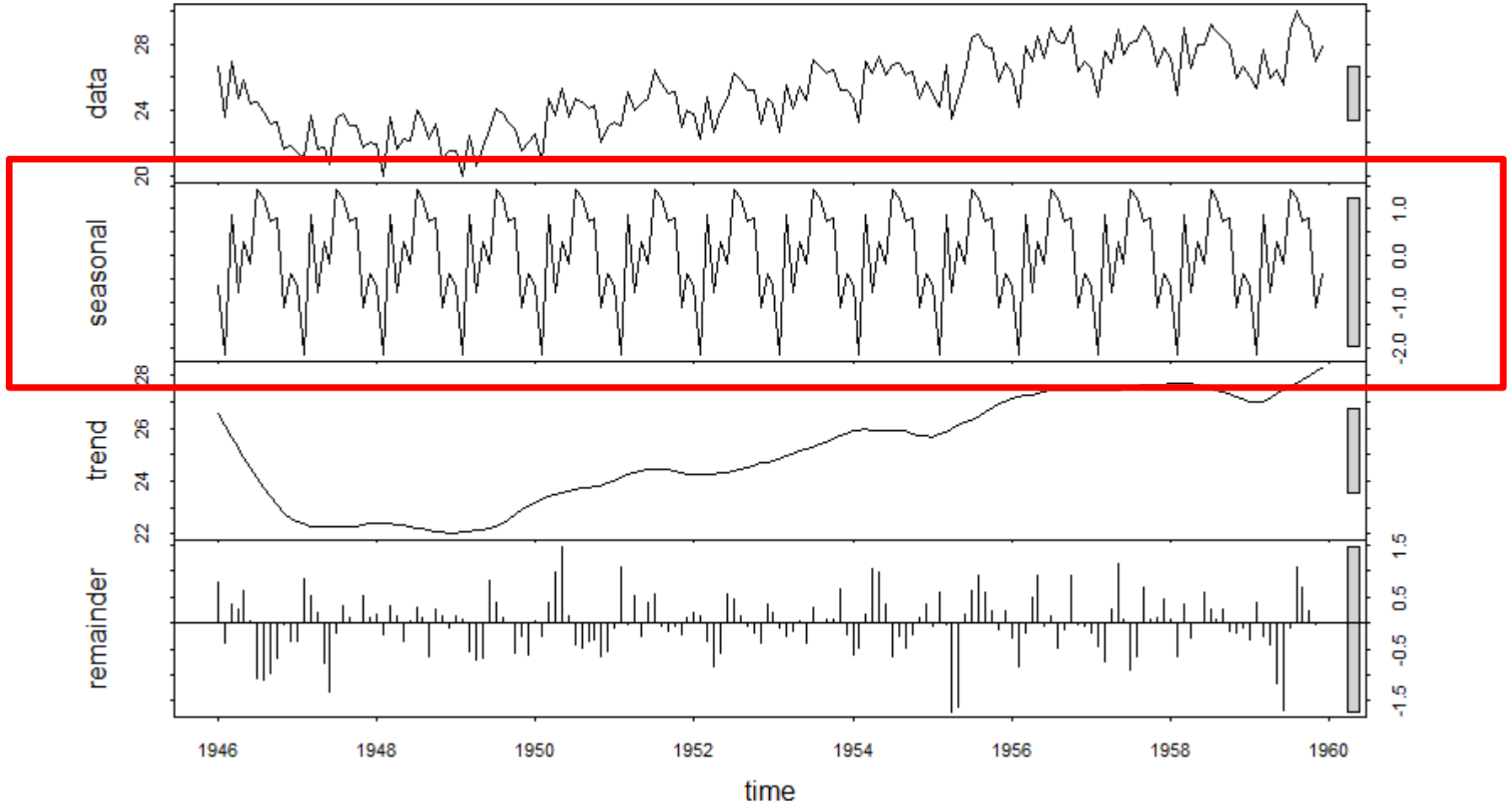
Notes on Seasonality

- Decomposing using seasonal relatives



Notes on Seasonality

- Decomposing using “stl” (Seasonal Trend using Loess)



Notes on Seasonality

- Decomposing using stl (Seasonal Trend using Loess)

Components

	seasonal	trend	remainder
Jan 1946	-0.6434274	26.53282	0.7736051527
Feb 1946	-2.1371410	26.11345	-0.3783113276
Mar 1946	0.8716451	25.69408	0.3652725397
Apr 1946	-0.8054689	25.29487	0.2505942245
May 1946	0.2712028	24.89567	0.6391302125
Jun 1946	-0.1817308	24.51802	0.0277121005
Jul 1946	1.4131922	24.14037	-1.0765626353
Aug 1946	1.2202346	23.77470	-1.0939302553
Sep 1946	0.7287774	23.40902	-0.9627982002
Oct 1946	0.7814986	23.10814	-0.6626395368
Nov 1946	-1.1207089	22.80726	-0.0145520992
Dec 1946	-0.3980735	22.62753	-0.3594580681
Jan 1947	-0.6434274	22.44780	-0.3653745927
Feb 1947	-2.1371410	22.37464	0.8515005042
Mar 1947	0.8716451	22.30148	0.5358759486
Apr 1947	-0.8054689	22.28716	0.1873104341
May 1947	0.2712028	22.27284	-0.7920407771
Jun 1947	-0.1817308	22.27073	-1.3280029842
Jul 1947	1.4131922	22.26863	-0.2028218152
Aug 1947	1.2202346	22.27383	0.3299339394
Sep 1947	0.7287774	22.27903	0.0971893690

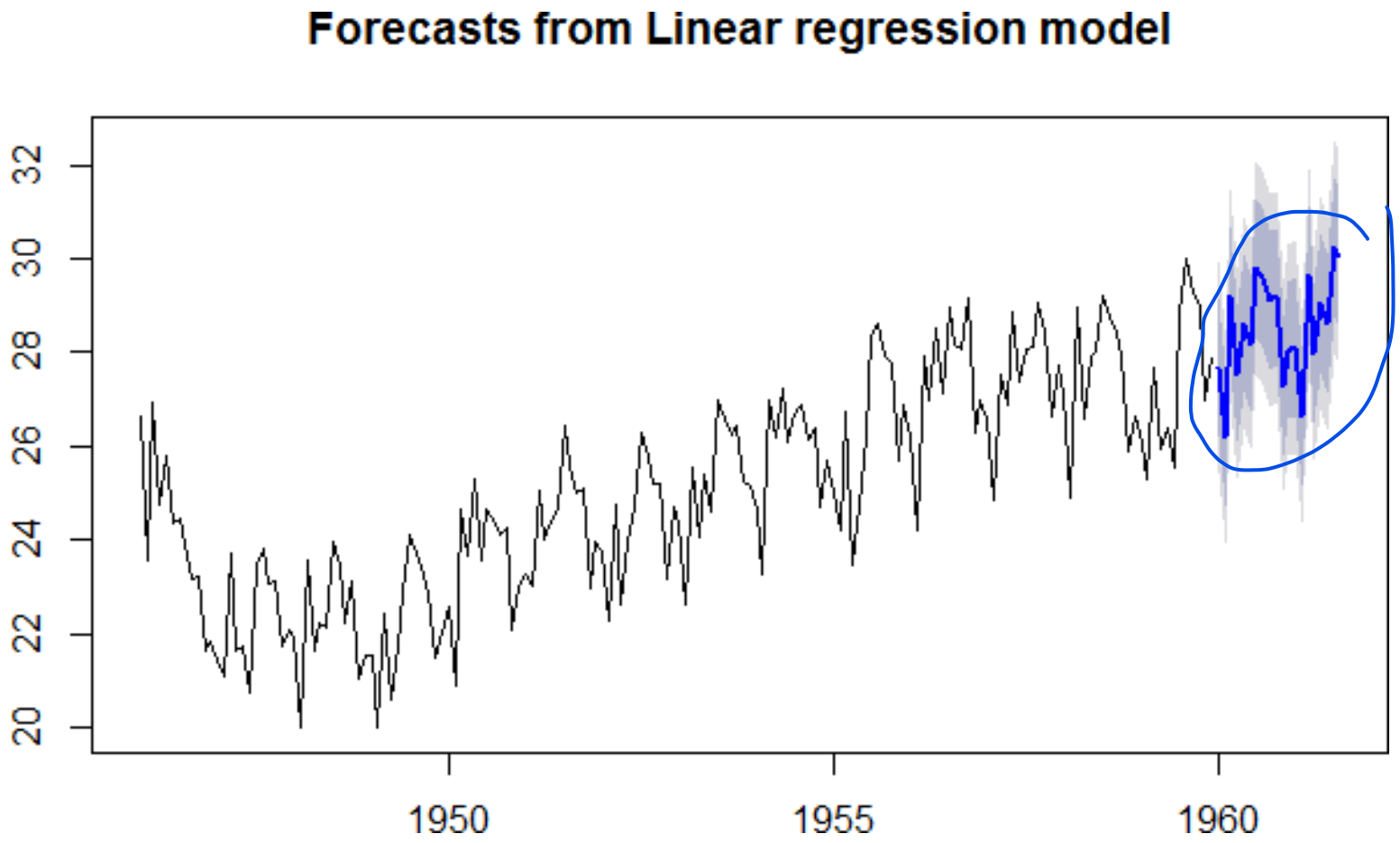
R Code: Decomposition 2

- `fit = tslm(birthsts~trend + season)`
- `summary(fit)`
- `plot(forecast(fit, h=20))`

Model with Seasonal Effects

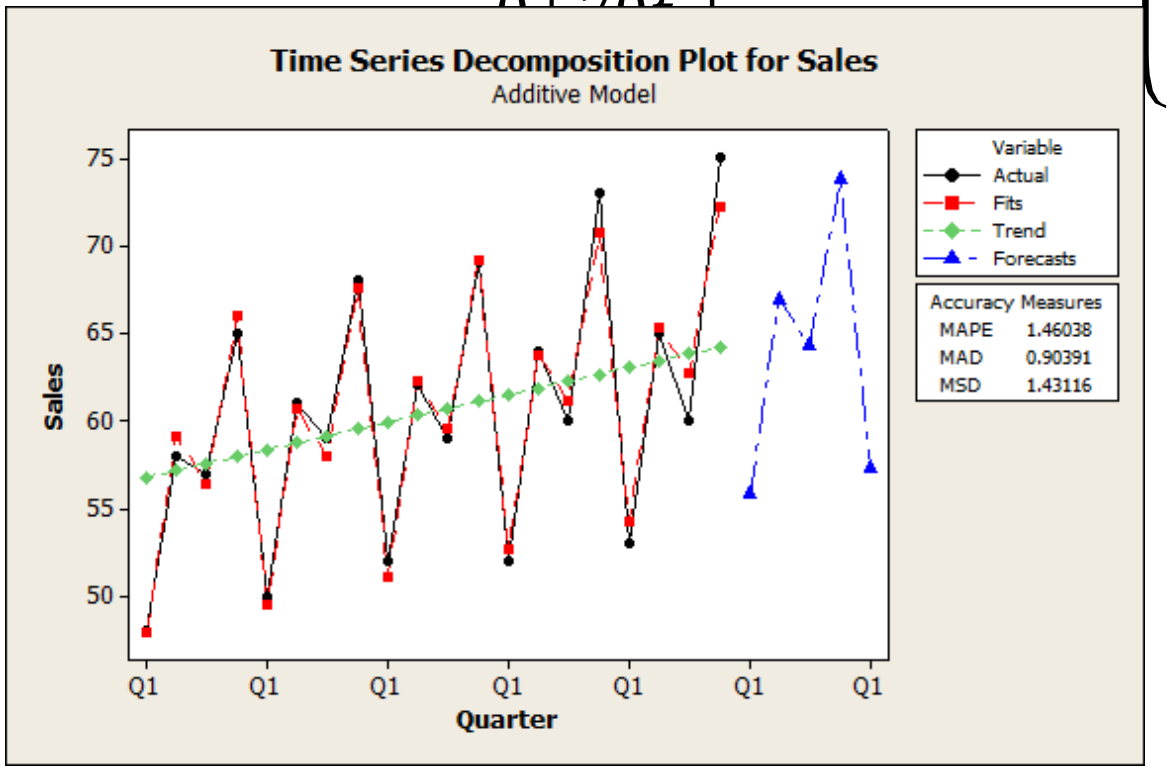
```
Call:
lm(formula = formula, data = "birthsts", na.action = na.exclude)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1819 -0.5458 -0.1180  0.4999  5.1607

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 21.465397   0.323663  66.320  < 2e-16 ***
trend        0.036877   0.001747  21.108  < 2e-16 ***
season2     -1.529948   0.414028  -3.695 0.000304 ***
season3      1.442604   0.414039   3.484 0.000642 ***
season4     -0.260772   0.414058  -0.630 0.529755
season5      0.789637   0.414084   1.907 0.058377 .
season6      0.307546   0.414117   0.743 0.458815
season7      1.873312   0.414157   4.523  1.2e-05 ***
season8      1.650150   0.414205   3.984 0.000104 ***
season9      1.128488   0.414260   2.724 0.007188 **
season10     1.157254   0.414323   2.793 0.005879 **
season11    -0.768908   0.414393  -1.856 0.065423 .
season12    -0.055213   0.414470  -0.133 0.894197
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.095 on 155 degrees of freedom
Multiple R-squared:  0.7929, Adjusted R-squared:  0.7768
F-statistic: 49.44 on 12 and 155 DF,  p-value: < 2.2e-16
```

Trend Analysis using Regression Model



Forecast Predictions With Final Additive Model

$$\hat{y}_t = 54.52 + 0.570t + \begin{cases} -8.644 & if\ Q = 1 \\ 1.785 & if\ Q = 2 \\ -1.75 & if\ Q = 3 \\ 8.645 & if\ Q = 4 \end{cases}$$

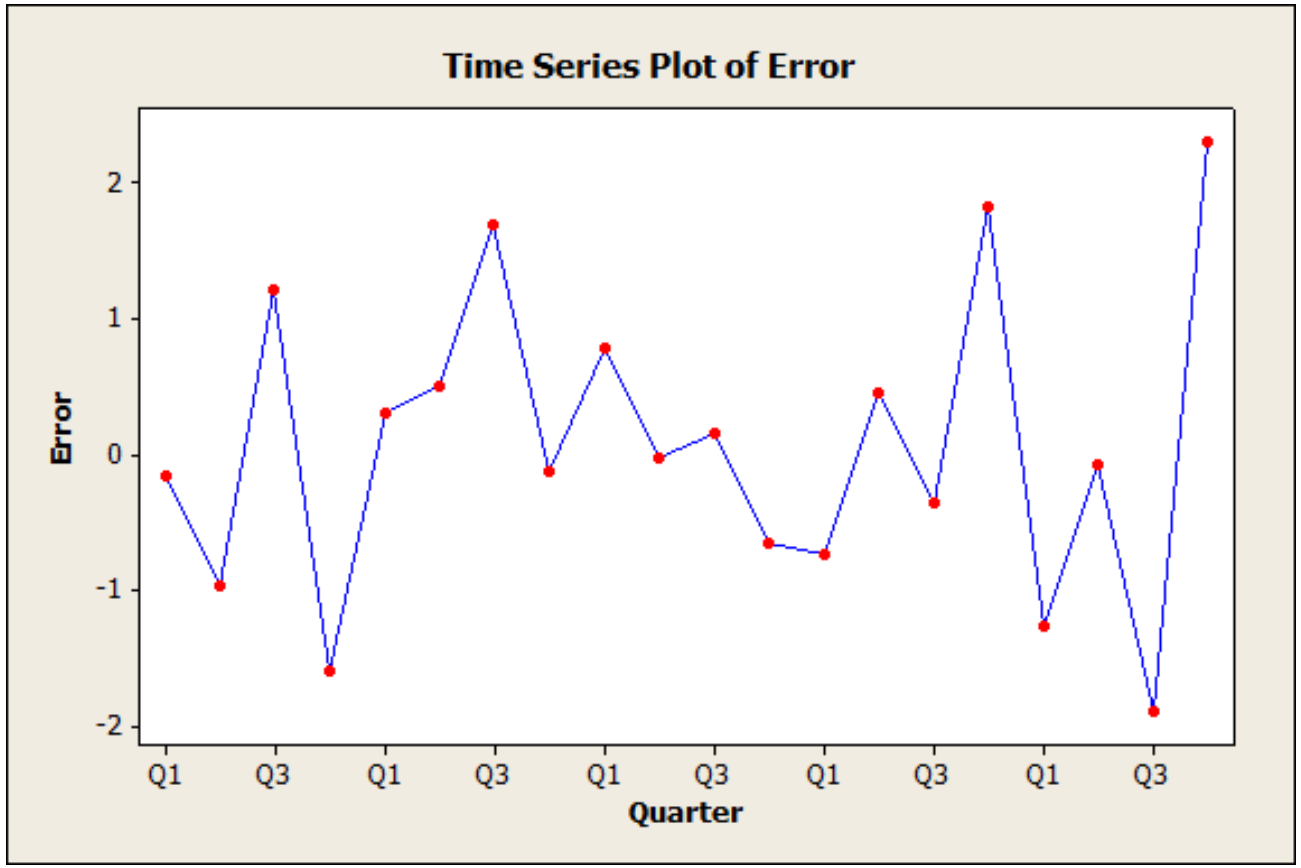


Steps in Decomposition Method

1. Plot the time series data
2. Identify time series model – Additive or Multiplicative
3. Detect the trend
4. De-trend the time series
5. Estimate the seasonality
6. Examine Irregular and Random variations
7. Reconstruct original time series

Error Component Computation (Additive Model)

- $y_t - Tt - S_t = I_t$

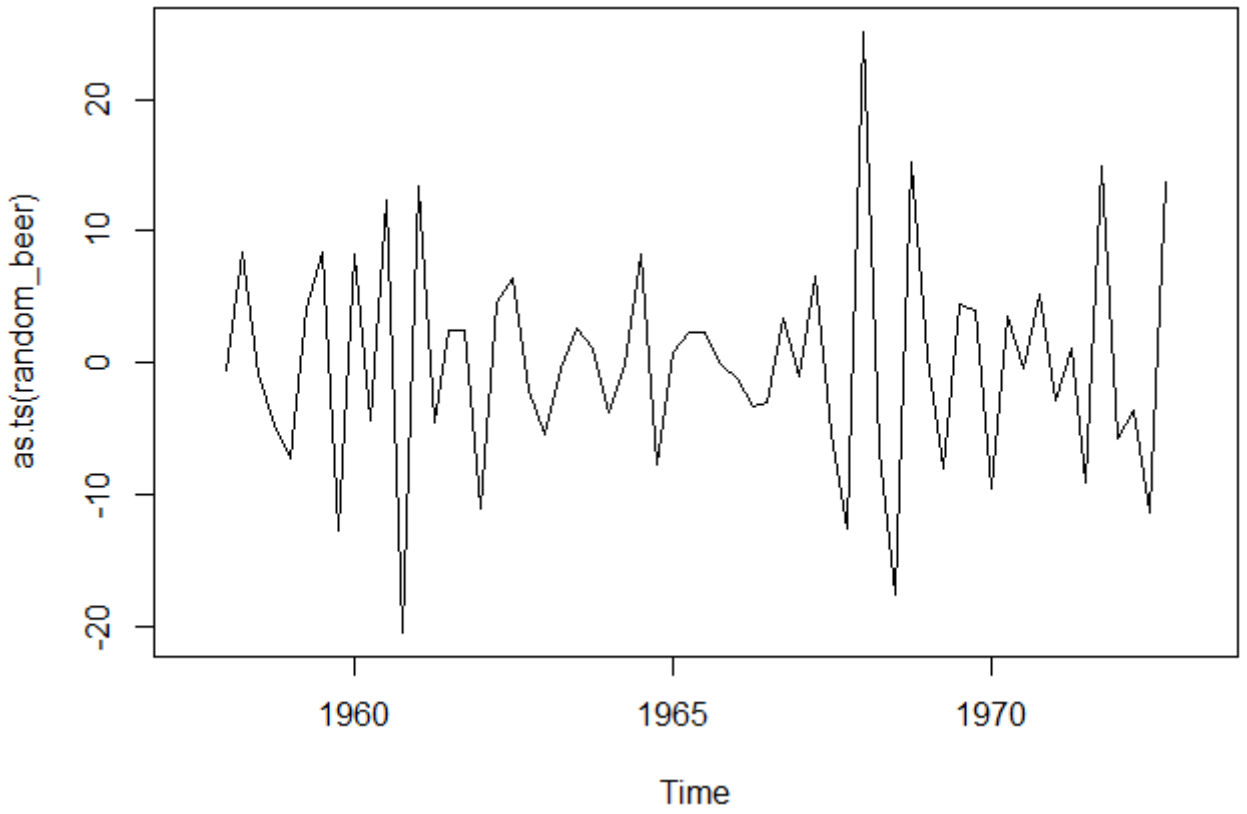


Steps in Decomposition Method

6. Examine irregular and random variations

- `random_beer = timeserie_beer - trend_beer - seasonal_beer`
- `plot(as.ts(random_beer))`

Steps in Decomposition Method




```
> random_beer
             Qtr1        Qtr2        Qtr3        Qtr4
1957                                    NA          NA
1958  -0.5500000   8.3916667  -0.8583333  -4.6333333
1959  -7.3000000   4.1416667   8.3916667 -12.6333333
1960   8.3250000  -4.3583333  12.3916667 -20.3833333
1961  13.4500000  -4.4833333   2.5166667   2.4916667
1962 -11.0500000   4.6416667   6.3916667  -2.0083333
1963  -5.4250000  -0.3583333   2.6416667   1.1166667
1964  -3.6750000  -0.1083333   8.2666667  -7.7583333
1965   0.8250000   2.2666667   2.2666667  -0.1333333
1966  -1.1750000  -3.2333333  -2.9833333   3.3666667
1967  -1.0500000   6.6416667  -5.2333333 -12.5083333
1968  25.0750000  -6.6083333 -17.4833333  15.2416667
1969   0.5750000  -7.9833333   4.5166667   3.9916667
1970  -9.5500000   3.5166667  -0.3583333   5.2416667
1971  -2.8000000   1.1416667  -9.1083333  14.8666667
1972  -5.6750000  -3.6083333 -11.3583333  13.7416667
1973          NA          NA
```

Steps in Decomposition Method

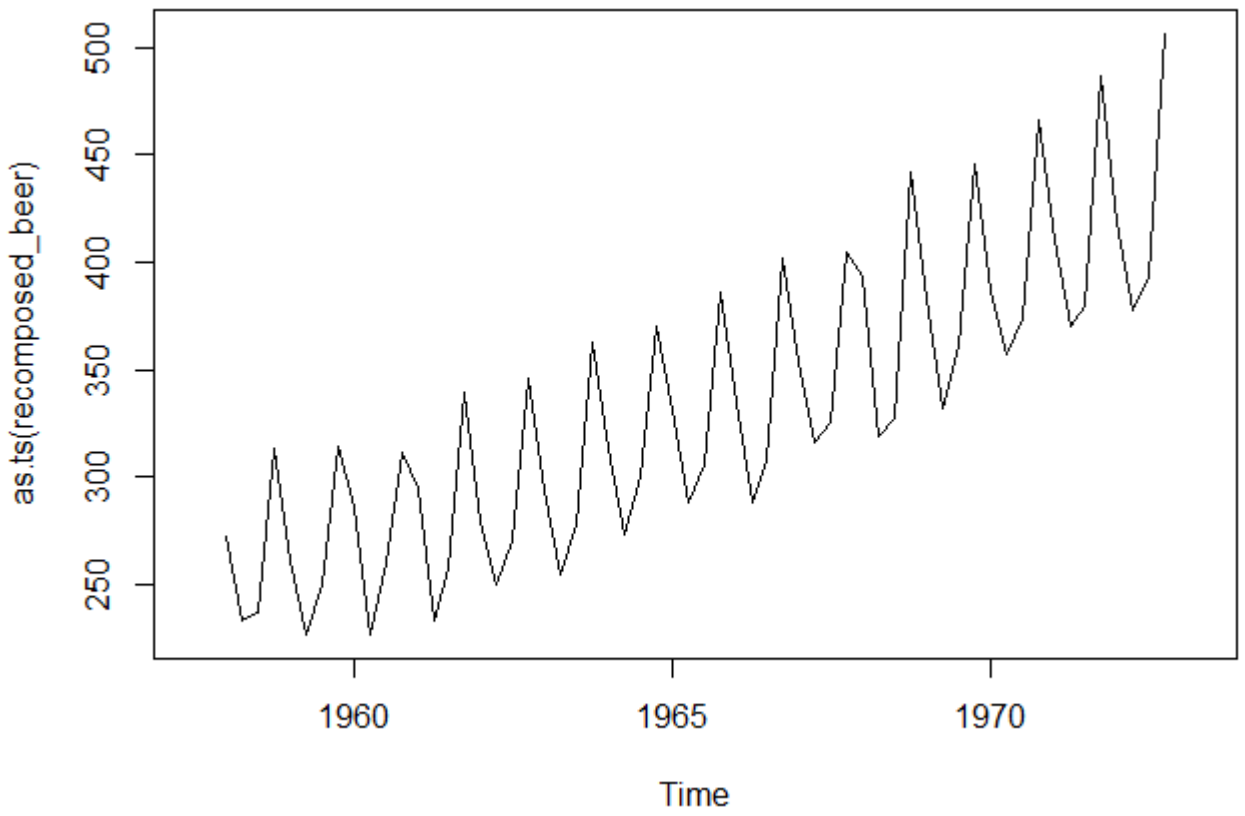
1. Plot the time series data
2. Identify time series model – Additive or Multiplicative
3. Detect the trend
4. De-trend the time series
5. Estimate the seasonality
6. Examine Irregular and Random variations
7. Reconstruct original time series

Steps in Decomposition Method

7. Reconstruct original time series

- `recomposed_beer = trend_beer+seasonal_beer+random_beer`
- `recomposed_beer`
- `recomposed_beer - ausbeer`
- `plot(as.ts(recomposed_beer))`

Steps in Decomposition Method




```
> recomposed_beer
     Qtr1 Qtr2 Qtr3 Qtr4
1957                   NA   NA
1958  272  233  237  313
1959  261  227  250  314
1960  286  227  260  311
1961  295  233  257  339
1962  279  250  270  346
1963  294  255  278  363
1964  313  273  300  370
1965  331  288  306  386
1966  335  288  308  402
1967  353  316  325  405
1968  393  319  327  442
1969  383  332  361  446
1970  387  357  374  466
1971  410  370  379  487
1972  419  378  393  506
1973   NA   NA
```

How to Choose?

Forecasting Method	Amount of Historical Data	Data Pattern	Forecast Horizon
Linear Regression, Triple Exponential Smoothing	10 to 20 observations, ≤5 observations/season	Stationary, trend and seasonality	Short to Medium
SMA	6 to 12 months, weekly data usually	Data should be stationary (no trend, no season)	Short
WMA, Single Exponential Smoothing	5 to 10 observations to start	Data should be stationary	Short
Double Exponential Smoothing	5 to 10 observations to start	Stationary and with trend	Short

Case Study in R

- Given Chocolates.csv and Airlines.csv dataset.
- See Case 3 for the instructions.

Some Notes

1. Forecasting makes a statement about the future but it does not tell us the actual future.
2. Forecasting needs regular monitoring and adjustments.
3. Forecasting should help the organization, the business, and the team decide on strategic decisions.
4. Forecasting assists in decision making.
5. Forecasting is just a subset of Business Analytics.

References

- Notes and Datasets from Montgomery, Peck and Vining, Introduction to Linear Regression Analysis 4th Ed. Wiley
- Notes from G. Runger, ASU IEE 578
- Trevor Hastie, Rob Tibshirani, Friedman: Elements of Statistical Learning (2nd Ed.) 2009
- Time Series Data Library: Australian Bureau of Statistics
- http://datamarket.com/data/list/?q=provider:tsdl
- For R: http://robjhyndman.com/hyndsight/r/
- Time Series Data in R - http://www.rdatamining.com

References

- Duong Tuan Anh, Faculty of Computer Science and Engineering, September 2011
- http://faculty.wiu.edu/F-Dehkordi/DS-533/Lectures/Multiple%20Regression.ppt
- http://faculty.wiu.edu/F-Dehkordi/DS-533/Lectures/The%20Box-Jenkins%20Methodology%20for%20RIMA%20Models.ppt
- www.cse.hcmut.edu.vn/~dtanh/download/TS_PartI_new.ppt

R

- R: http://cran.r-project.org/bin/windows/base/R-3.1.3-win.exe
- R Studio: http://download1.rstudio.org/RStudio-0.98.1103.exe
- Make sure to install R first before R Studio which is a GUI of R.
- R is the most widely used Data Mining tool since it is fast, comprehensive and free.
- A free software environment for statistical computing and graphics
- http://cran.r-project.org/web/views/TimeSeries.html