

NATIONAL ENGINEERING CENTER

University of the Philippines
Diliman, Quezon City



Forecasting and Time Series Analysis with R

Day 3

Raymond Freth A. Lagria

Instructor

Department Industrial Engineering and Operations Research

University of the Philippines Diliman

*Module 5 of the Business Intelligence and Analytics Track of
UP NEC and the UP Center of Business Intelligence*

Outline for This Training

1. Introduction to Forecasting in Business Intelligence
2. Demand Forecasting Techniques
 - Qualitative
 - Quantitative
3. Accuracy of Forecasts
4. Monitoring of Forecasts
5. Forecasting with R
6. Introduction to Time Series Data Mining
7. Advanced Time Series



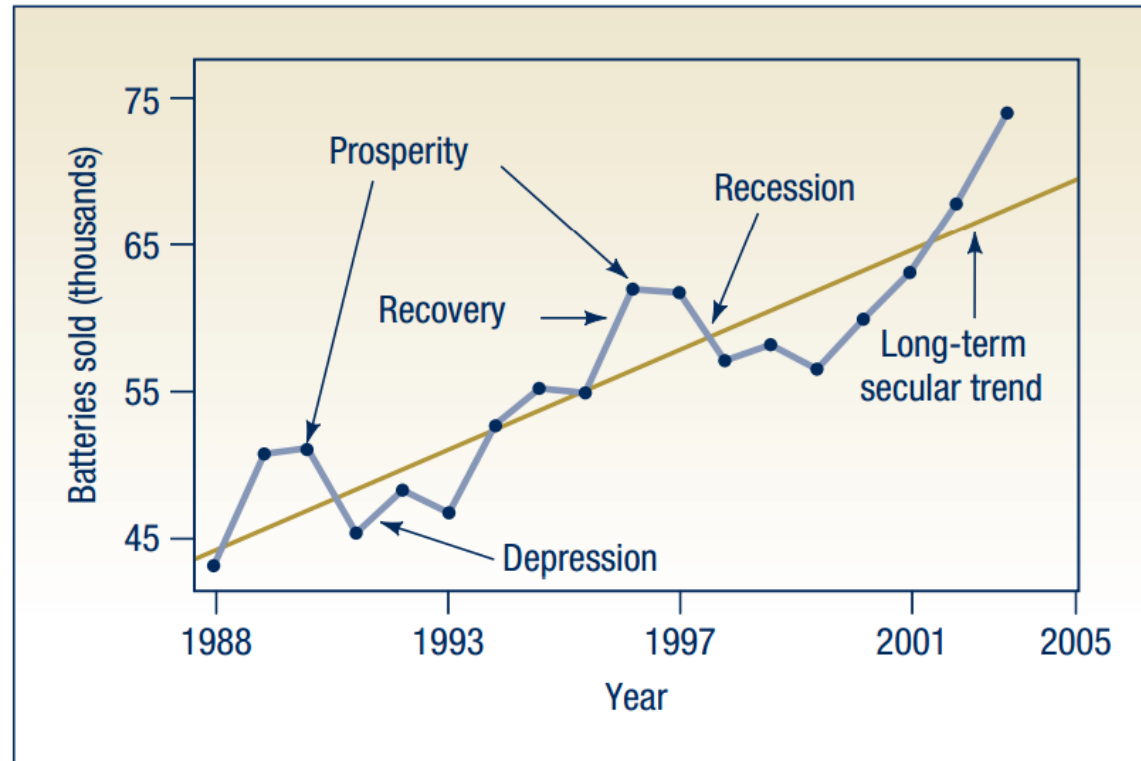
Outline for This Training

1. Introduction to Forecasting in Business Intelligence
2. Demand Forecasting Techniques
 - Qualitative
 - Quantitative
3. Accuracy of Forecasts
4. Monitoring of Forecasts
- 5. Forecasting with R**
6. Introduction to Time Series Data Mining
7. Advanced Time Series



Quick Review

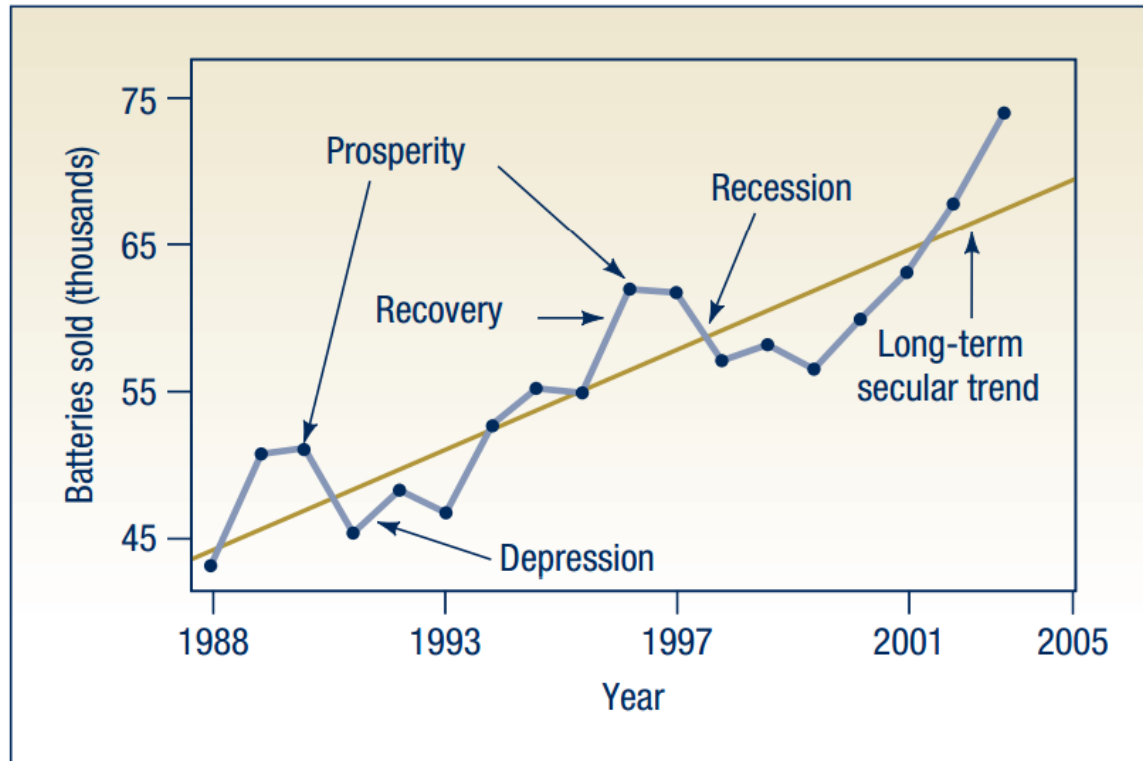
- A time series is a collection of observations made **sequentially in time**.



- 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

General Types of Forecasting

- Qualitative
 - Delphi
 - Sales Force
 - Executive Opinion
 - Consumer Market Survey
- Quantitative
 - Time Series
 - Causal / Associative



Qualitative

- Delphi Method
 - Panel of experts questioned on their perceptions on future events
- Executive Opinion
 - Views of executives or experts from specified departments
- Consumer Surveys
 - Interviews
- Sales Force Polling
 - Insights from sales people

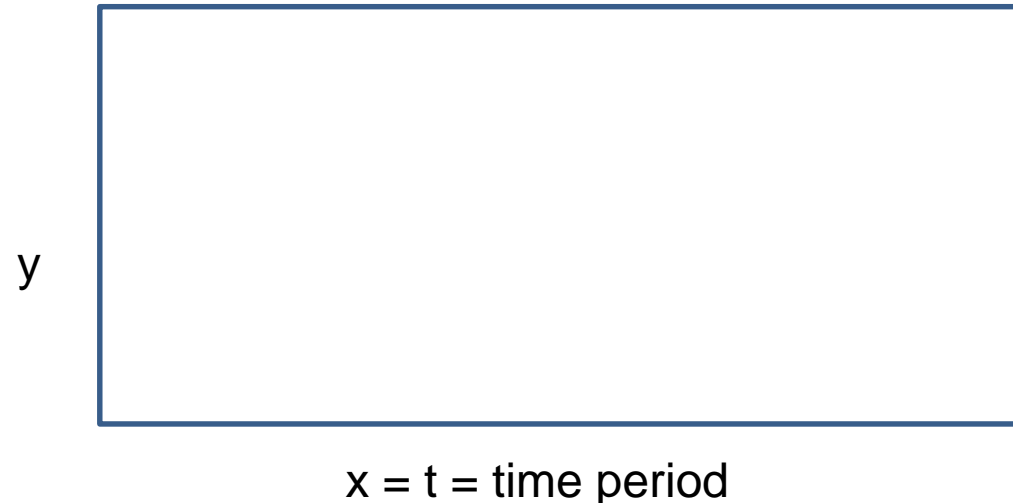
Quantitative

- Associative Forecasting
- Time-series Forecasting Methods
 - Naïve Method
 - Simple Moving Average
 - Weighted Moving Average
 - Exponential Smoothing
 - Trend Adjusted Exponential Smoothing
 - Linear Trend / Linear Regression
 - Decomposition Method



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.

Time Series

- Time-series Examples
 - Annual rainfall
 - GDP each quarter
 - Daily stock market index
 - Number of students recorded per year
 - Population index per census



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.



The Effect of the Smoothing Factor

Period (<i>t</i>)	Actual Demand	$\alpha = .10$ Forecast	$\alpha = .40$ Forecast
1	42	—	—
2	40	42	42
3	43	41.8	41.2
4	40	41.92	41.92
5	41	41.73	41.15
6	39	41.66	41.09
7	46	41.39	40.25
8	44	41.85	42.55
9	45	42.07	43.13
10	38	42.35	43.88
11	40	41.92	41.53
12		41.73	40.92

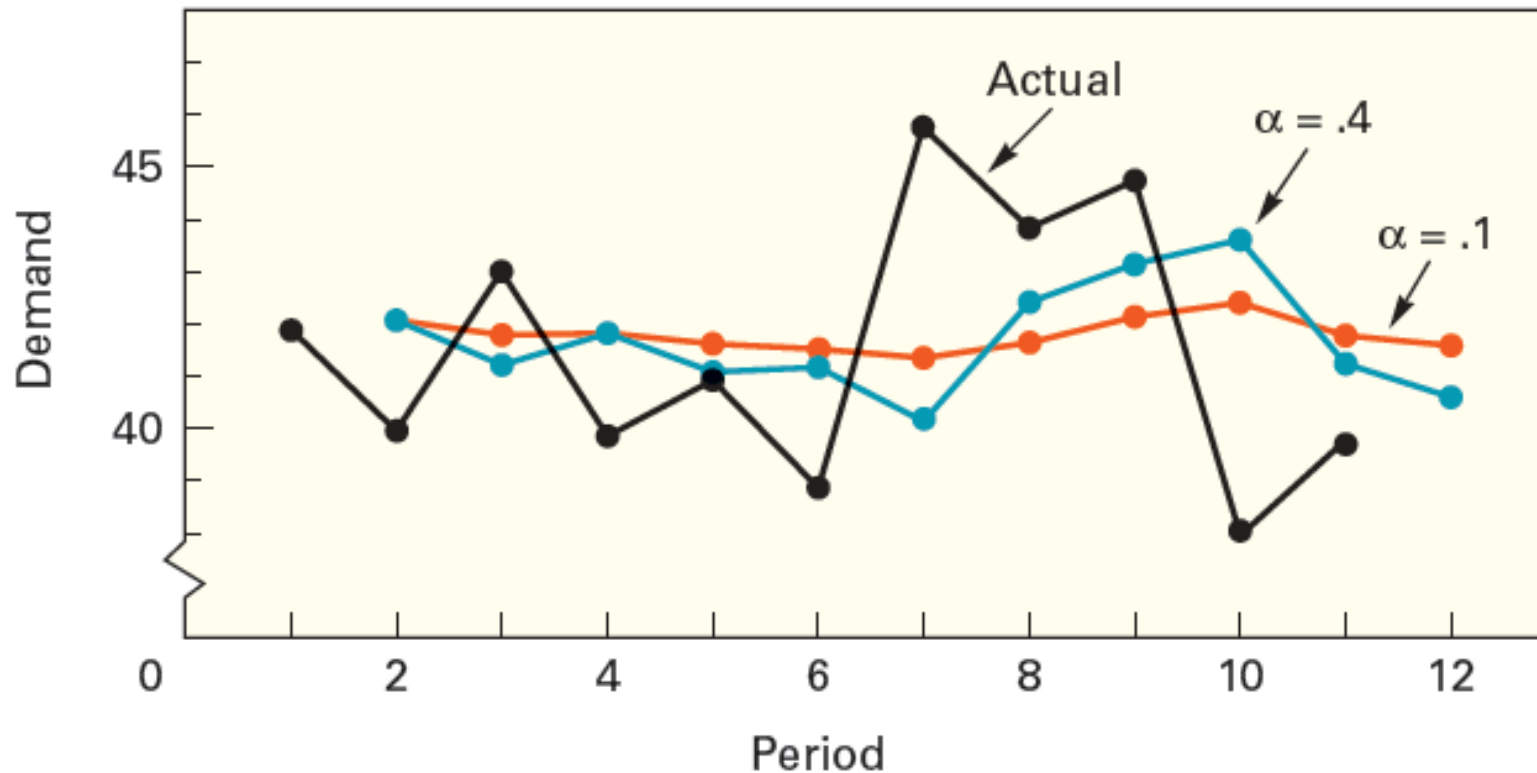


The Effect of the Smoothing Factor

What does the Smoothing Factor
tells us?



The Effect of the Smoothing Factor



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

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

Averaging Methods

- Centered Moving Average
 - Useful in getting overall idea of trends
- Centered Moving Average (*n* is odd)


$$\hat{y}_t = \frac{y_{t+\frac{n-1}{2}} + y_{t+\frac{n-1}{2}-1} + \cdots + y_t + \cdots + y_{t-\frac{n-1}{2}}}{n}$$

- Centered Moving Average (*n* is even)

$$\hat{y}_t = \frac{y_{t+\frac{n}{2}} + y_{t+\frac{n}{2}-1} + \cdots + y_{t-\frac{n}{2}+1}}{2n} + \frac{y_{t+\frac{n}{2}-1} + y_{t+\frac{n}{2}-2} + \cdots + y_{t-\frac{n}{2}}}{2n}$$

Averaging Methods

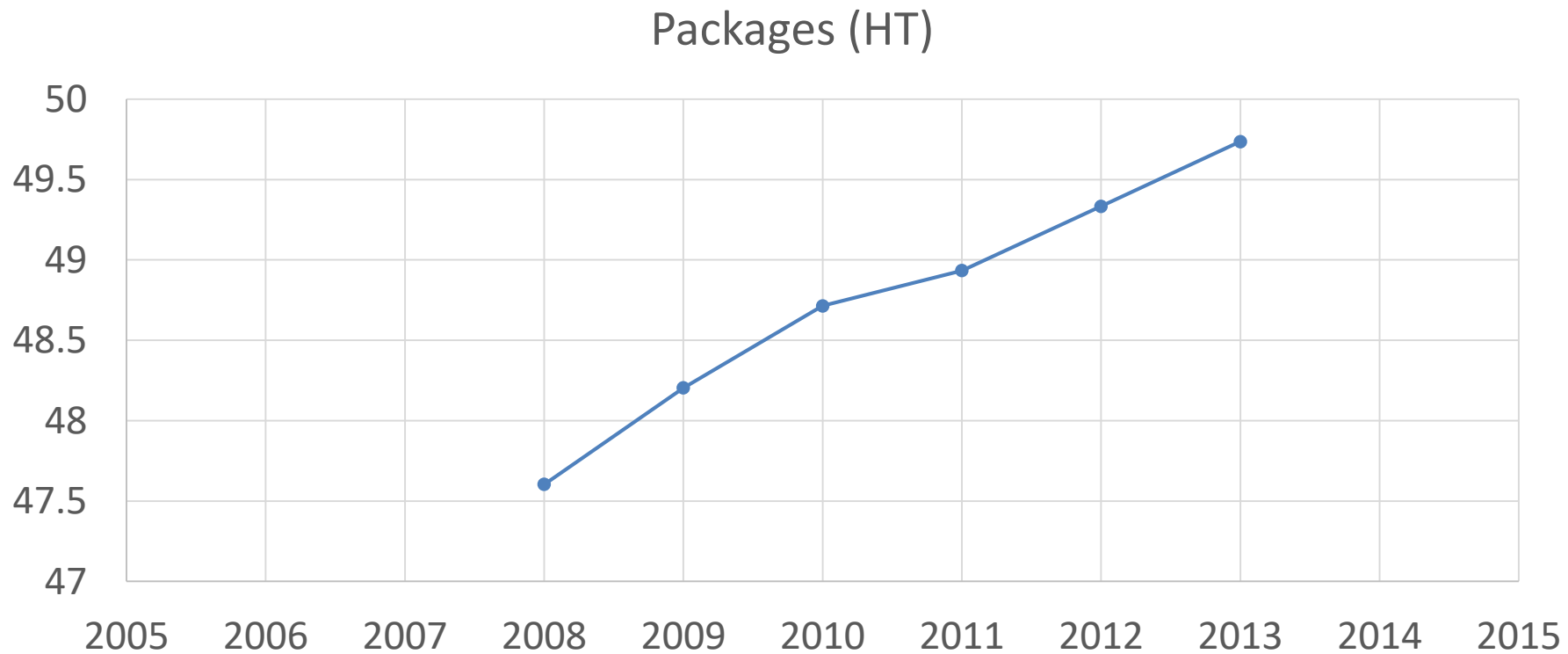
- Example, $n = 5$

Year	# of Small Packages (Hundred Thousands)	5 - CMA
2006	46.16	
2007	46.99	
2008	47.81	
2009	48.31	
2010	48.75	
2011	49.16	
2012	49.54	
2013	48.91	
2014	50.31	
2015	50.76	



Averaging Methods

- Example, $n = 5$



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

- **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

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

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

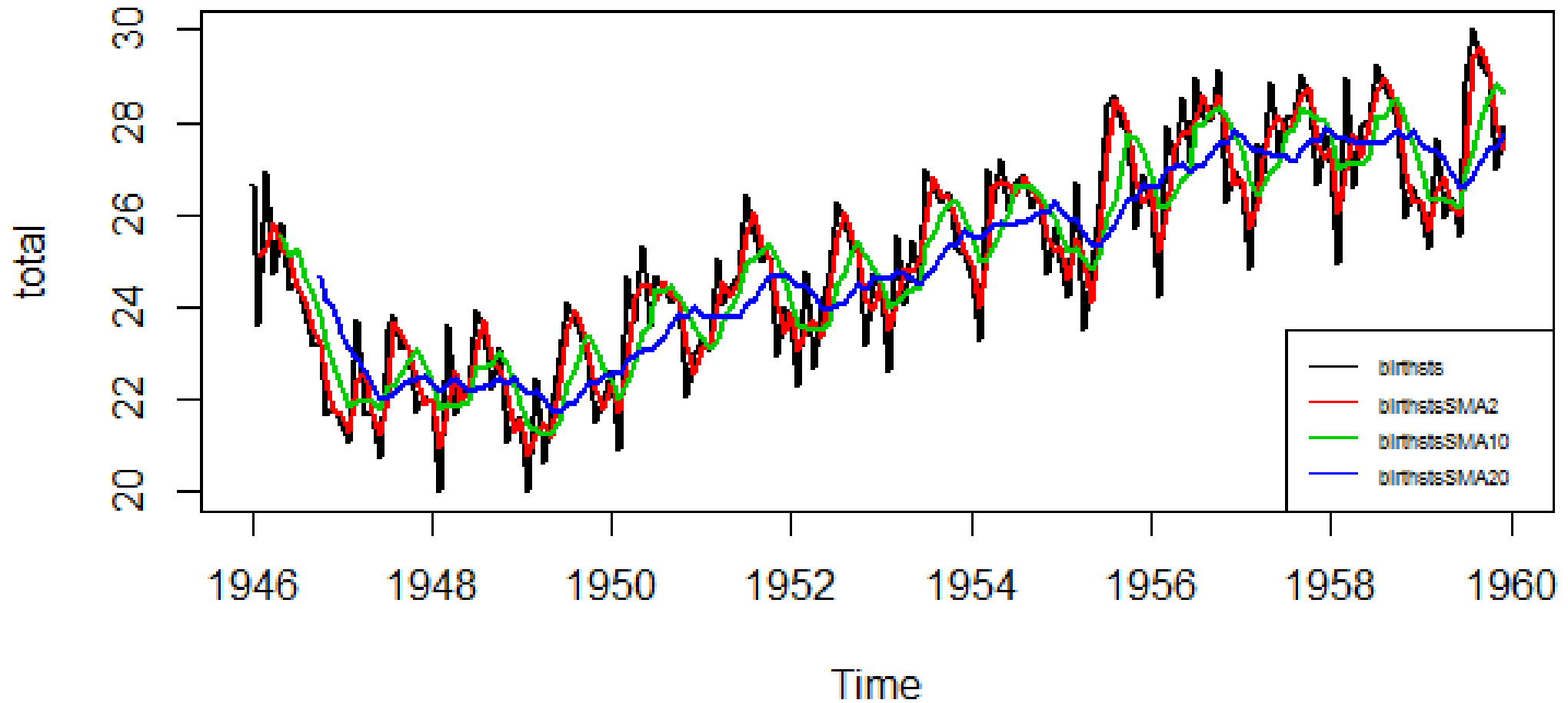
Preliminary: ts

- `install.packages("TTR")`
- `install.packages("forecast")`
- `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 Accuracy

- `accuracy(birthstsSMA2, birthsts)`

```
> 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



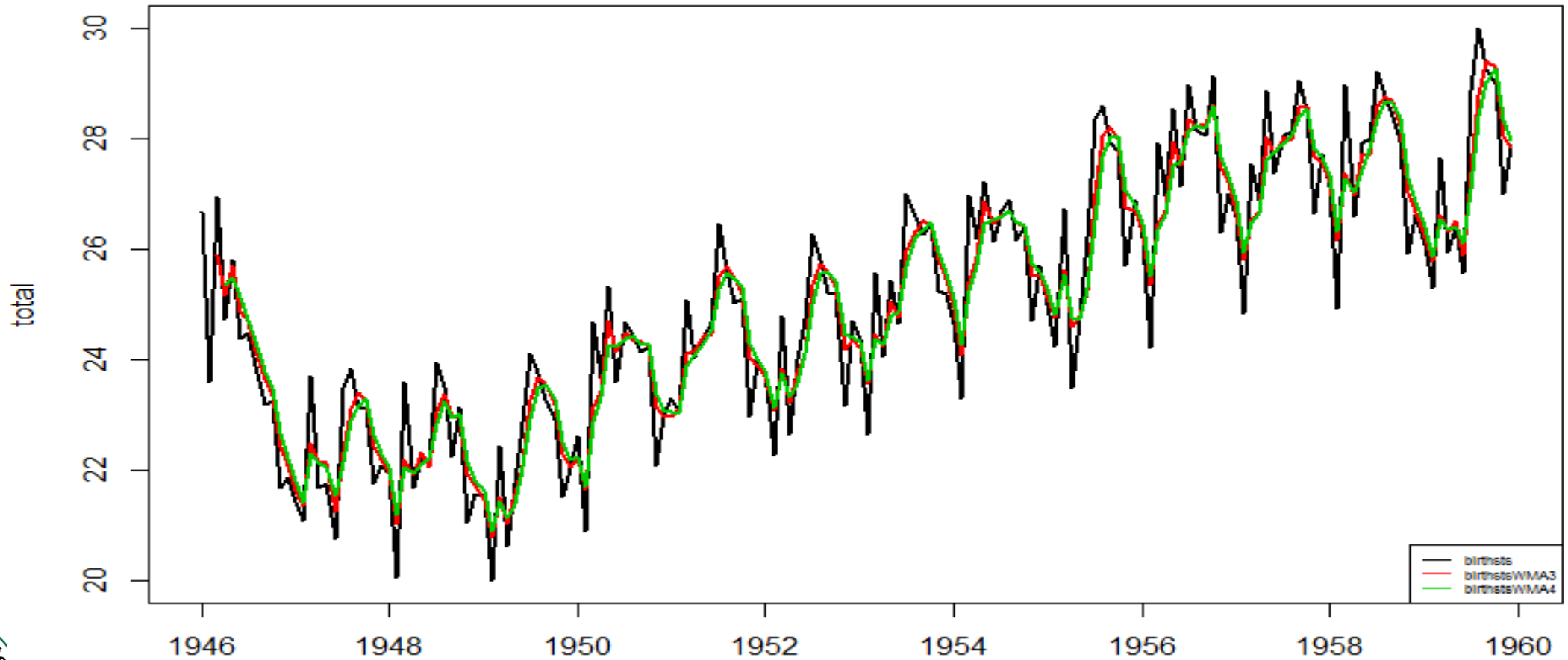
Weighted Moving Average

- 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)`
- `birthstsWMA3 = WMA(birthsts, n=3, wts=xx)`
- `birthstsWMA3`
- `yy <- c(.1, .2, .3, .4)`
- `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



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) + $(1 - \alpha) \times$ 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.

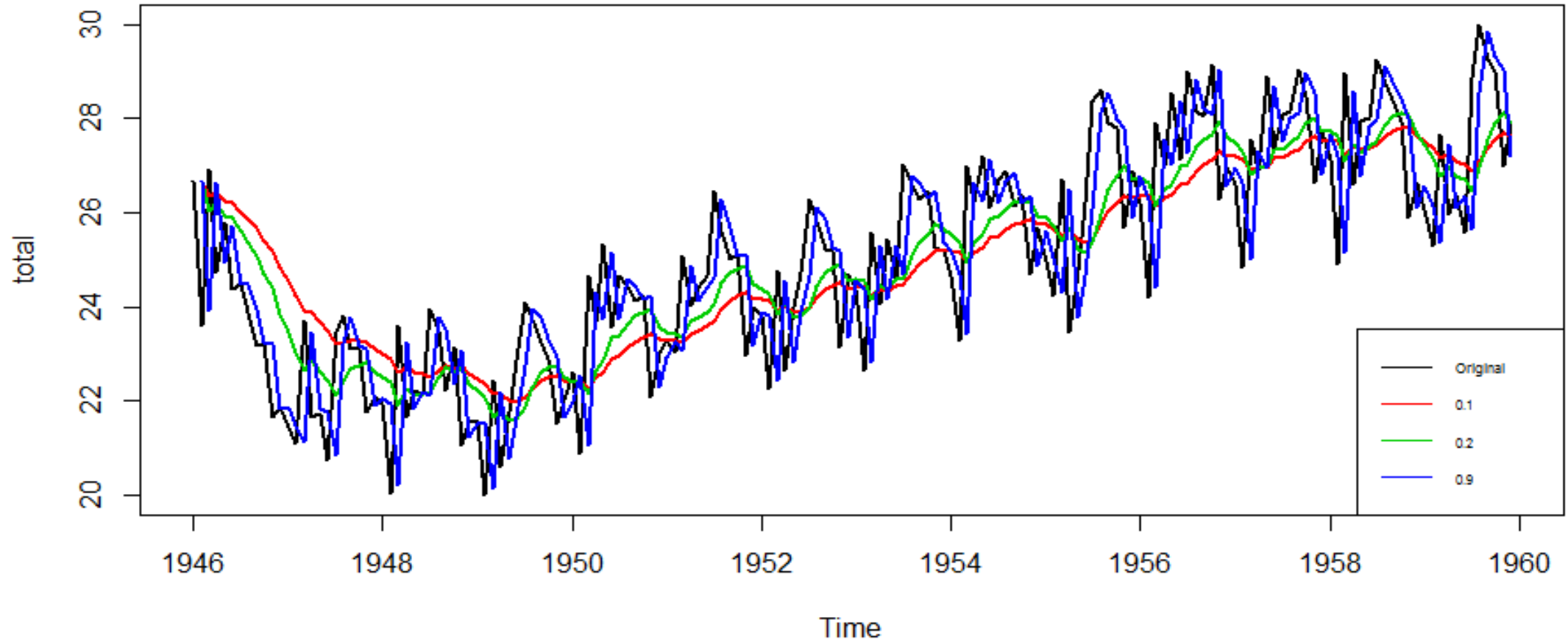


Single Exponential Smoothing: R Code

- `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","0.1","0.2","0.9"), col=1:ncol(total), lty = c(1, 1), cex=.5, y.intersp = 1)`



Choice of Alpha, α



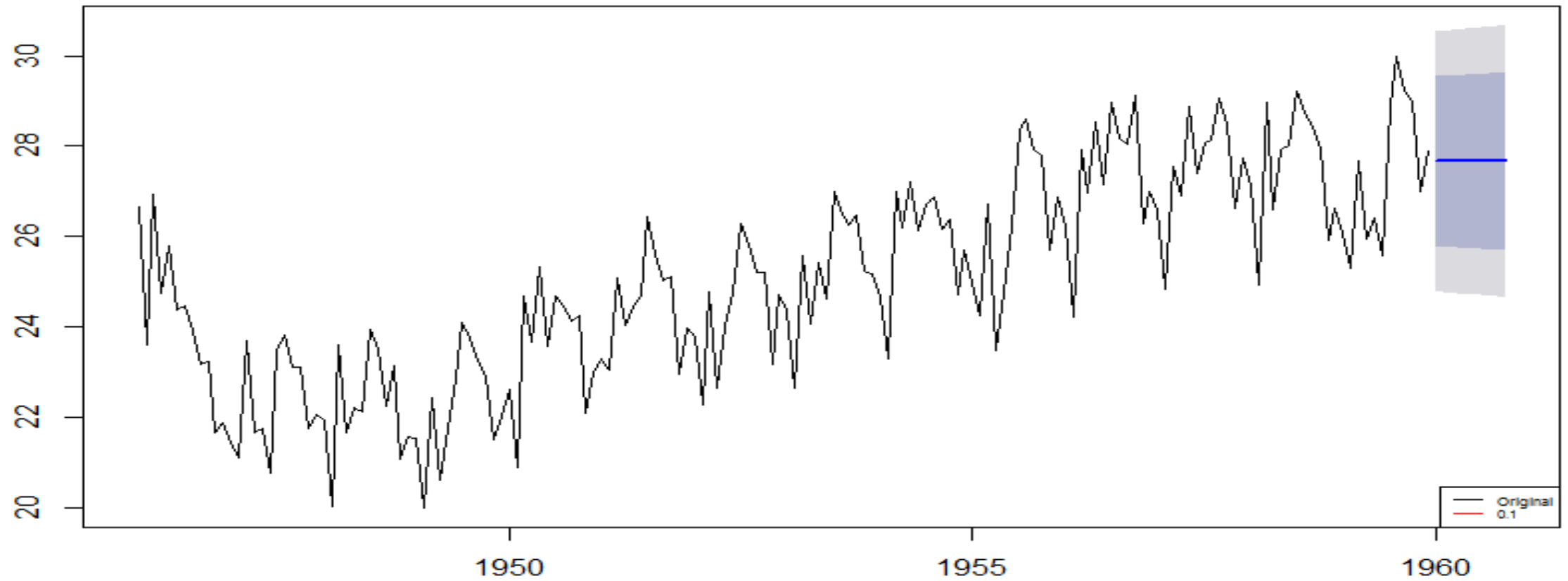
Single Exponential Smoothing Forecast: R Code

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



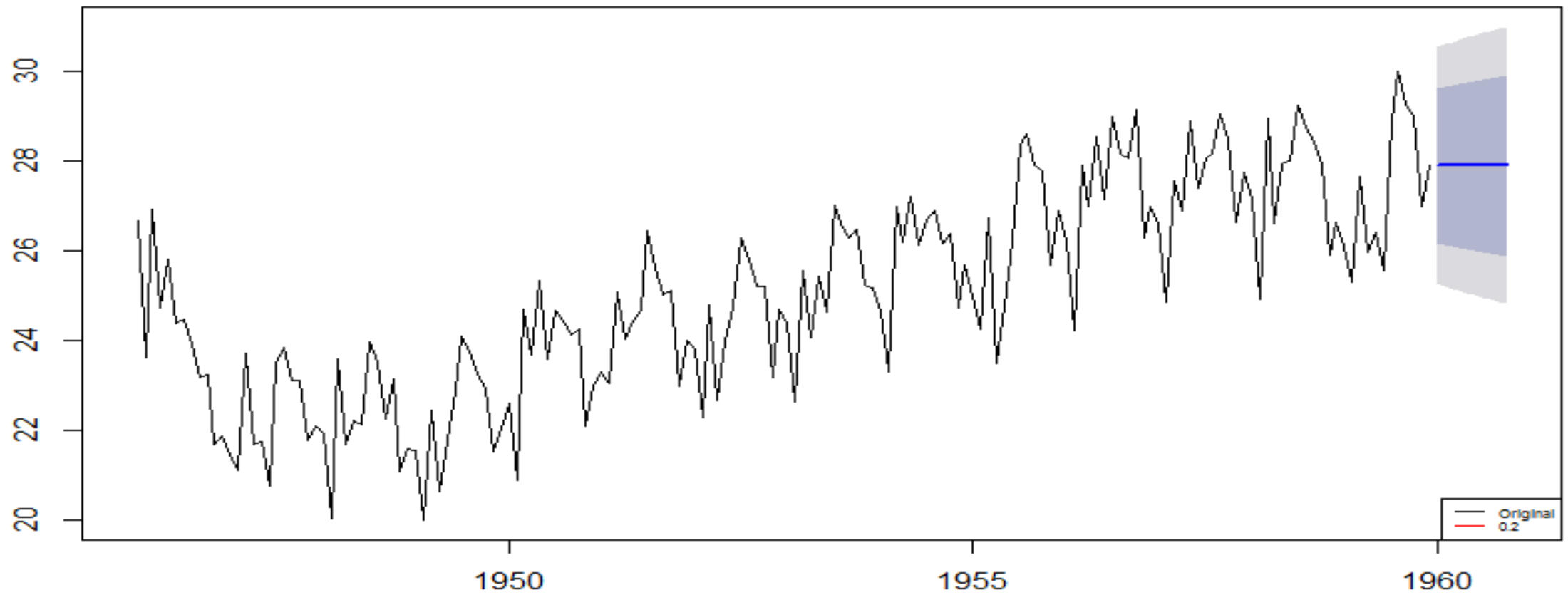
Forecast

Forecasts from HoltWinters



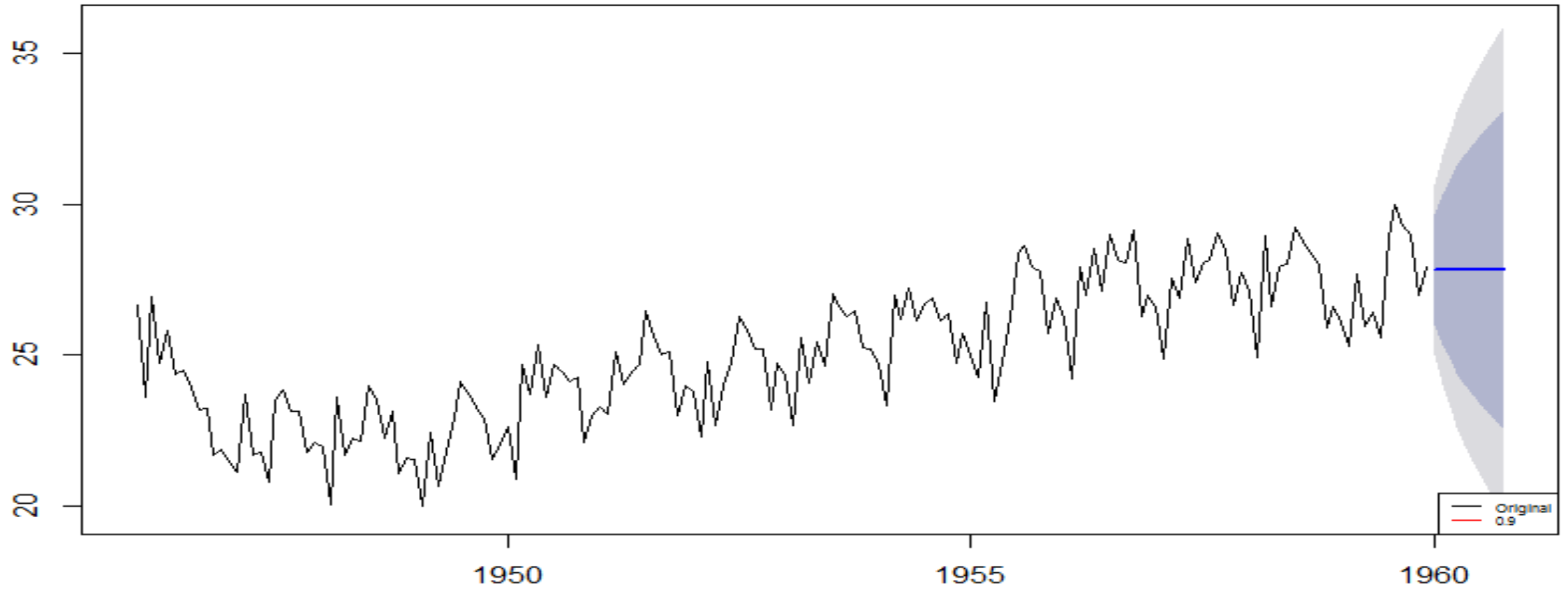
Forecast

Forecasts from HoltWinters



Forecast

Forecasts from HoltWinters



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)
```

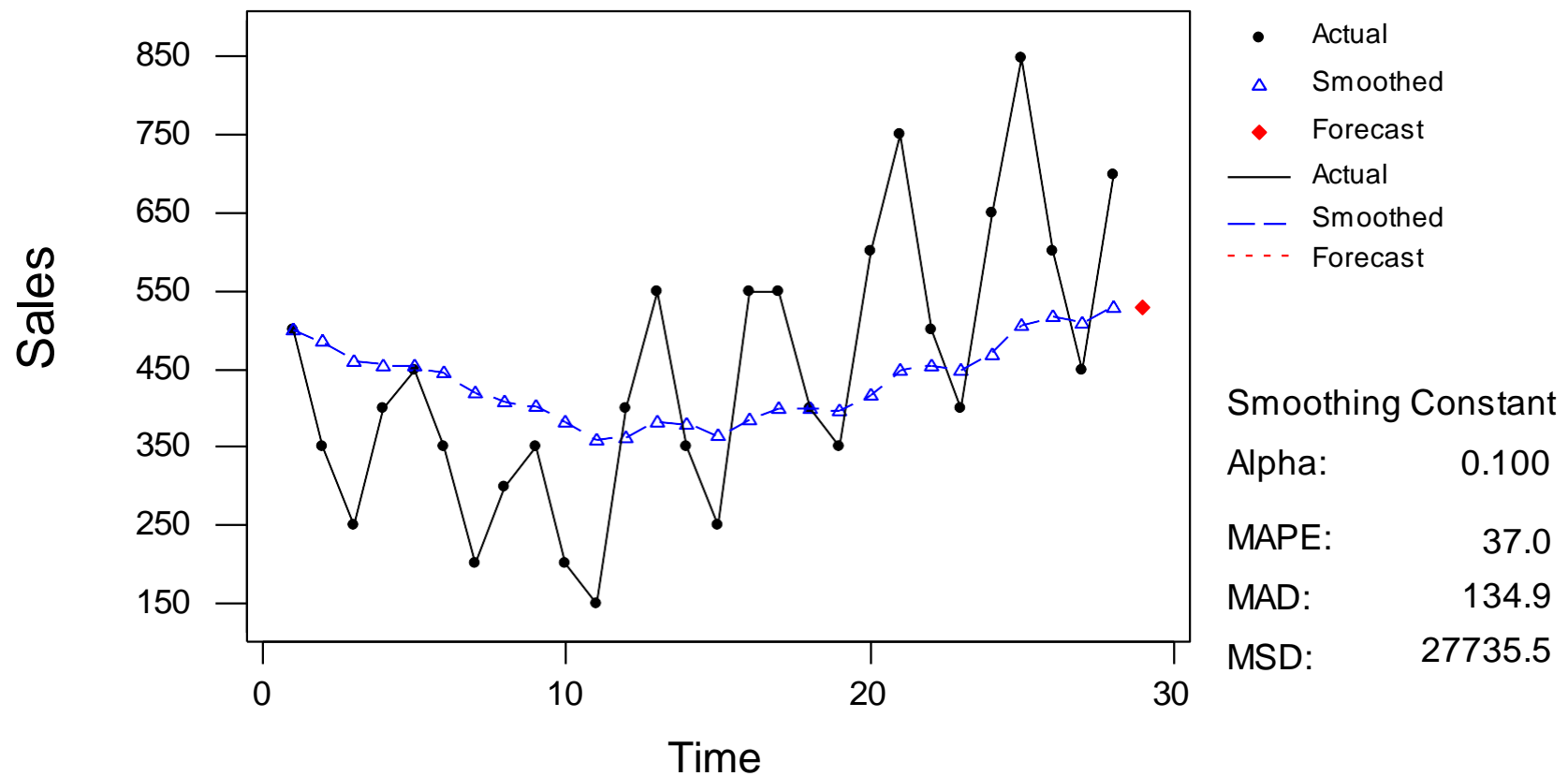
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	0.007744566	1.434287	1.124833	0.1983202	4.568417	1.192009	-0.4568299

Monitoring of Forecasts: Tracking

- Use a **tracking signal** (measure of errors over time) and setting limits.
- For example, if we forecast n periods, count the number of negative and positive errors.
 - If the number of positive errors is substantially less or greater than $n/2$, then the process is out of control
- Or use the running sum of forecast error

Sample Data

Sales data Single Exponential Smoothing $\alpha = 0.1$



Tracking Using a Confidence Interval

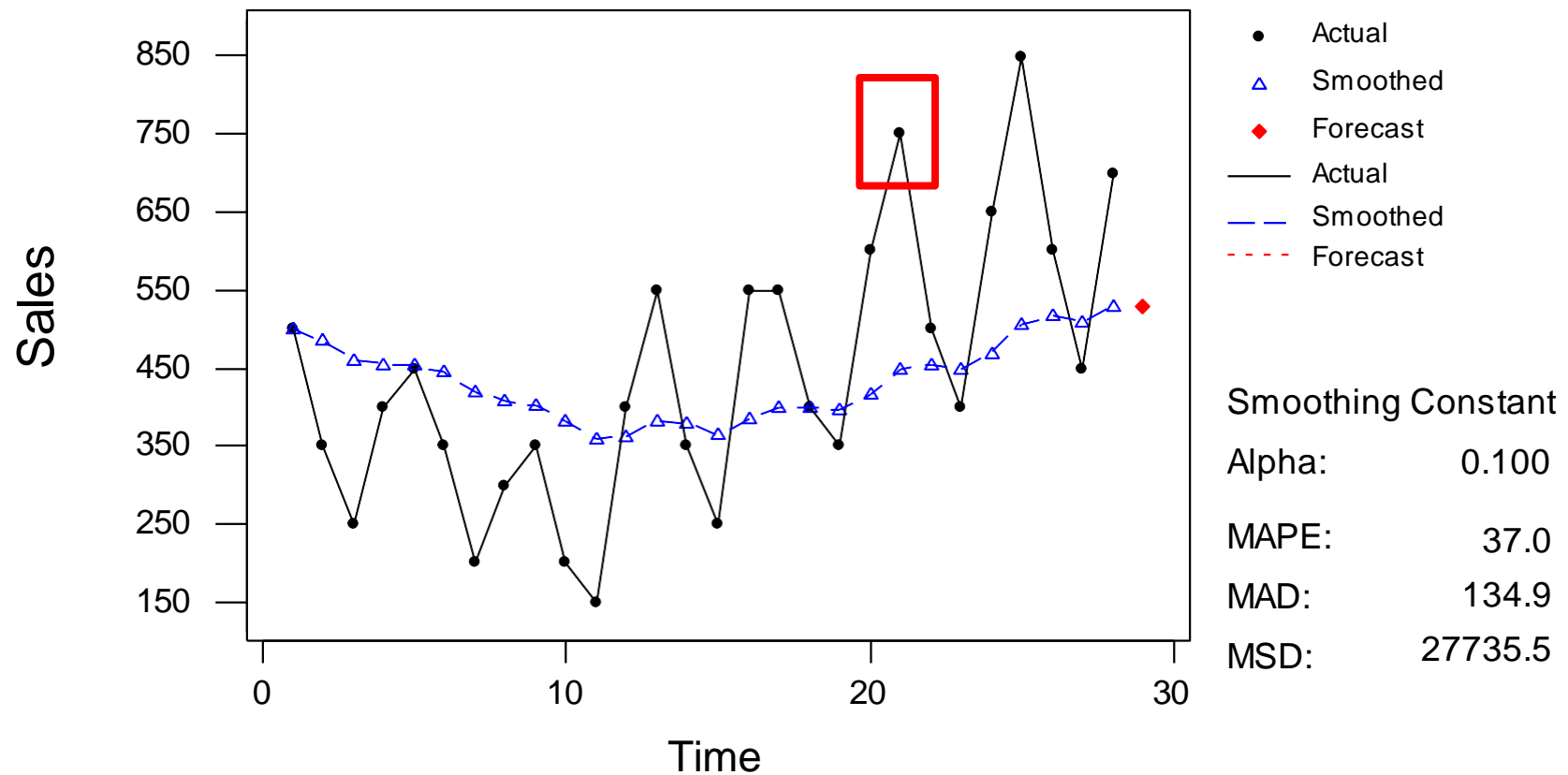
- Can also use 95% *prediction interval*

$$\hat{y} \pm z_{\frac{\alpha}{2}} \sqrt{MSE}$$

- If the forecast error is **outside of the interval**, use a new optimal α .
- Looking back at the 0.1 single exponential smoothing:
 - Observation #21 is **out-of-control**. We need to re-evaluate alpha level because this technique is bi $\hat{y} \pm 1.96 * \sqrt{27735.5} = \hat{y} \pm 326.4$

Sample Data

Sales data Single Exponential Smoothing $\alpha = 0.1$



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:

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

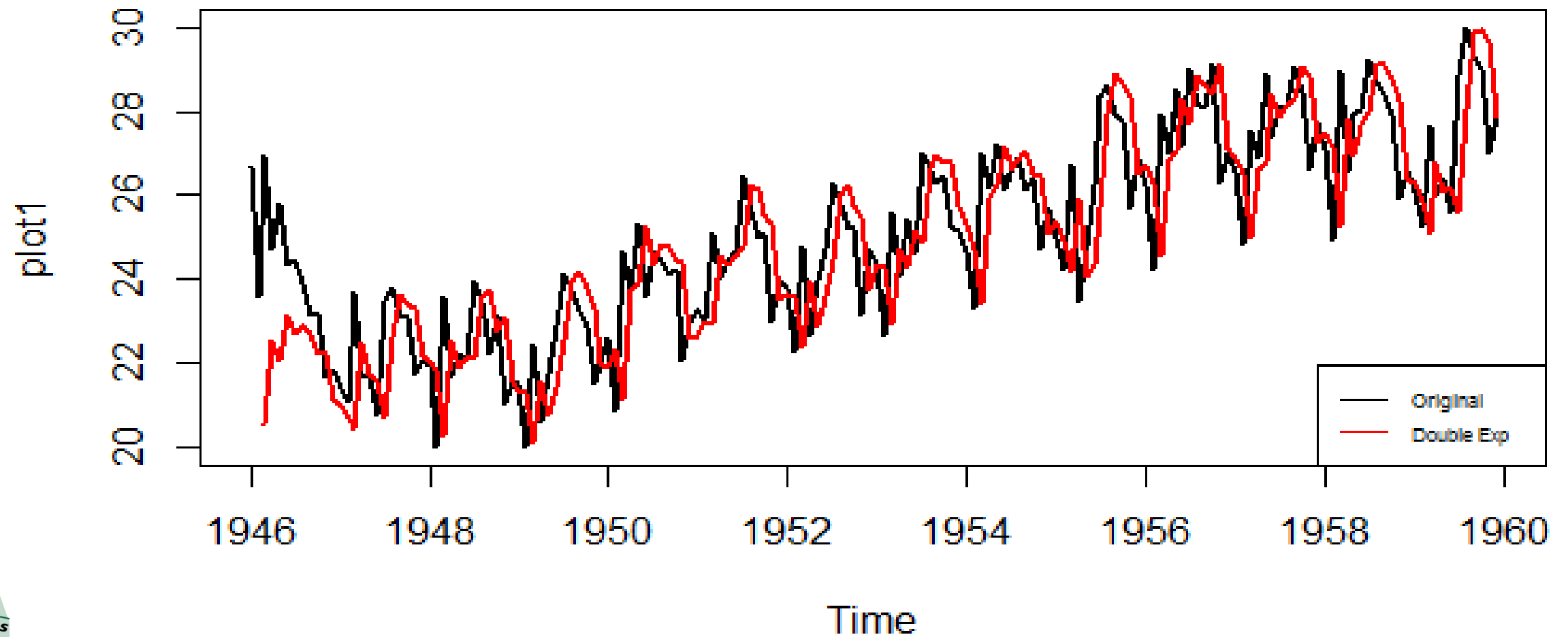
- Where:

- 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

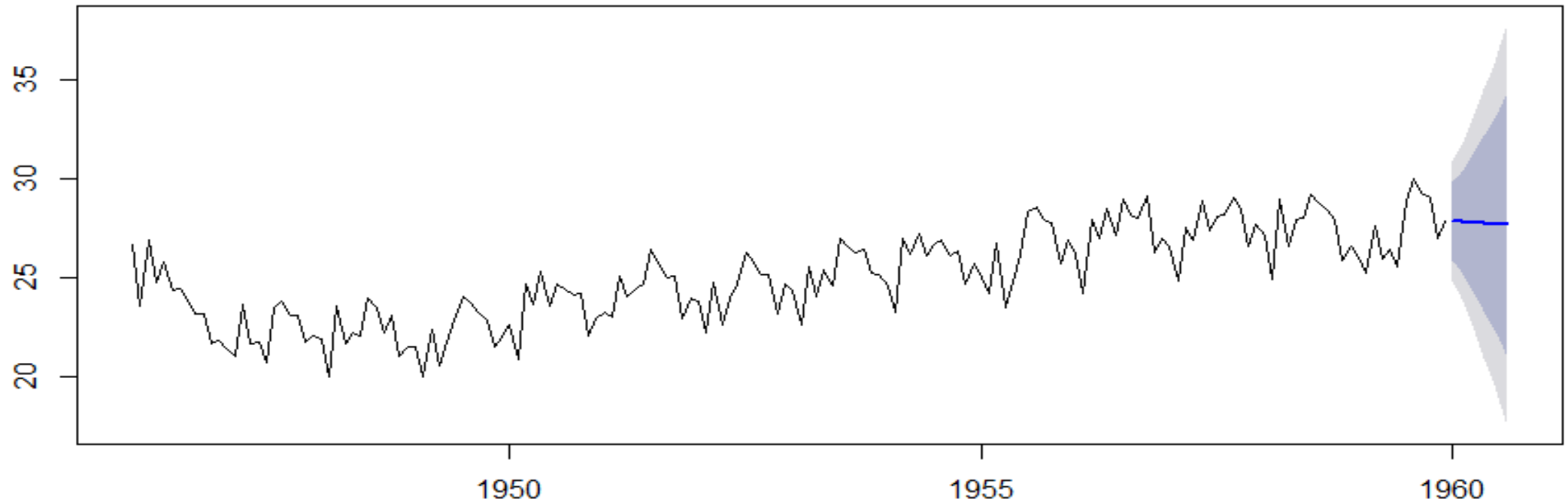


R Code: Forecast - Holt's Method

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

Forecast: Holt's Method

Forecasts from HoltWinters



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:

$$- S_t = \frac{\gamma y_t}{\hat{y}_t} + (1 - \gamma)S_{t-s}$$

- **Forecast** m periods into the future:

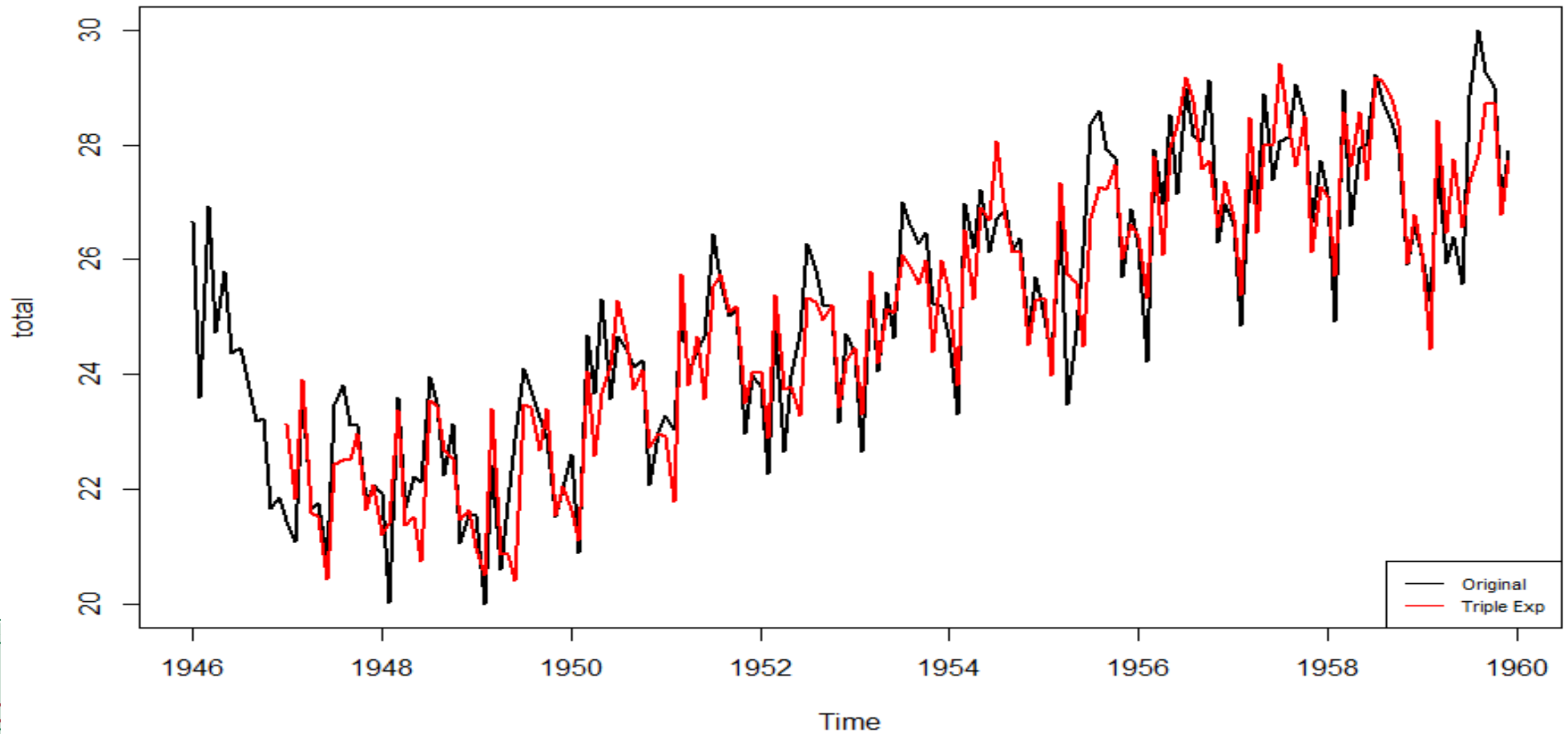
$$- \hat{y}_{t+m} = (\hat{y}_t + mT_t) S_{t-s+m}$$

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

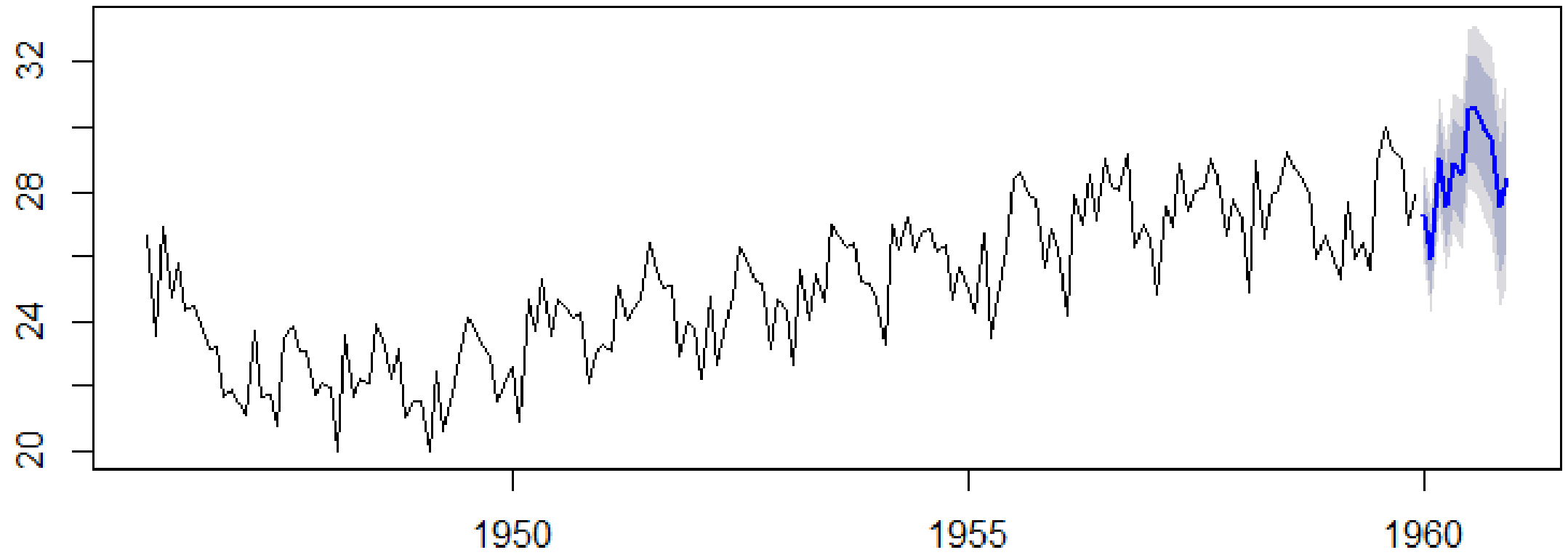


Forecast: Winter's Method

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

Forecast: Winter's Method

Forecasts from HoltWinters



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: $y_t = T_t \times S_t \times I_t \times Ct$
- 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

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

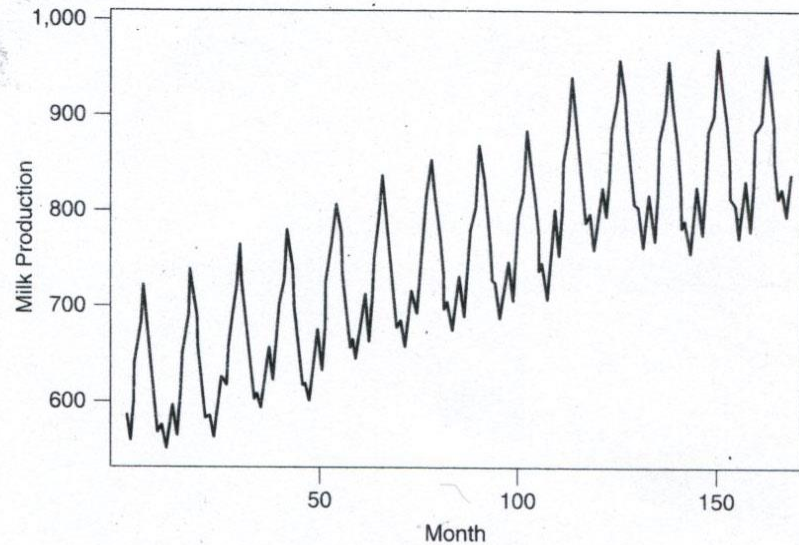
- The multiplicative model works best when the variability of the time series **increased with the level**.
 - Variance of the series varies as the trend progresses
 - More prevalent with economic series since most seasonal variation increases with the level of the series



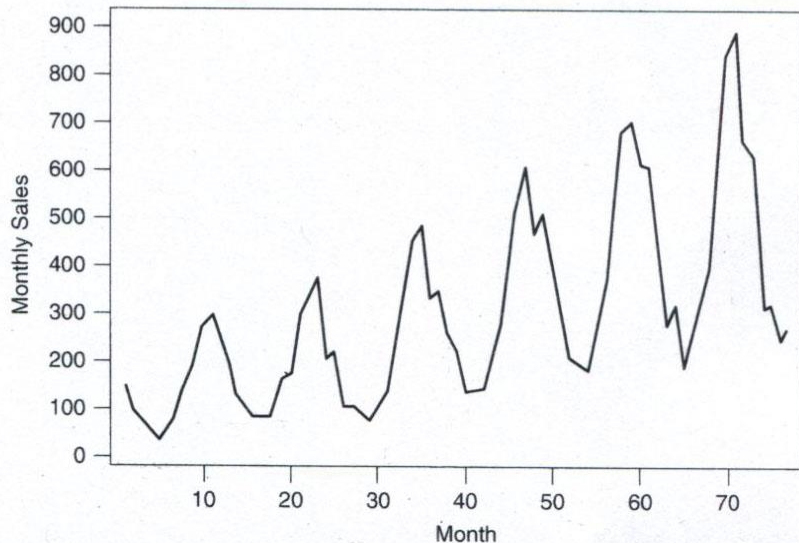
Note on Cycles

- 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

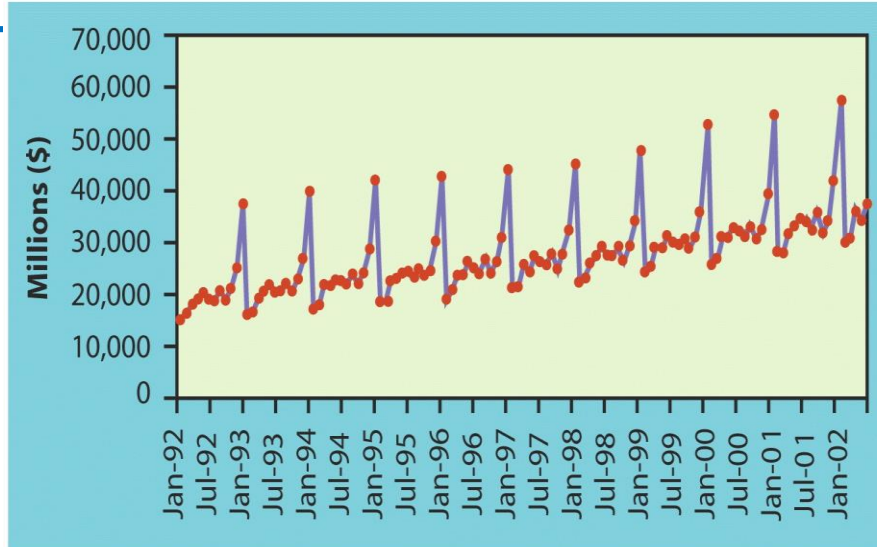


(a) A time series with constant variability

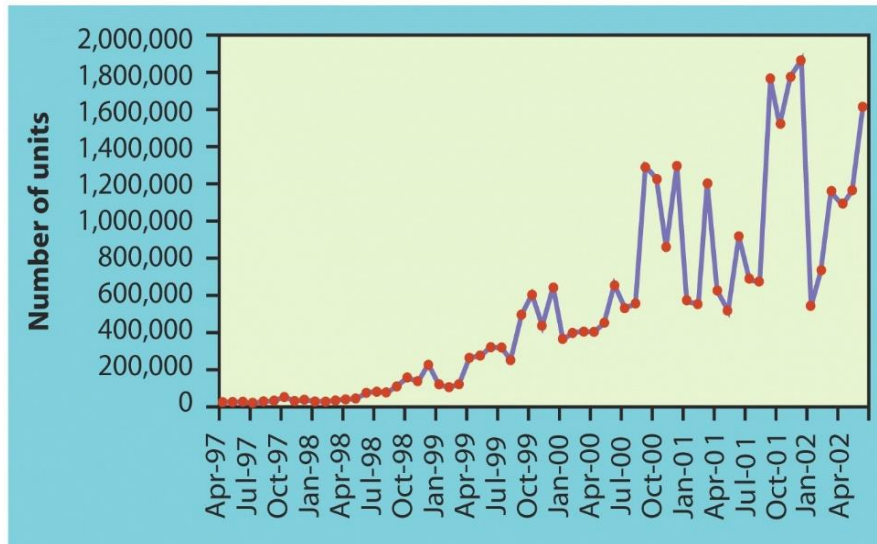


(b) A time series with variability increasing with level

Additive and Multiplicative Models

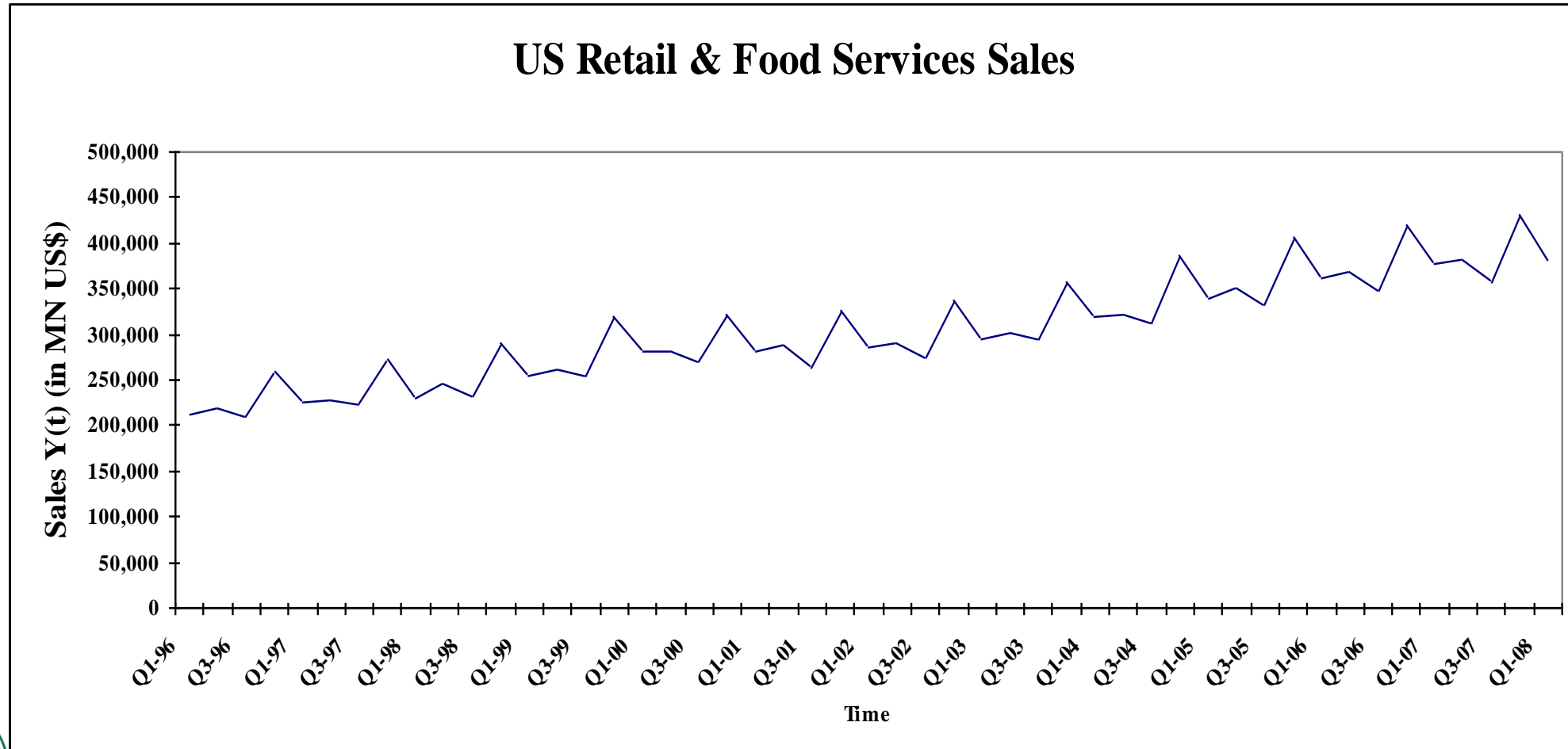


(a) A time series with constant variability

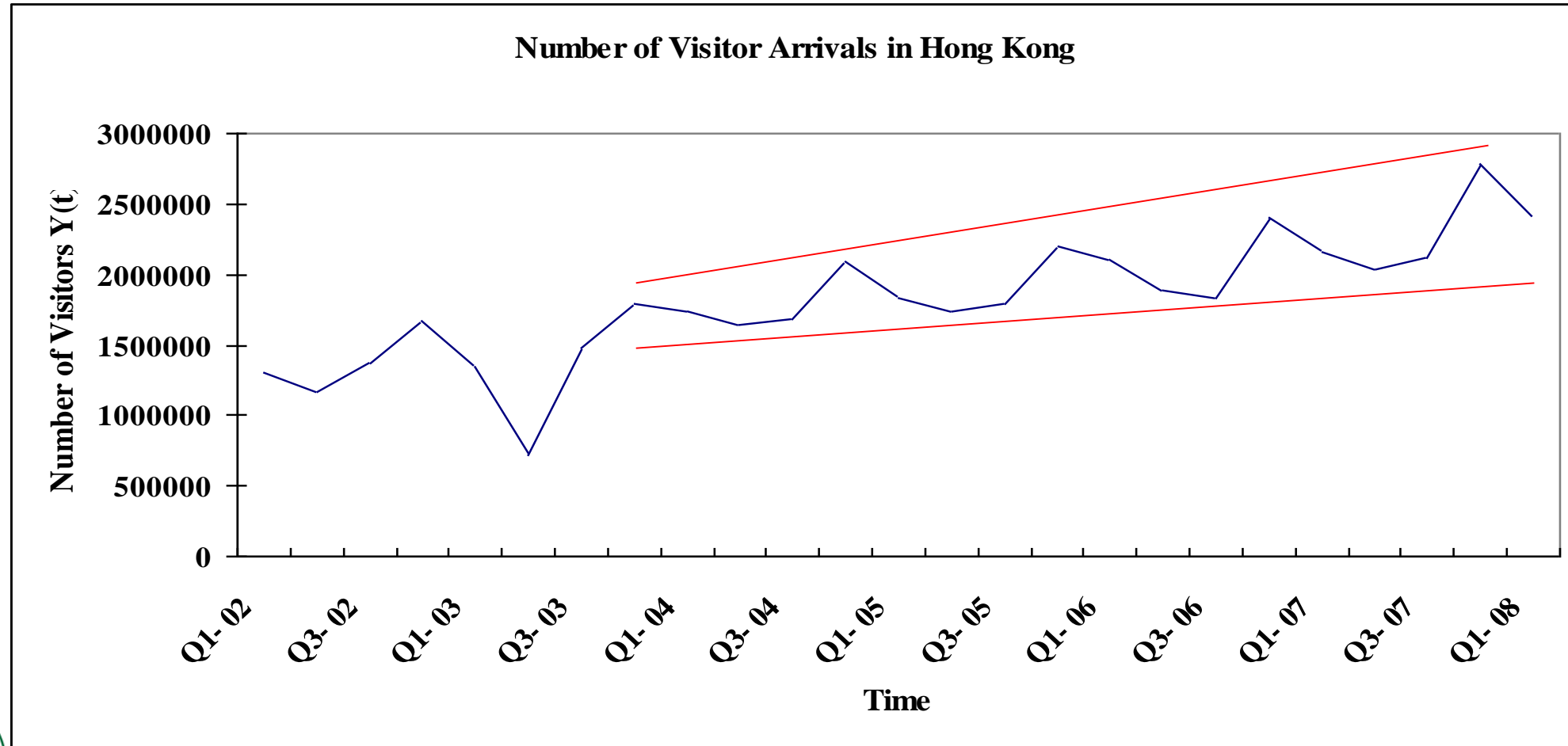


(b) A time series with variability increasing with level

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



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 Indices

- The seasonal indices measure the **seasonal variation in the series**. Seasonal indices **are percentages** that show changes over time.
 - With monthly data, a seasonal index of 1.0 for a particular month means the expected value for that month is equal to the average of the year
 - An index of 1.25 for a different month implies the observation for that month is expected to be 25% more than the average of the year
 - A monthly index of 0.80 indicates that the expected level of that month is 20% less than the average of the year

Seasonal Indices

- The centered moving average is another method to determine seasonal indices
- The centered moving averages represent the de-seasonalized data
- The degree of seasonality, called seasonal factor (SF), is the ratio of the actual value to the de-seasonalized value

$$SF_t = \frac{Y_t}{CMA_t}$$

Seasonal Indices: Example

Time	QRT	Y	CMA(4)	CMA(2)	SRI
1	1	10			
2	2	18			
3	3	20	15	15.25	1.31
4	4	12	15.5	15.75	0.76
5	1	12	16	16.5	0.73
6	2	20	17	17.125	1.17
7	3	24	17.25	17.5	1.37
8	4	13	17.75	18	0.72
9	1	14	18.25	18.75	0.75
10	2	22	19.25	19.625	1.12
11	3	28	20		
12	4	16			

Seasonal Indices: Example

- For Quarter 3

$$SF_3 = \frac{1.31 + 1.37}{2} = 1.34$$

- For Quarter 4

$$SF_4 = \frac{0.76 + 0.72}{2} = 0.74$$

- For Quarter 1

$$SF_1 = 0.73$$

- For Quarter 2

$$SF_2 = 1.14$$



Seasonal Adjustment

- After the seasonal component has been isolated, it can be used to calculate **seasonally adjusted data**.
- Seasonal adjustment techniques are ad hoc methods of computing **seasonal indices**
- Use these indices to de-seasonalize the series by removing those seasonal variation
- Additive Model: **de-seasonalized data = raw data - seasonal index**
- Multiplicative Model: **de-seasonalized data = raw data / seasonal index**



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.

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

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.



Computing Seasonal Indices for Each Quarter

- Example Seasonal Indices for an Additive Model
 - Quarter 1: -8.645
 - Quarter 2: 1.783
 - Quarter 3: -1.785
 - Quarter 4: 8.645



Identification of Seasonal Components

- If an additive decomposition is employed, estimates of the trend, seasonal components are **added together** to produce the original series.
- If an multiplicative decomposition is employed, estimates of individual components must be **multiplied together** to produce the original series

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:

$$T_t = \beta_0 \beta_1^t$$

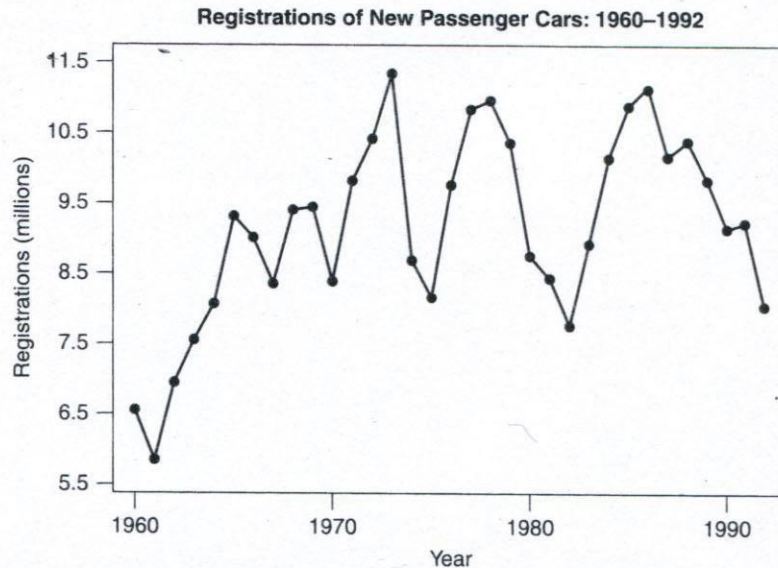
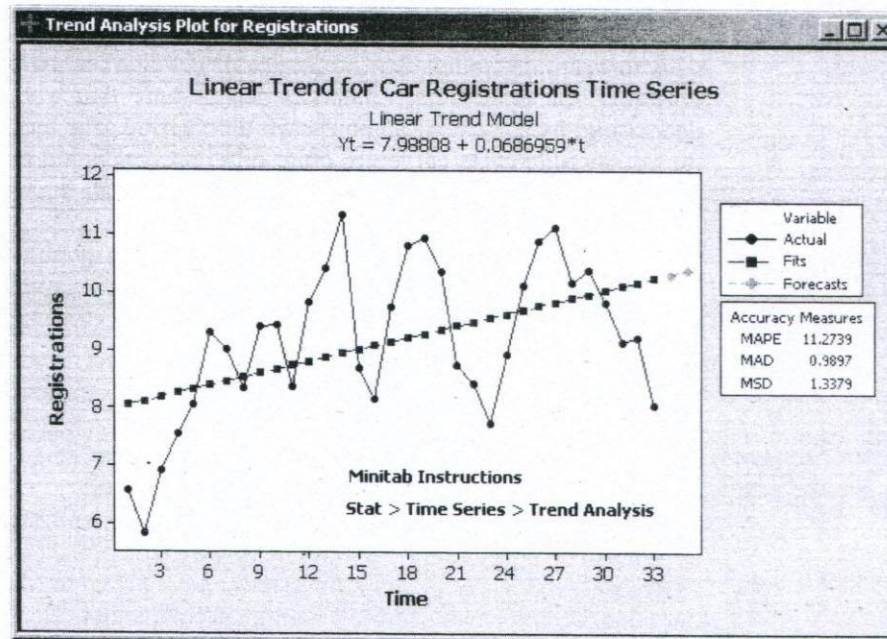
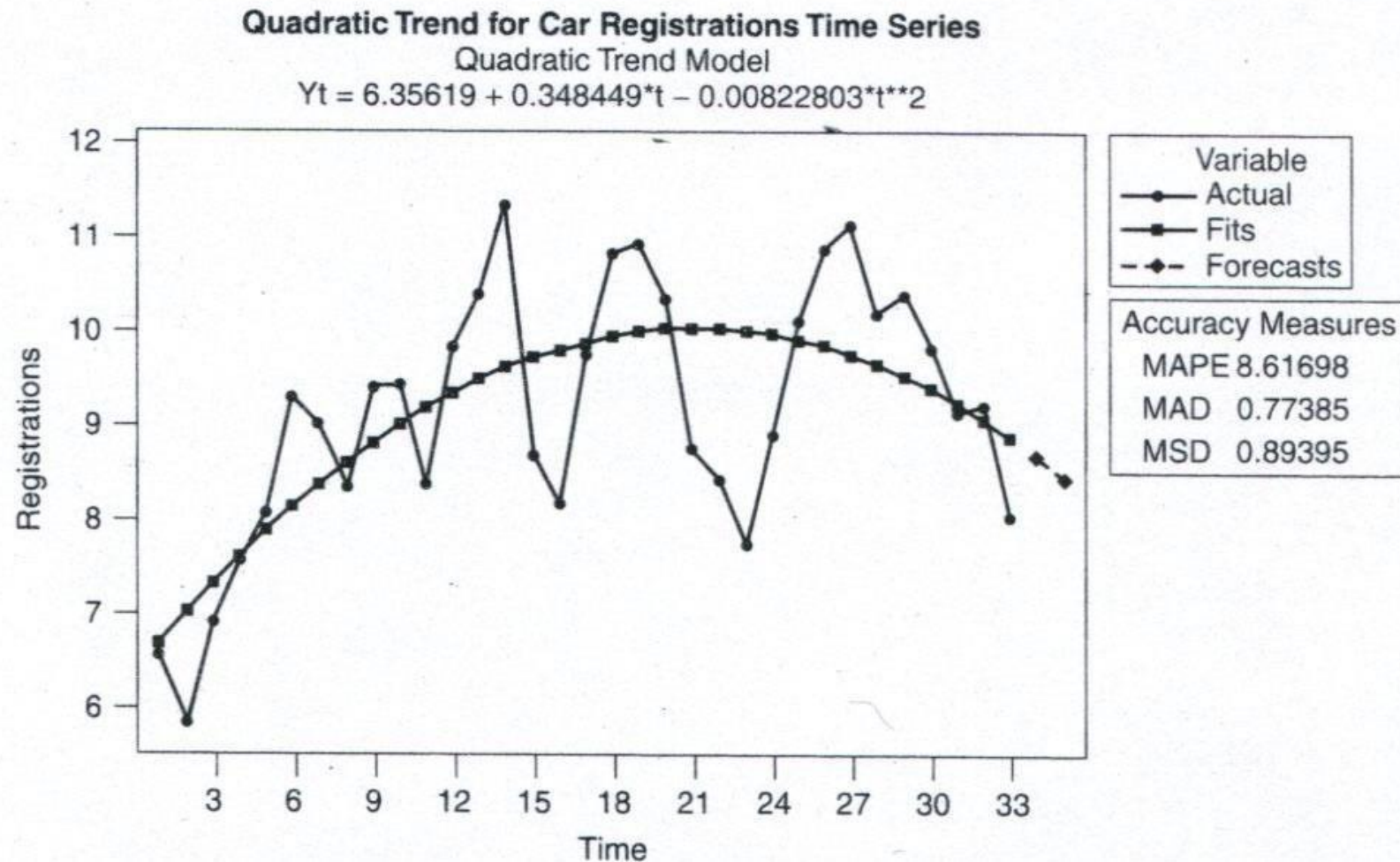


FIGURE 5-2 Car Registrations Time Series for Example 5.1

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

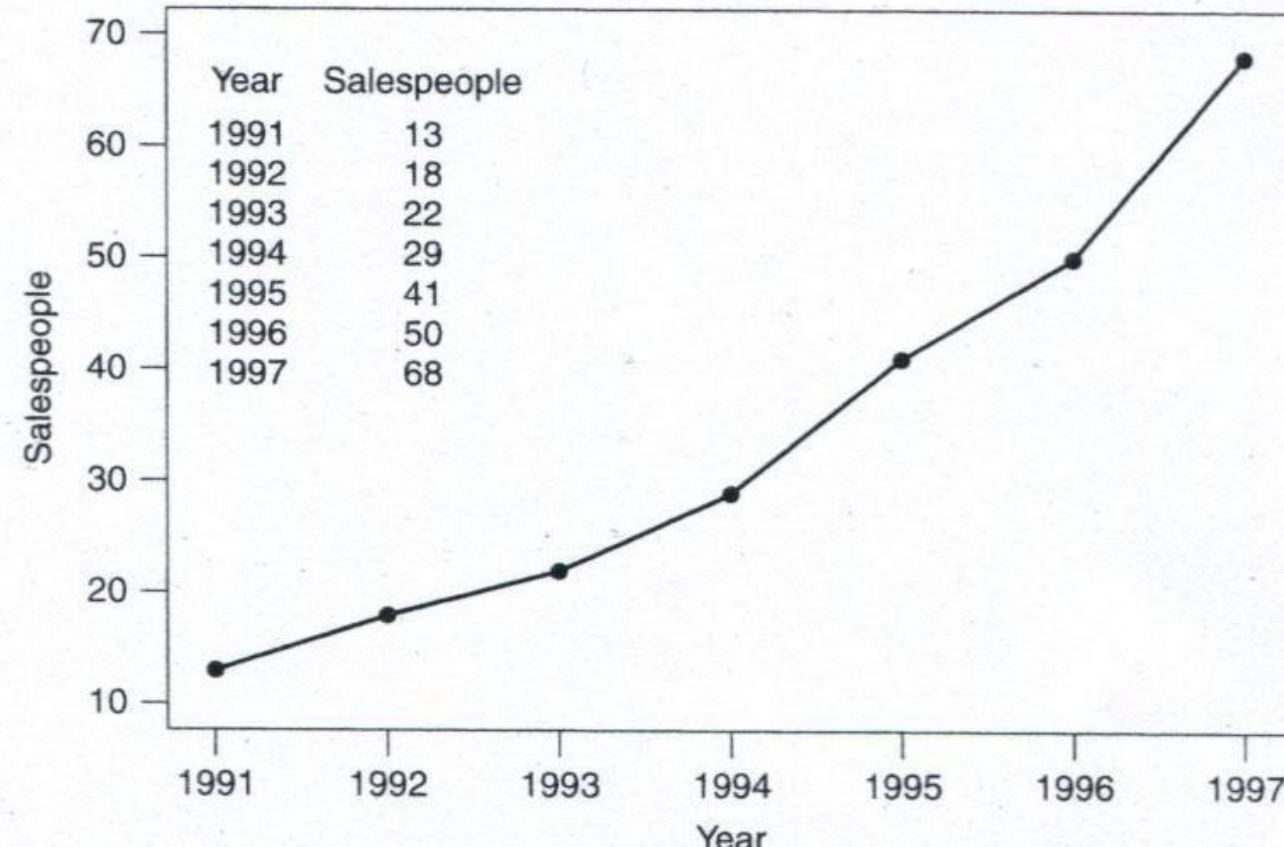


Trend Equations Using Quadratic Regression: Trend Line for the Car Registrations Time Series

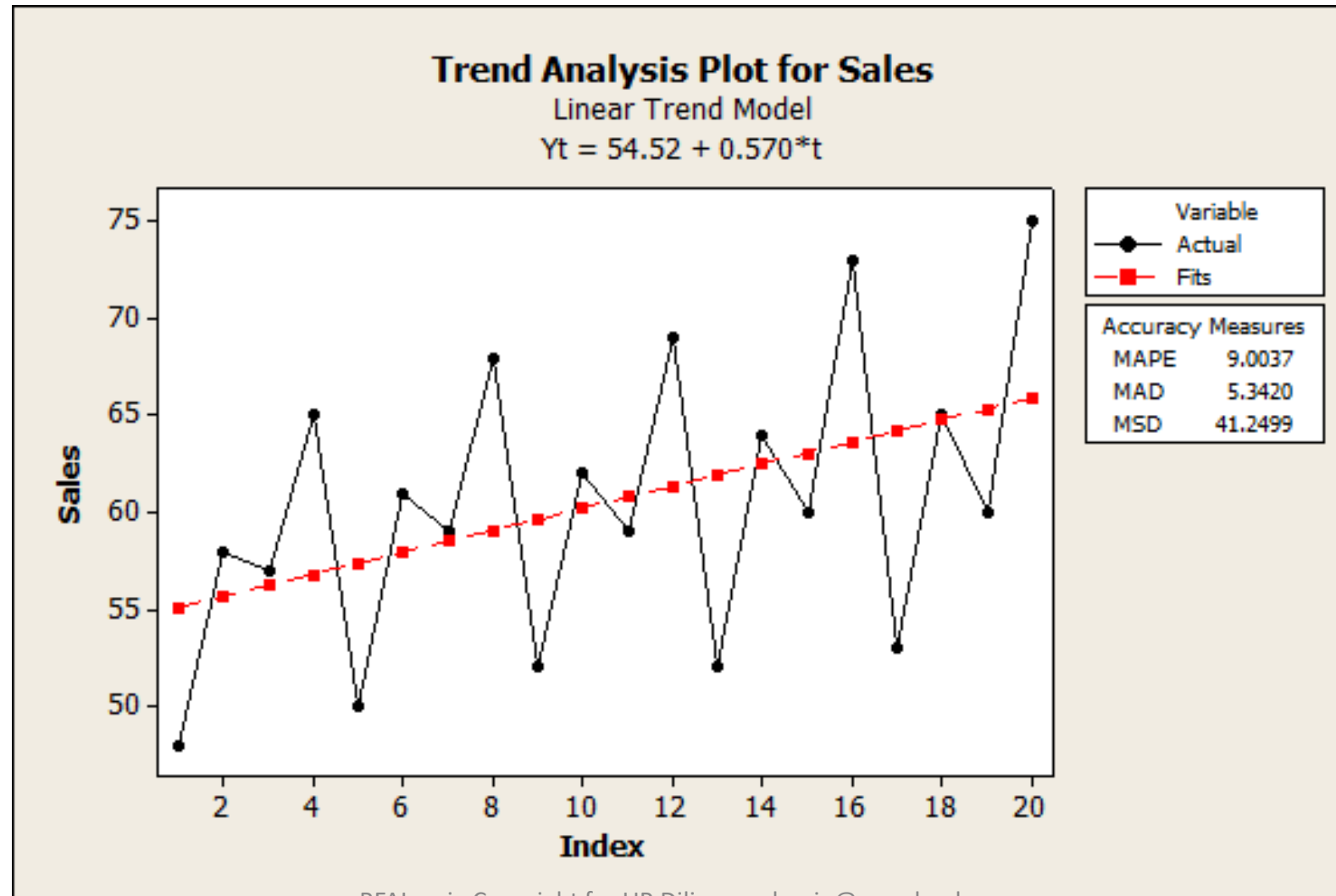


Exponential Trend

- 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:
 - $T'_t = 10.016(1.313)^t$



Trend Analysis: Minitab Output



De-trended Sales Data (Additive Model)

- $y_t - TC_t = S_t + I_t$

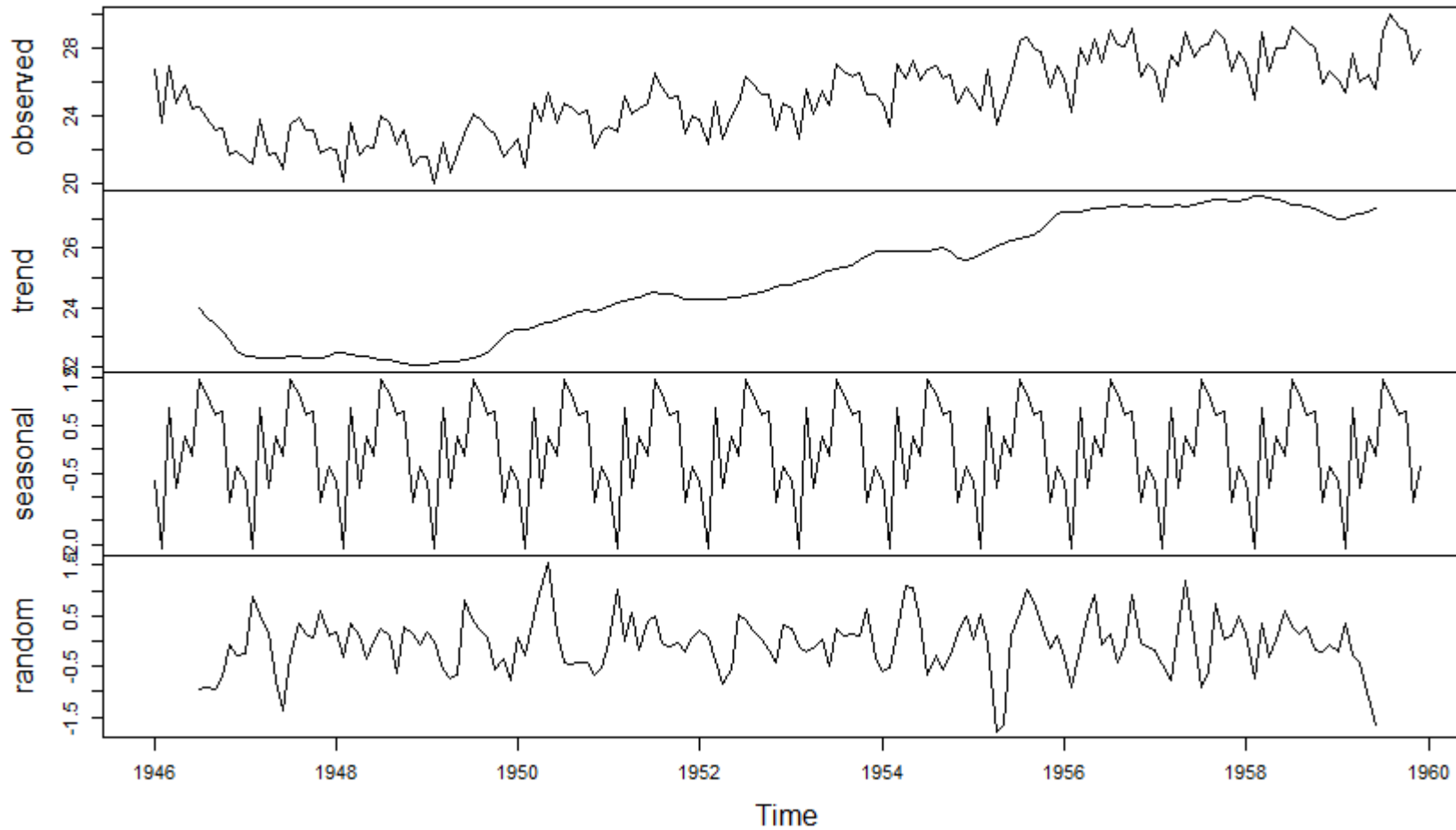


R Code: Decomposition 1

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

R Code: Decomposition 1

Decomposition of additive time series



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`

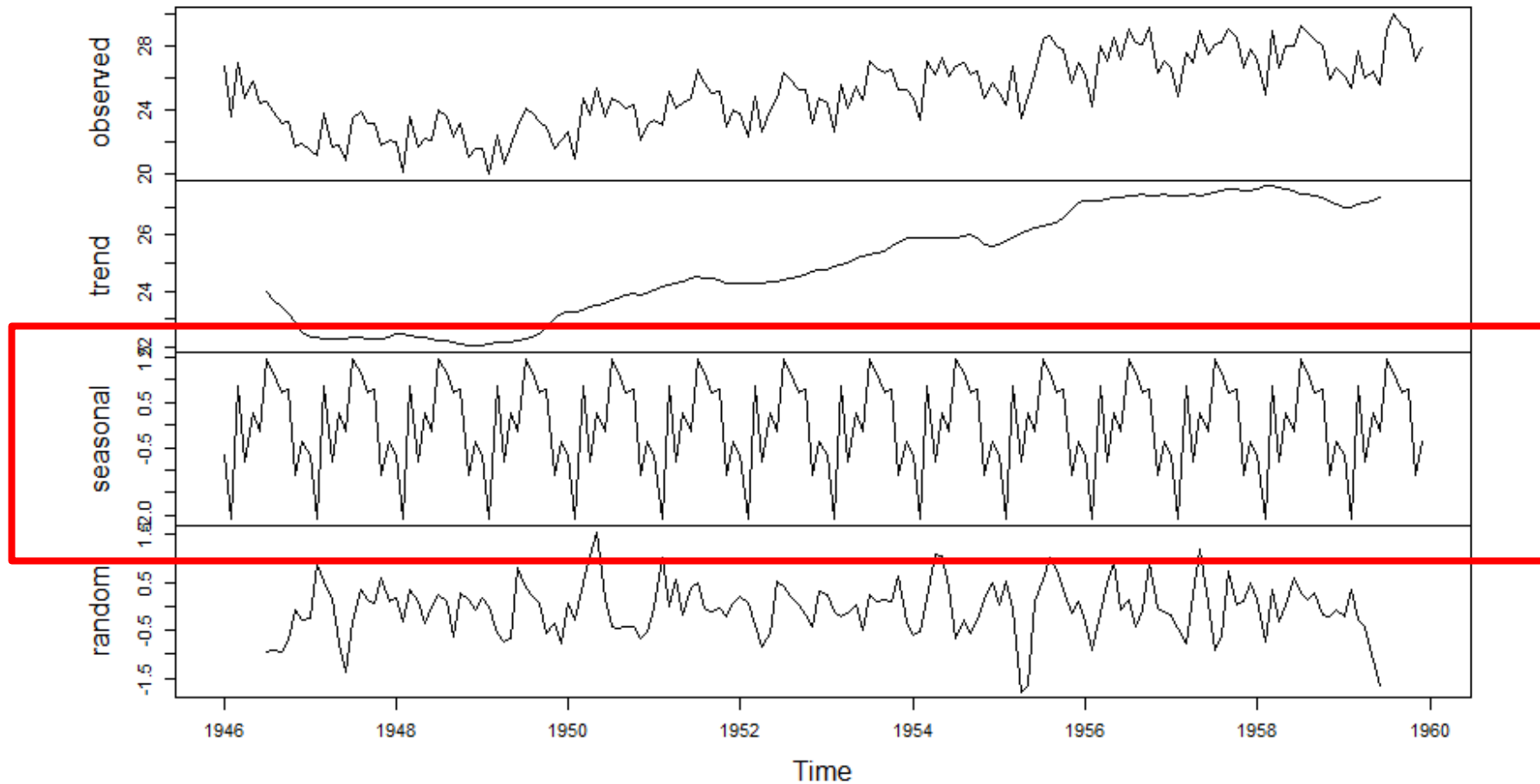
Notes on Seasonality

- 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

Decomposition of additive time series



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

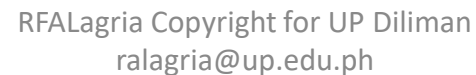
Notes on Seasonality

- 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

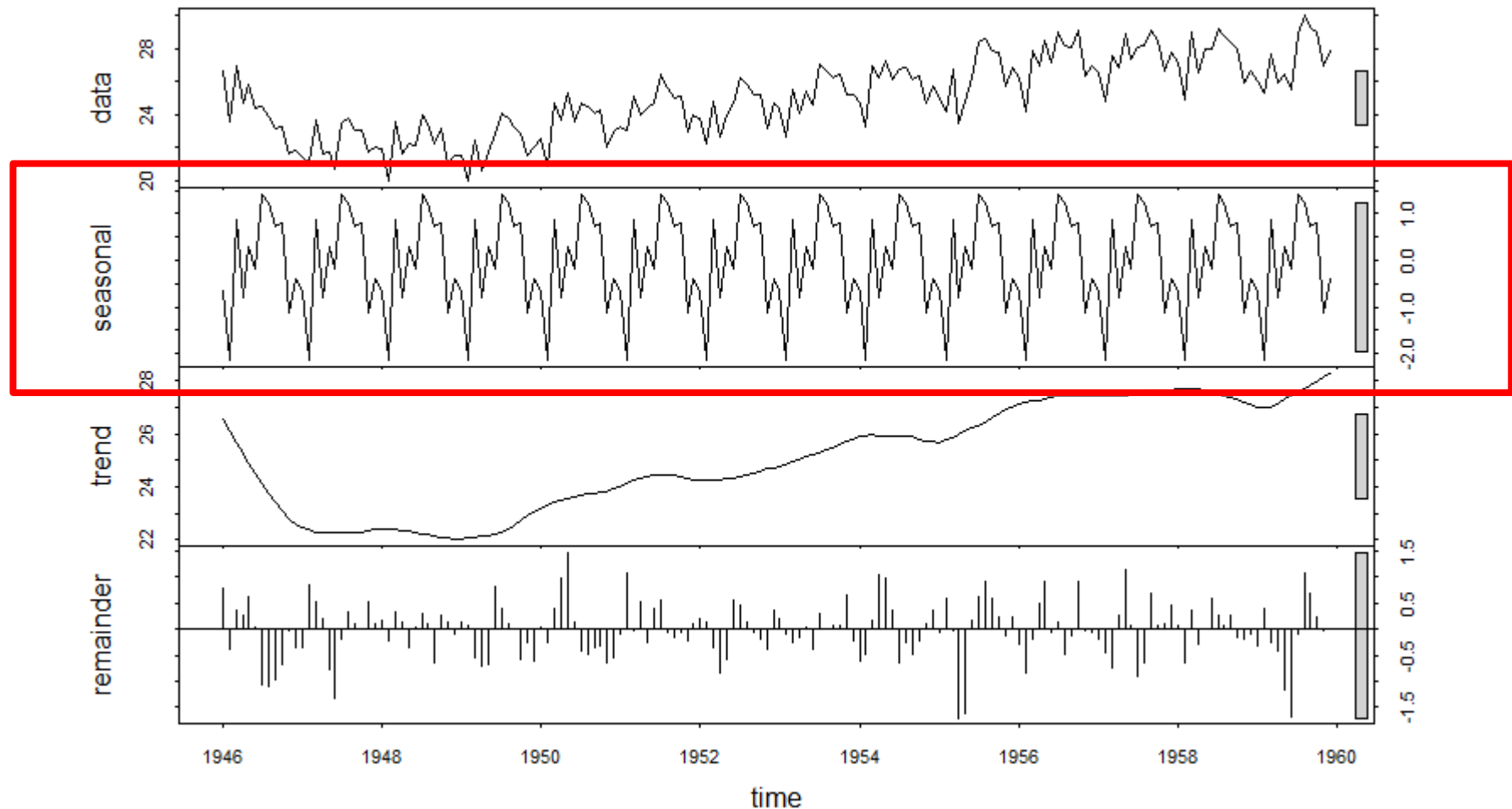


- # Seasonal Relatives



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

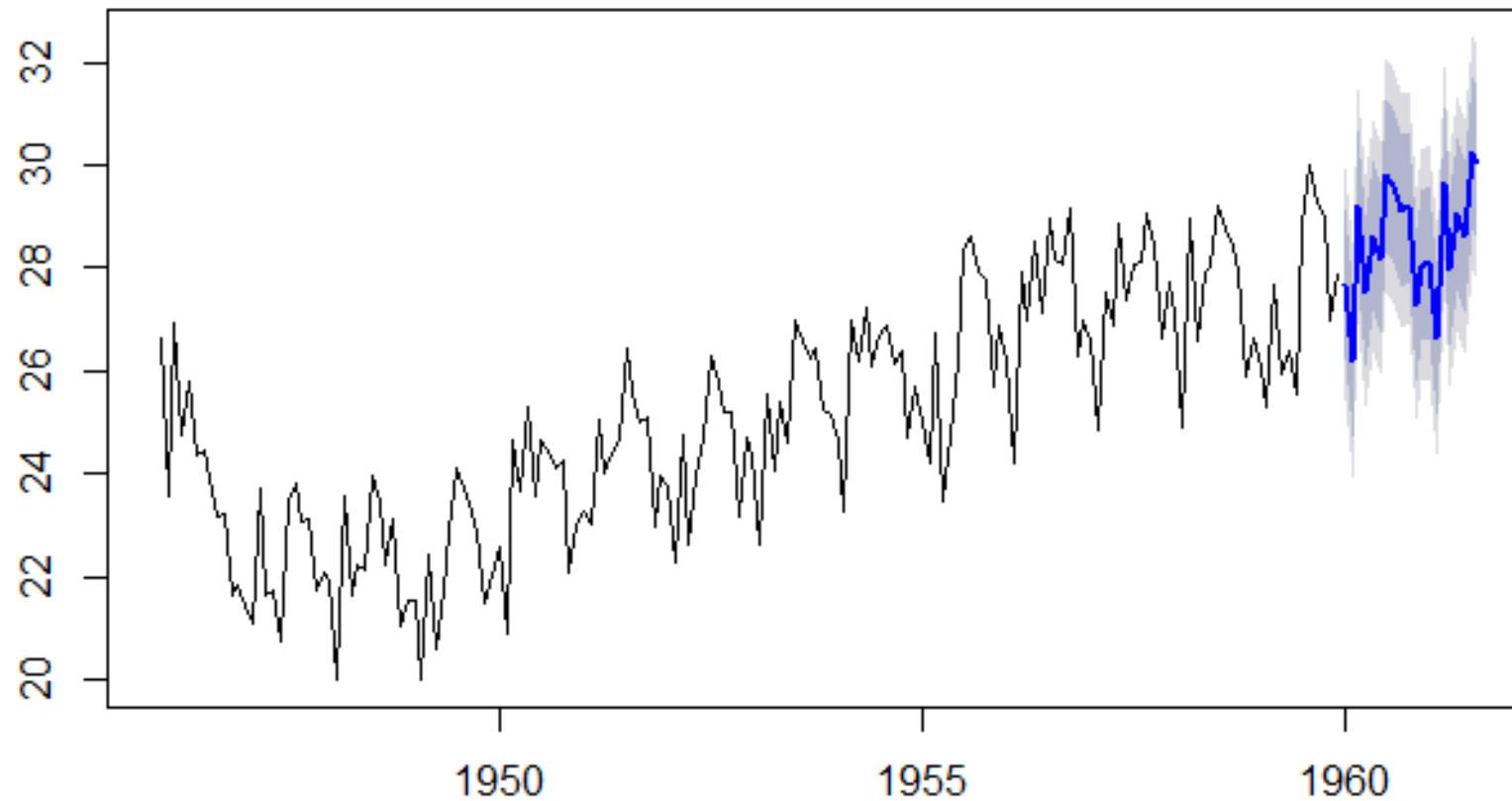


R Code: Decomposition 2

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

Trend Analysis using Regression Model

Forecasts from Linear regression model



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

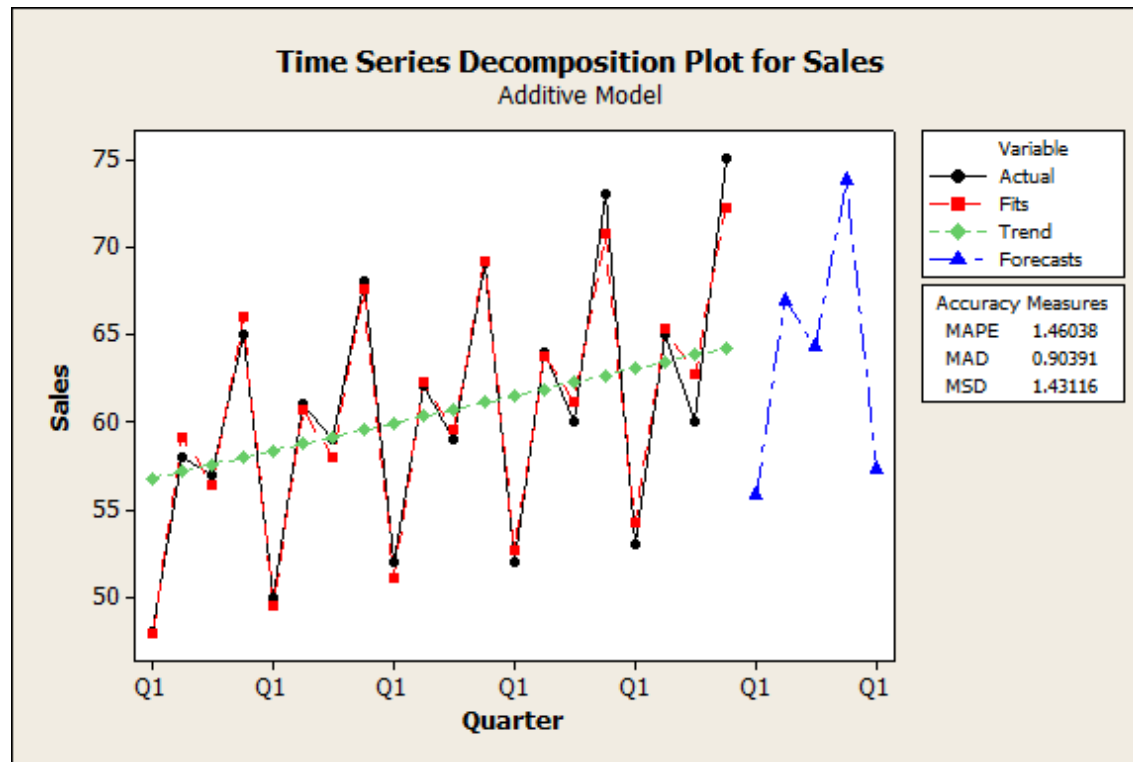
Multiple R-squared: 0.7929, Adjusted R-squared: 0.7768

F-statistic: 49.44 on 12 and 155 DF, p-value: < 2.2e-16



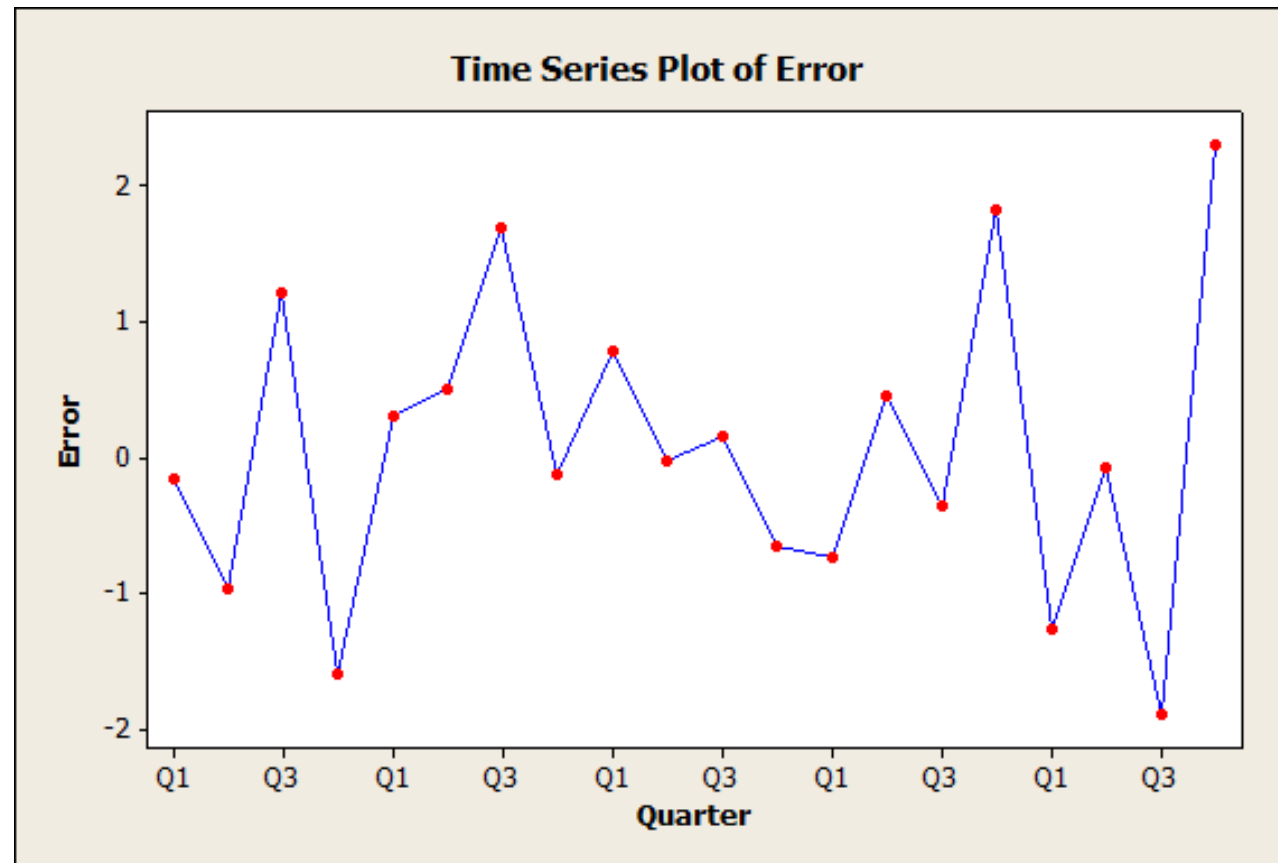
Forecast Predictions With Final Additive Model

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



Error Component Computation (Additive Model)

- $y_t - T_t - S_t = I_t$



How to Choose?

Forecasting Method	Amount of Historical Data	Data Pattern	Forecast Horizon
Linear Regression	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 1 & 2

- Given M&L.csv data set.
- Data provided is the weekly demand for two products from order records of the previous 14 weeks.

Case Study 1 & 2

- Given Chocolates.csv and Airlines.csv dataset.
- Data provided is the quarterly production of chocolates (in metric tons) in Australia from 1957 to 1994.

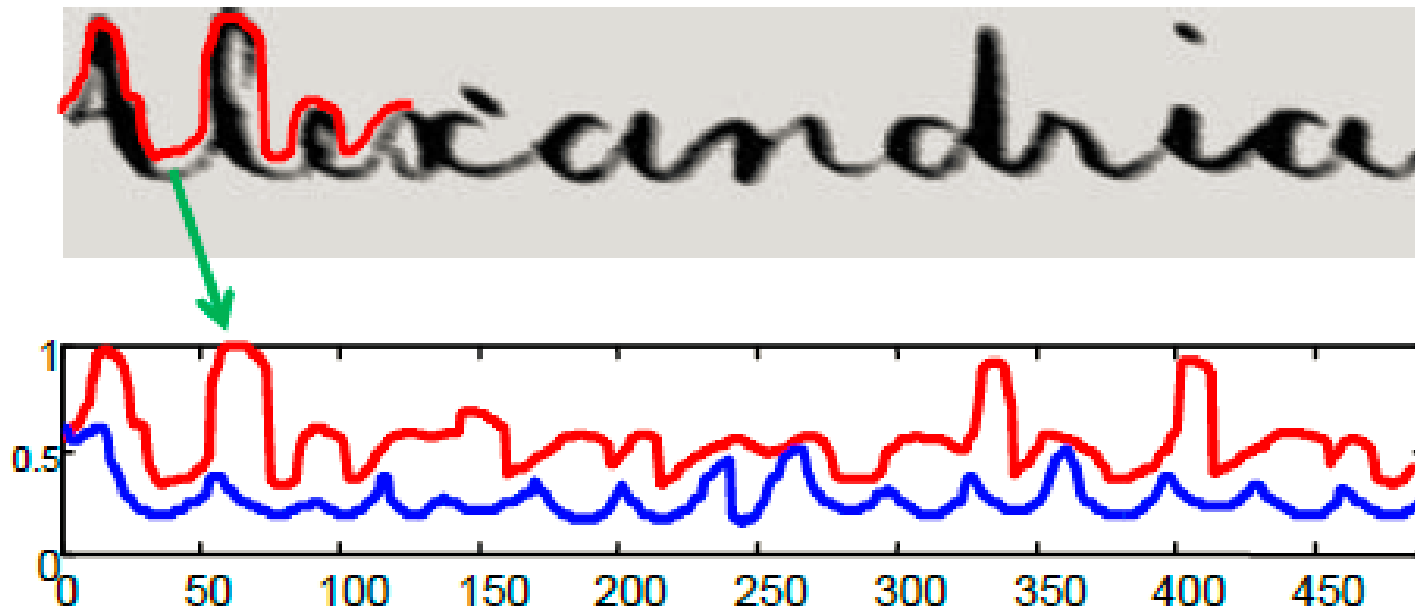
Outline for This Training

1. Introduction to Forecasting in Business Intelligence
2. Demand Forecasting Techniques
 - Qualitative
 - Quantitative
3. Accuracy of Forecasts
4. Monitoring of Forecasts
5. Forecasting with R
- 6. Introduction to Time Series Data Mining**
7. Advanced Time Series



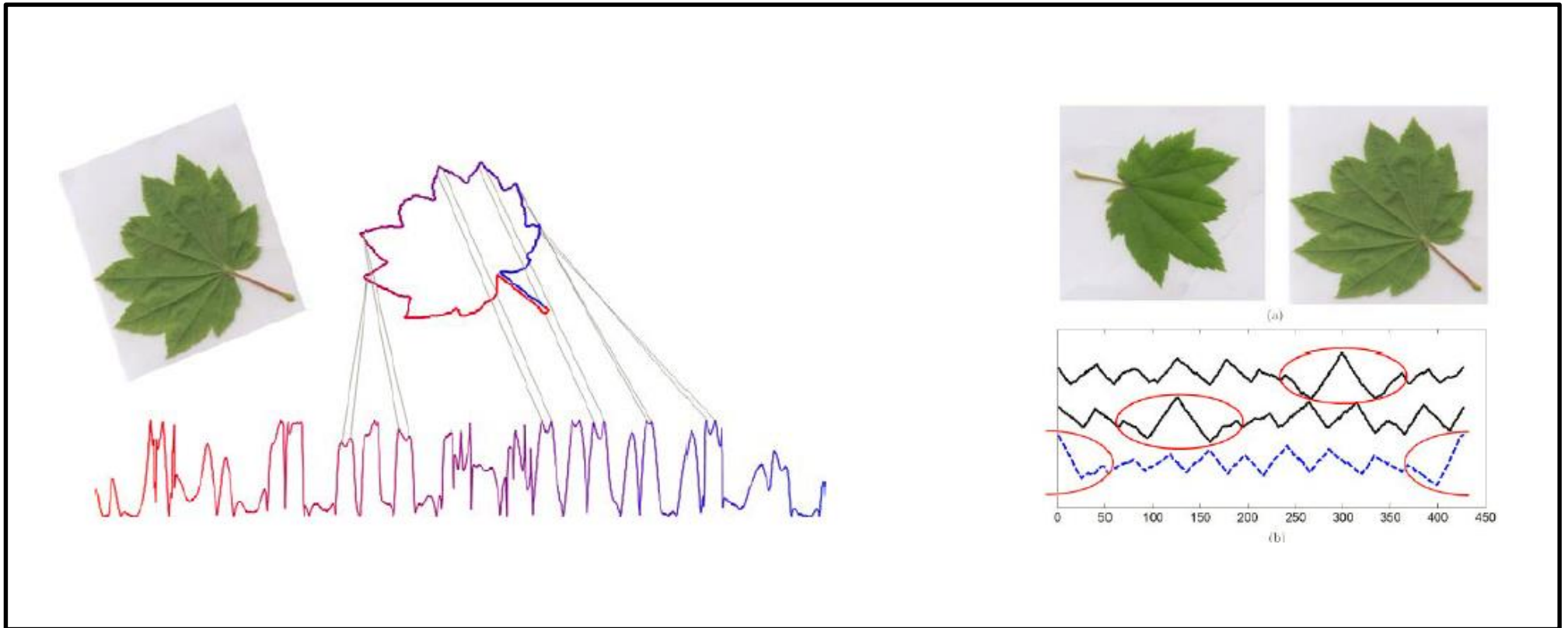
Time Series Analysis

- A word can be represented by two time series created by moving over and under the word



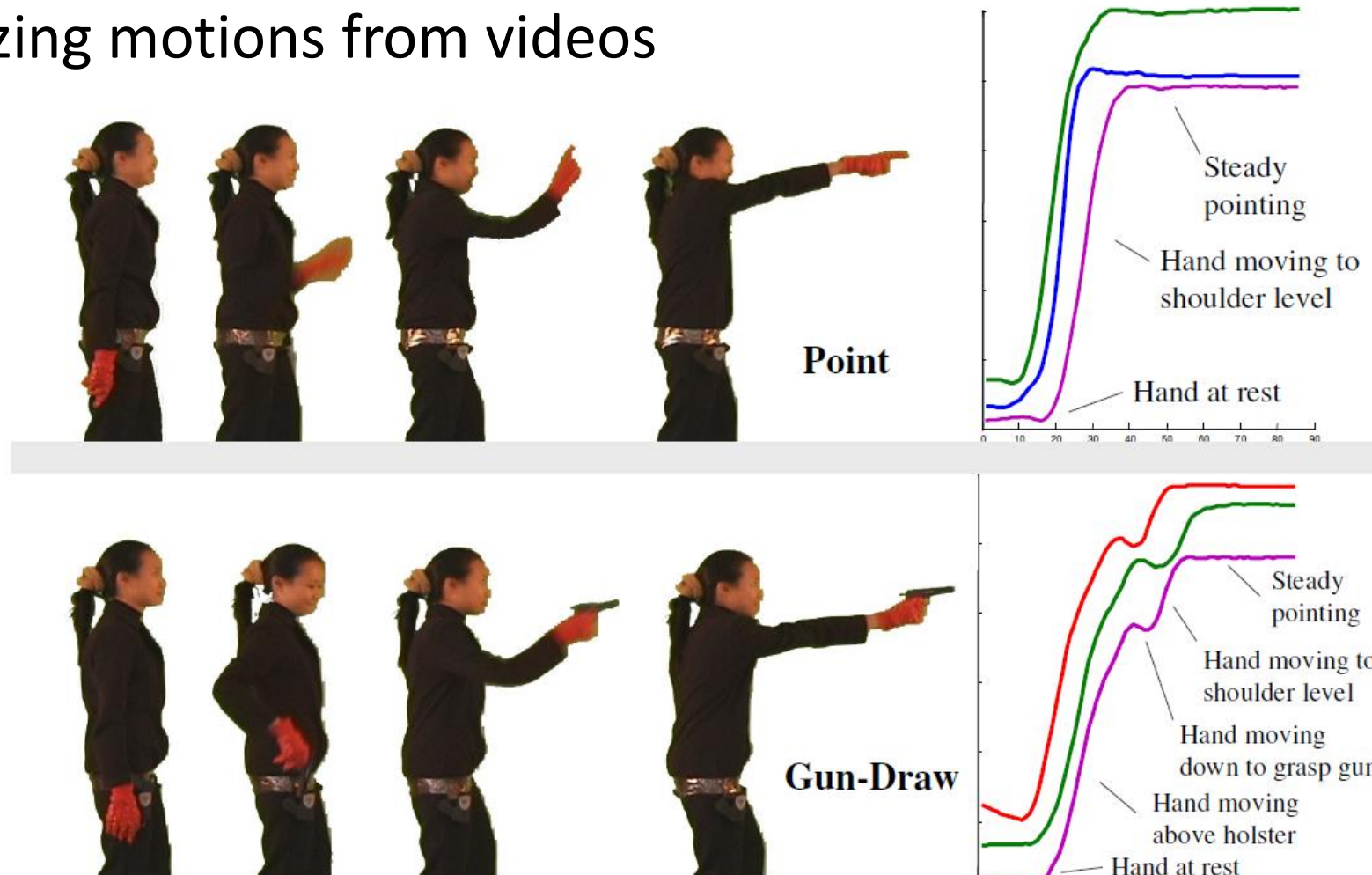
Time Series Analysis

- Recognizing trees from the leaf images



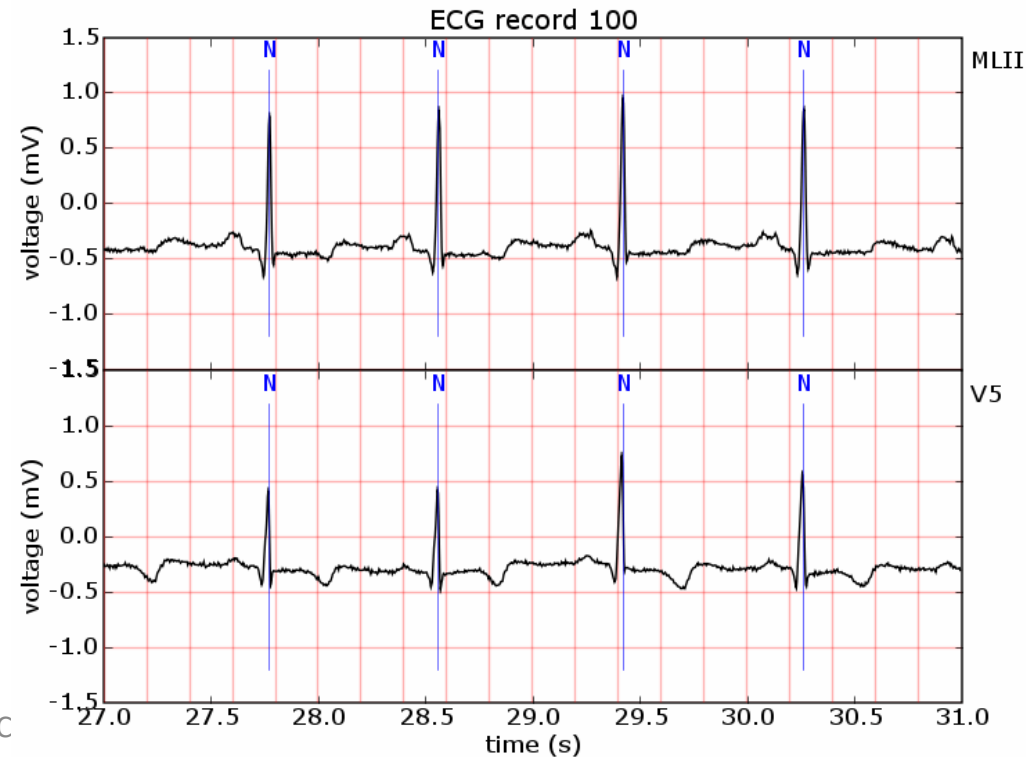
Time Series Analysis

- Recognizing motions from videos



Time Series Mining

- You go to the doctor because of chest pains
- Your ECG looks strange
- The doctor wants to search a database to find similar ECGs to find clues to your condition



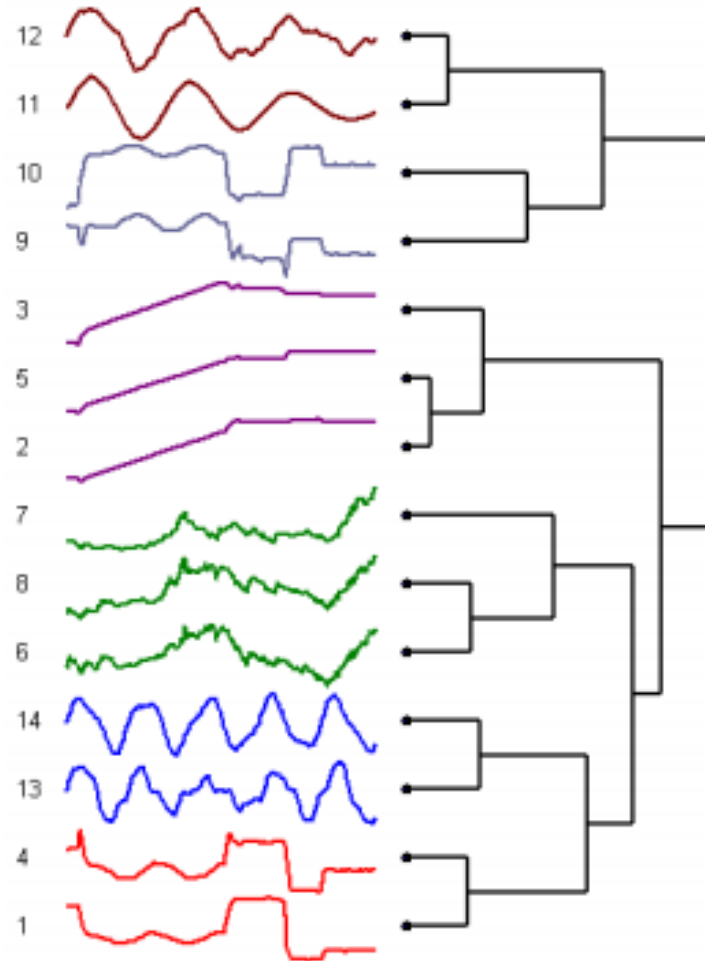
Time Series Analysis

- Some Time Series Data Mining Tasks
 - Clustering
 - Classification
 - Rule Discovery
 - Anomaly Detection
 - Visualization
 - Query by Content



Time Series Clustering

- Identify which time series are similar to each other



Time Series Clustering

- To partition time series data into groups based on similarity or distance
 - Time series data in the same cluster are the same
- First step is to work out an appropriate distance/similarity metric
- Second step is to use existing clustering techniques
 - Hierarchical clustering
 - K-means



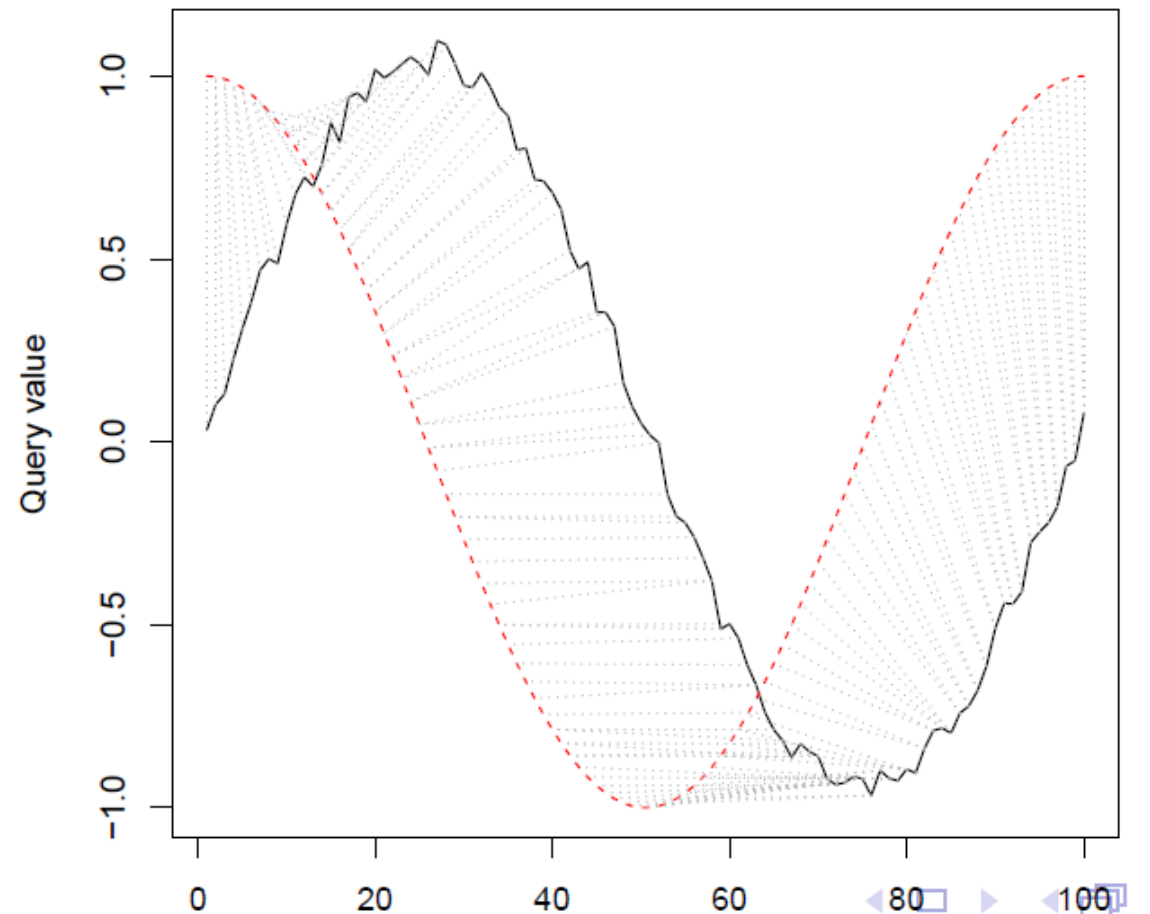
Time Series Clustering

- Measure of distance/similarity
 - Euclidean distance
 - Manhattan distance
 - Maximum norm
 - Hamming distance
 - Inner product
 - Dynamic Time Warping (DTW) distance



Dynamic Time Warping (DTW)

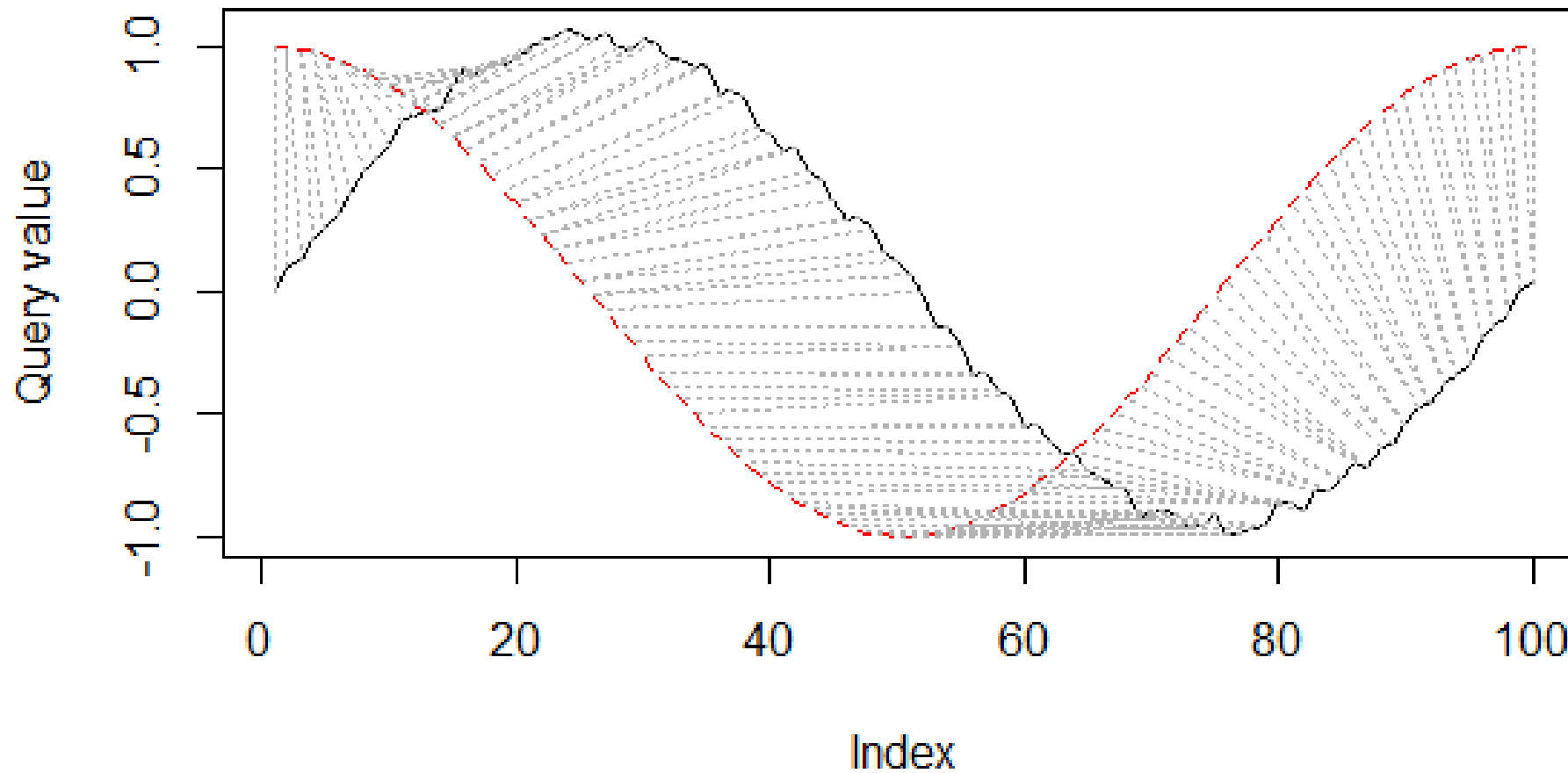
- Finds the optimal non-linear alignment between two time series



Dynamic Time Warping (DTW)

- `library('dtw')`
- `idx <- seq(0, 2*pi, len=100)`
- `aa <- sin(idx) + runif(100)/10`
- `bb <- cos(idx)`
- `align <- dtw(aa, bb, step = asymmetricP1, keep = T)`
- `dtwPlotTwoWay(align)`

Dynamic Time Warping (DTW)



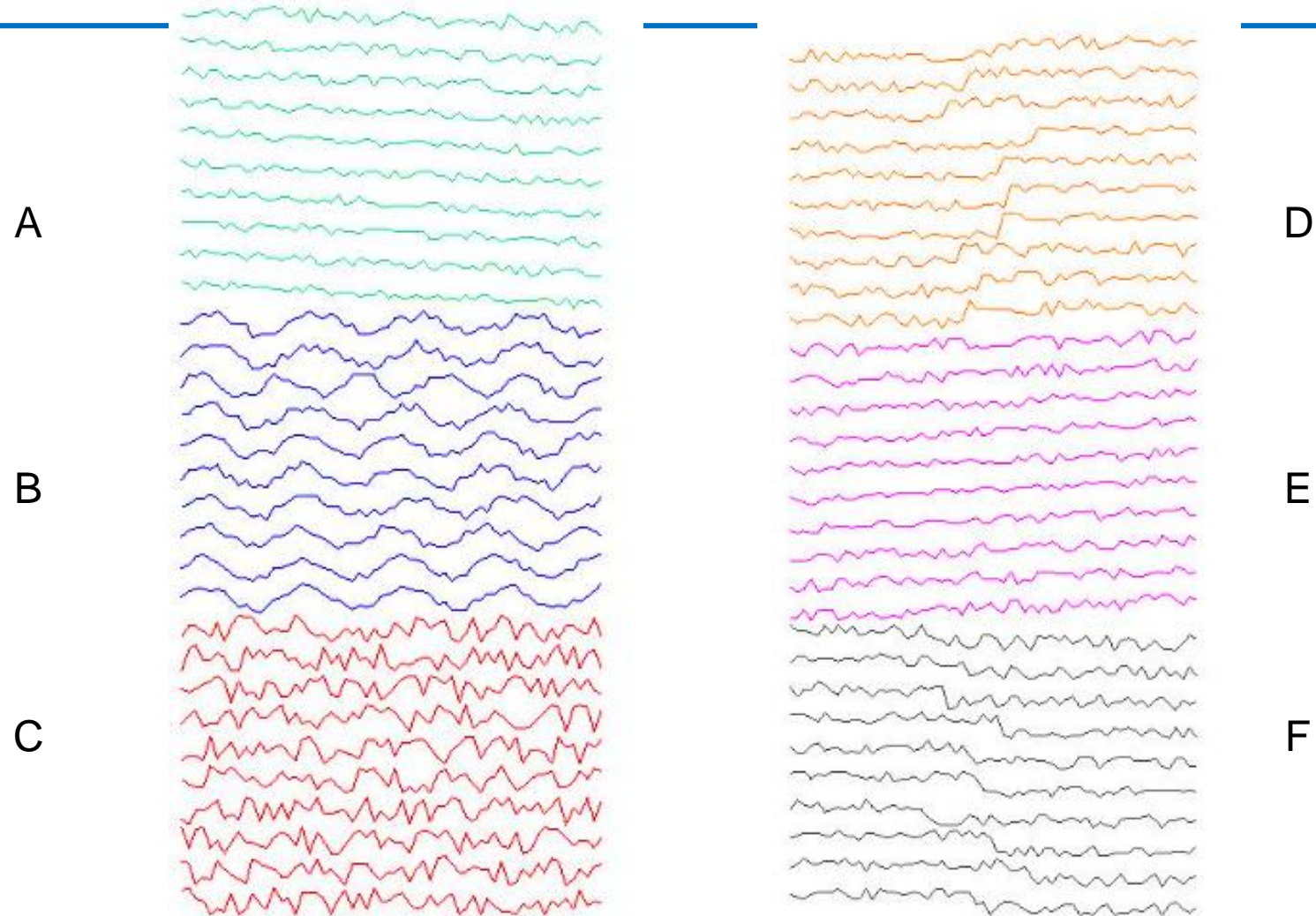
Example

- Synthetic Control Chart Time Series
- Dataset contains 600 examples of control charts synthetically generated by the process in Alcock and Manolopoulos (1999)

Example

- Each control chart is a time series with 60 values
- Six classes are known:
 - 1 – 100 Normal
 - 101 – 200 Cyclic
 - 201 – 300 Increasing Trend
 - 301 – 400 Decreasing Trend
 - 401 – 500 Upward Shift
 - 501 – 600 Downward Shift

Example

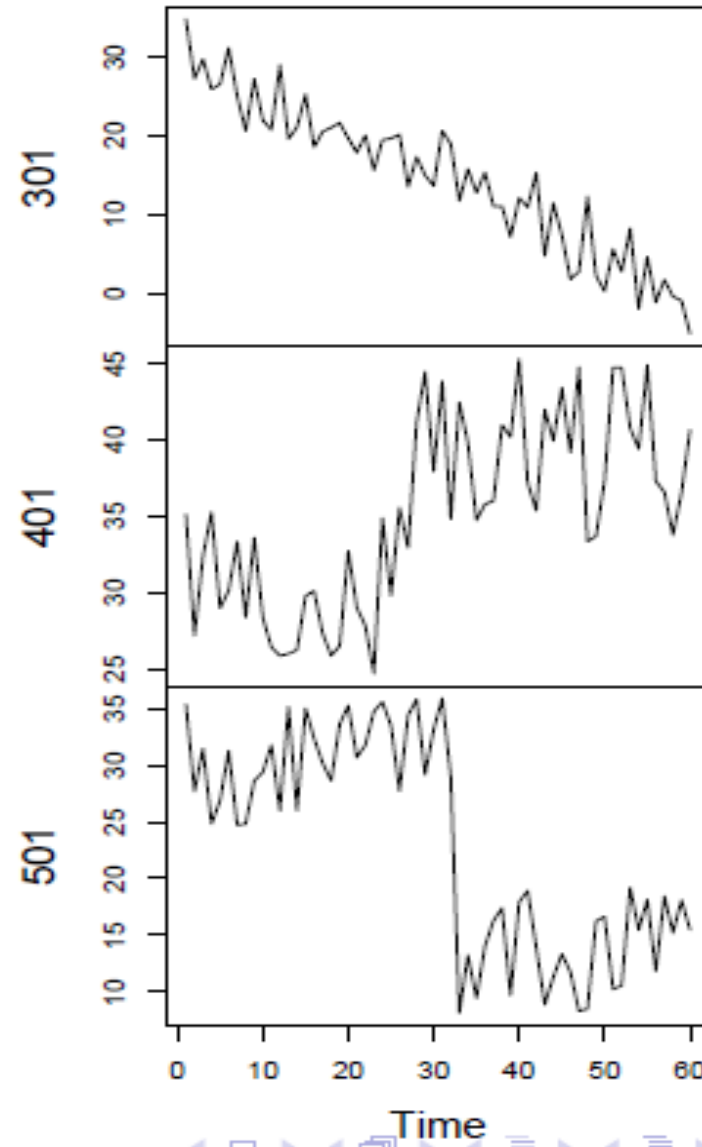
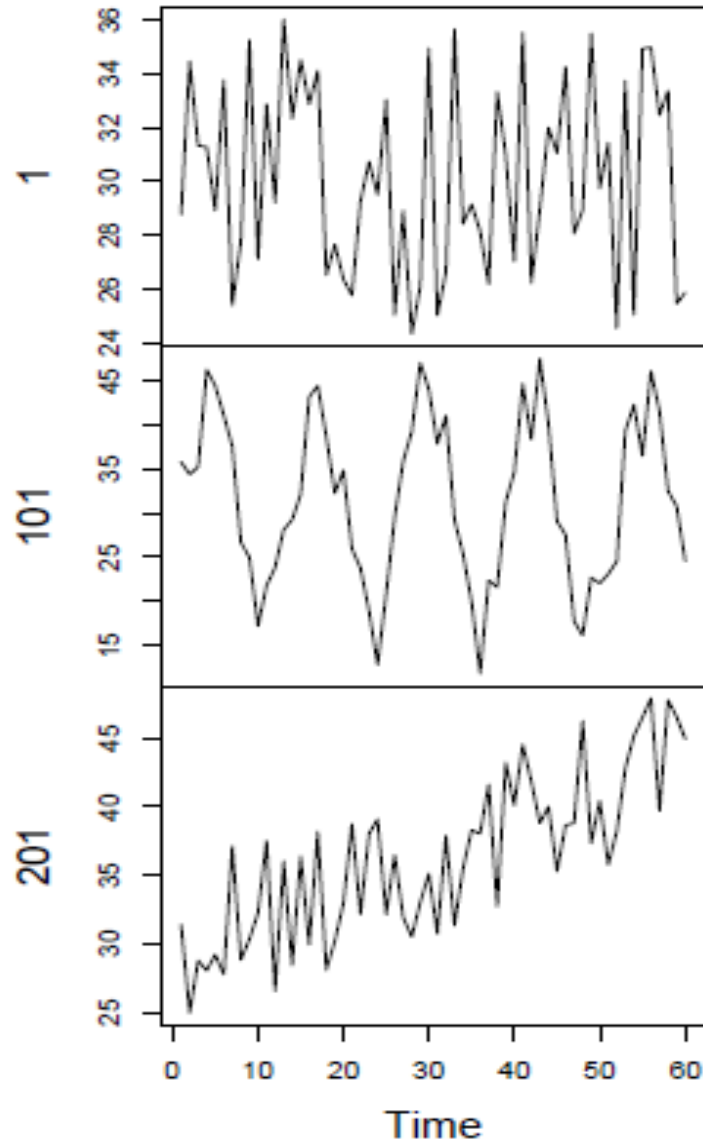


Synthetic Control Chart: R Code

- `#Read Data into R`
- `sc <- read.table("synthetic_control.data",
header=F, sep="")`
- `#Show one sample from each class`
- `idx <- c(1,101,201,301,401,501)`
- `sample1 <- t(sc[idx,])`
- `sample1`
- `plot.ts(sample1, main="")`



Example



Synthetic Control Chart: R Code

- `#Sample n cases from every class`
- `n <- 10`
- `s <- sample (1:100, n)`
- `idx <- c(s, 100+s, 200+s, 300+s, 400+s, 500+s)`
- `sample2 <- sc[idx,]`
- `observedLabels <-
c(rep(1,n), rep(2,n), rep(3,n), rep(4,n), rep(5,n)
, rep(6,n))`

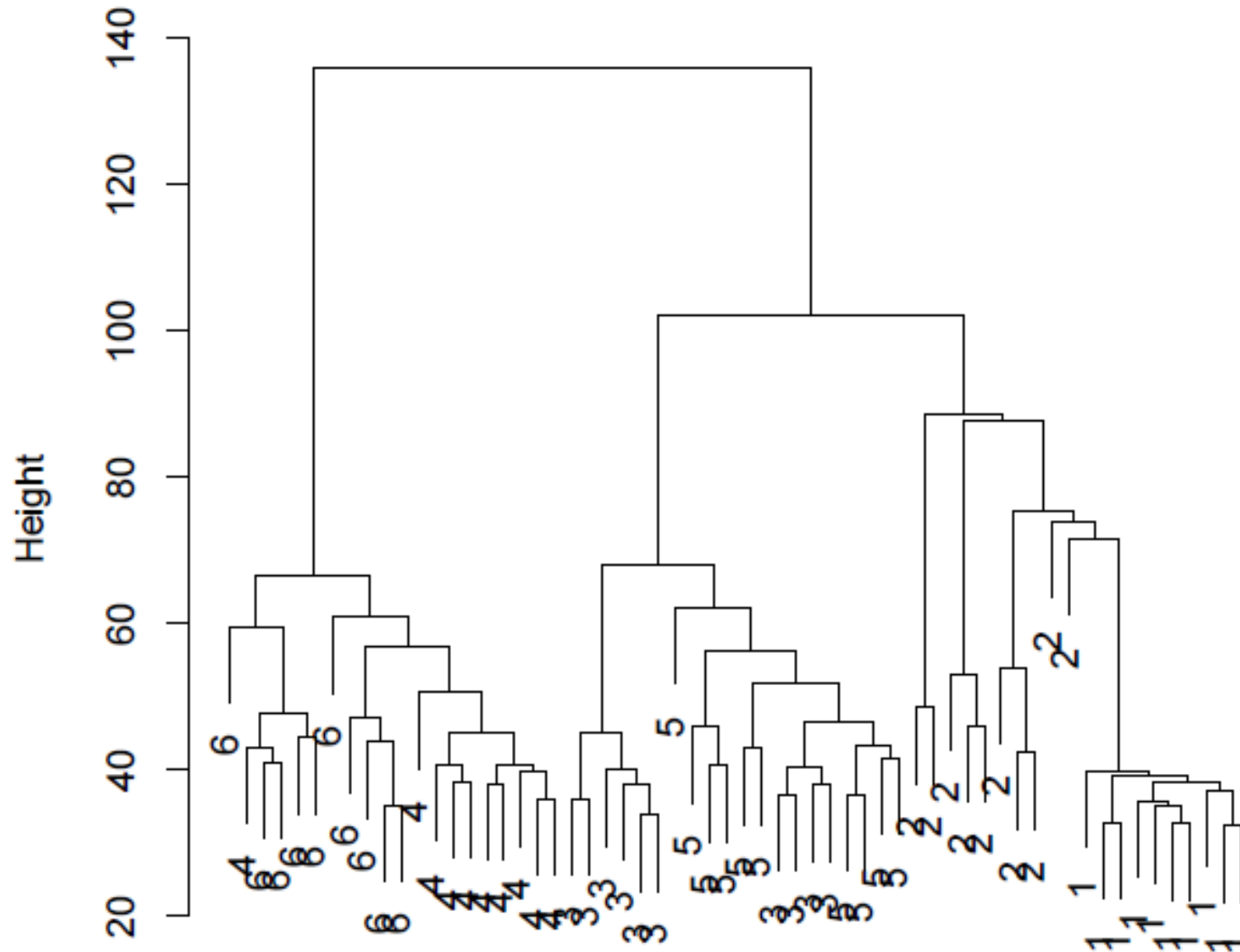
Synthetic Control Chart: R Code

- `#Compute DTW distances`
- `library(dtw)`
- `distMatrix <- dist(sample2, method="DTW")`

Synthetic Control Chart: R Code

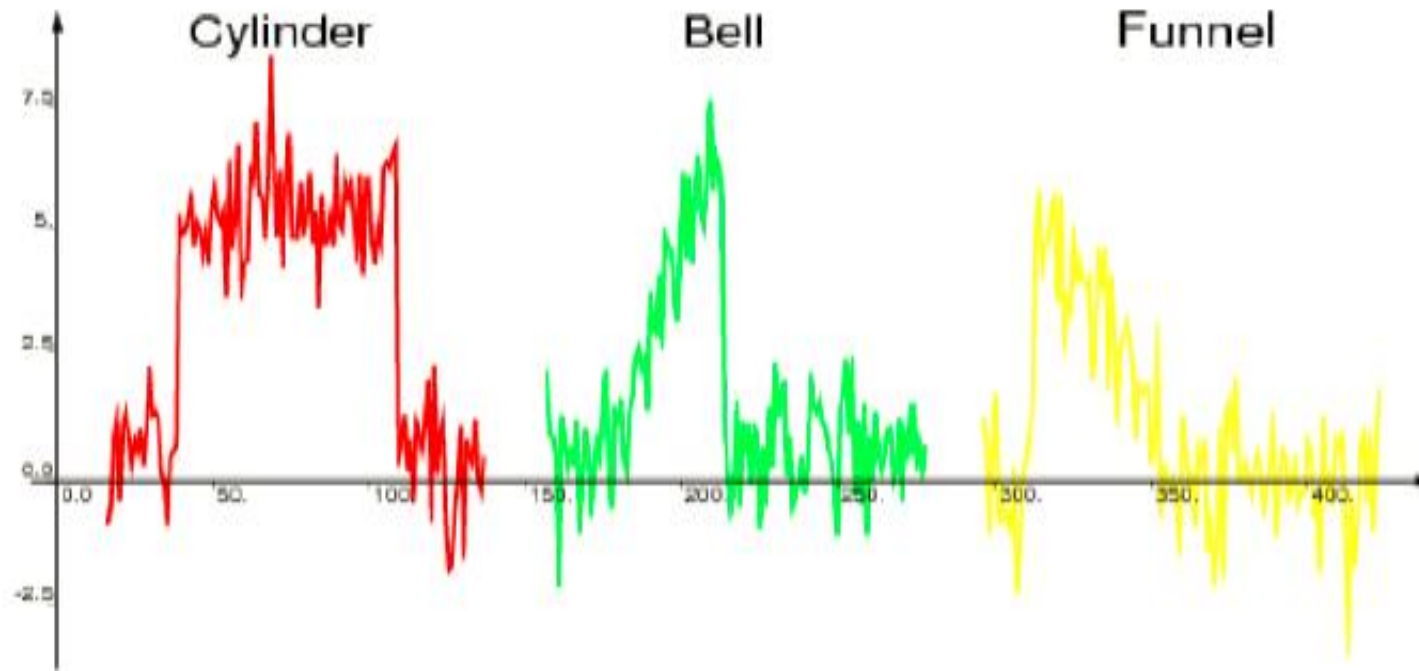
- `#Hierarchical Clustering with Euclidean distance`
- `hc <- hclust(distMatrix, method="ave")`
- `plot(hc, labels=observedLabels, main = "")`

Example



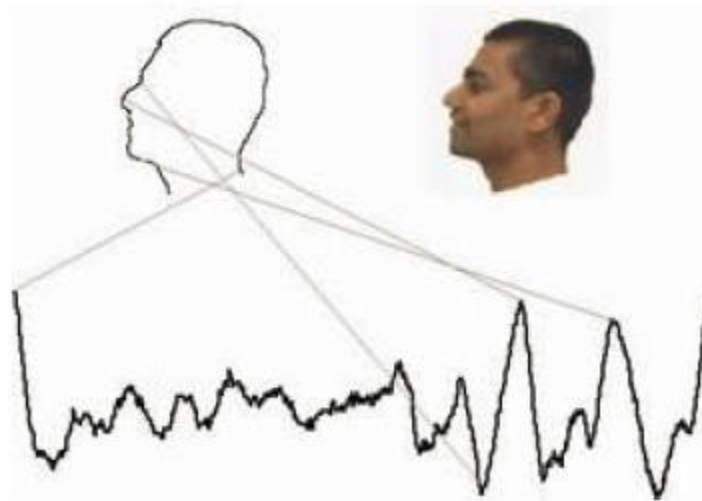
Time Series Classification

- A supervised learning problem aimed at labeling temporally structured univariate (or multivariate) sequences of certain (or variable) length



Time Series Classification: Example

- Task: Classify grad students based on their faces images transformed into “time series”
- Why is difficult?
 - Variation of head angle and expression.
 - Some have glasses/no glasses versions



Time Series Classification: Example

- Build a classification model based on labelled time series
- Use this model to predict unlabeled time series (future)
- Steps include feature extraction and then apply existing classification techniques
 - SVM
 - K-NN
 - Decision trees

Time Series Classification

- Feature Extraction:
 - Singular Value Decomposition
 - Discrete Fourier Transform (DFT)
 - Discrete Wavelet Transform (DWT)
 - Piecewise Aggregate Approximation (PAA)
 - Perpetually Important Points (PIP)
 - Piecewise Linear Representation
 - Symbolic Representation



R Code: Discrete Wavelet Transform (DWT)

- #Extracting DWT coefficients (with Haar filter)
- library(wavelets)
- wtData <- NULL
- for (i in 1:nrow(sc)) {
 a <- t(sc[i,])
 wt <- dwt(a, filter="haar", boundary =
 "periodic")
 wtData <- rbind(wtData,
 unlist(c(wt@W,wt@V[[wt@level]]))) }
- wtData <- as.data.frame(wtData)
- wtData



R Code: Discrete Wavelet Transform (DWT)

- #Set class labels into categorical values
- `classId <- c(rep("1",100), rep("2",100),
rep("3",100),rep("4",100), rep("5",100), rep("6",100))`
- `wtSc <- data.frame(cbind(classId, wtData))`
- `wtSc`



R Code: Discrete Wavelet Transform (DWT)

- # Build a Decision Tree with ctree() in package party
- `ct <- ctree(classId ~ ., data=wtSc)`
- `J48M <- J48(classId ~ ., data=wtSc)`
- `J48M`
- `pClassId <- predict(J48M)`
- `pClassId <- predict(ct)`
- `pClassId`



R Code: Discrete Wavelet Transform (DWT)

Model formula:

```
classId ~ w11 + w12 + w13 + w14 + w15 + w16 + w17 + w18 + w19 +  
          w110 + w111 + w112 + w113 + w114 + w115 + w116 + w117 + w118 +  
          w119 + w120 + w121 + w122 + w123 + w124 + w125 + w126 + w127 +  
          w128 + w129 + w130 + w21 + w22 + w23 + w24 + w25 + w26 +  
          w27 + w28 + w29 + w210 + w211 + w212 + w213 + w214 + w215 +  
          w31 + w32 + w33 + w34 + w35 + w36 + w37 + w41 + w42 + w43 +  
          w5 + v57
```



R Code: Discrete Wavelet Transform (DWT)

Fitted party:

```
[1] root
|
| [2] v57 <= 116.92152
| |
| | [3] w43 <= -3.9333
| | |
| | | [4] w5 <= -8.25063: 4 (n = 67, err = 3.0%)
| | | [5] w5 > -8.25063: 6 (n = 7, err = 28.6%)
| | |
| | | [6] w43 > -3.9333
| | | |
| | | | [7] w31 <= -5.69811: 4 (n = 9, err = 11.1%)
| | | | [8] w31 > -5.69811: 6 (n = 86, err = 26.7%)
| |
| | [9] v57 > 116.92152
| | |
| | | [10] v57 <= 140.30128: 6 (n = 31, err = 6.5%)
| | | [11] v57 > 140.30128
| | | |
| | | | [12] v57 <= 177.87899
| | | | |
| | | | | [13] w22 <= -6.3472: 2 (n = 80, err = 2.5%)
| | | | | [14] w22 > -6.3472
| | | | | |
| | | | | | [15] w31 <= -12.68737: 2 (n = 9, err = 0.0%)
| | | | | | [16] w31 > -12.68737
| | | | | | |
| | | | | | | [17] w36 <= -4.27022: 1 (n = 7, err = 14.3%)
| | | | | | | [18] w36 > -4.27022: 1 (n = 92, err = 0.0%)
| | | |
| | | | [19] v57 > 177.87899
| | | | |
| | | | | [20] w31 <= -14.59748: 2 (n = 12, err = 0.0%)
| | | | | [21] w31 > -14.59748
| | | | | |
| | | | | | [22] w43 <= 3.34525
| | | | | | |
| | | | | | | [23] w32 <= 6.14667: 5 (n = 91, err = 11.0%)
| | | | | | | [24] w32 > 6.14667: 3 (n = 12, err = 25.0%)
| | | | | |
| | | | | | [25] w43 > 3.34525
| | | | | | |
| | | | | | | [26] w123 <= -3.701: 5 (n = 17, err = 47.1%)
| | | | | | | [27] w123 > -3.701
| | | | | | | |
| | | | | | | | [28] w210 <= 6.33655: 3 (n = 73, err = 5.5%)
| | | | | | | | [29] w210 > 6.33655: 3 (n = 7, err = 42.9%)
```

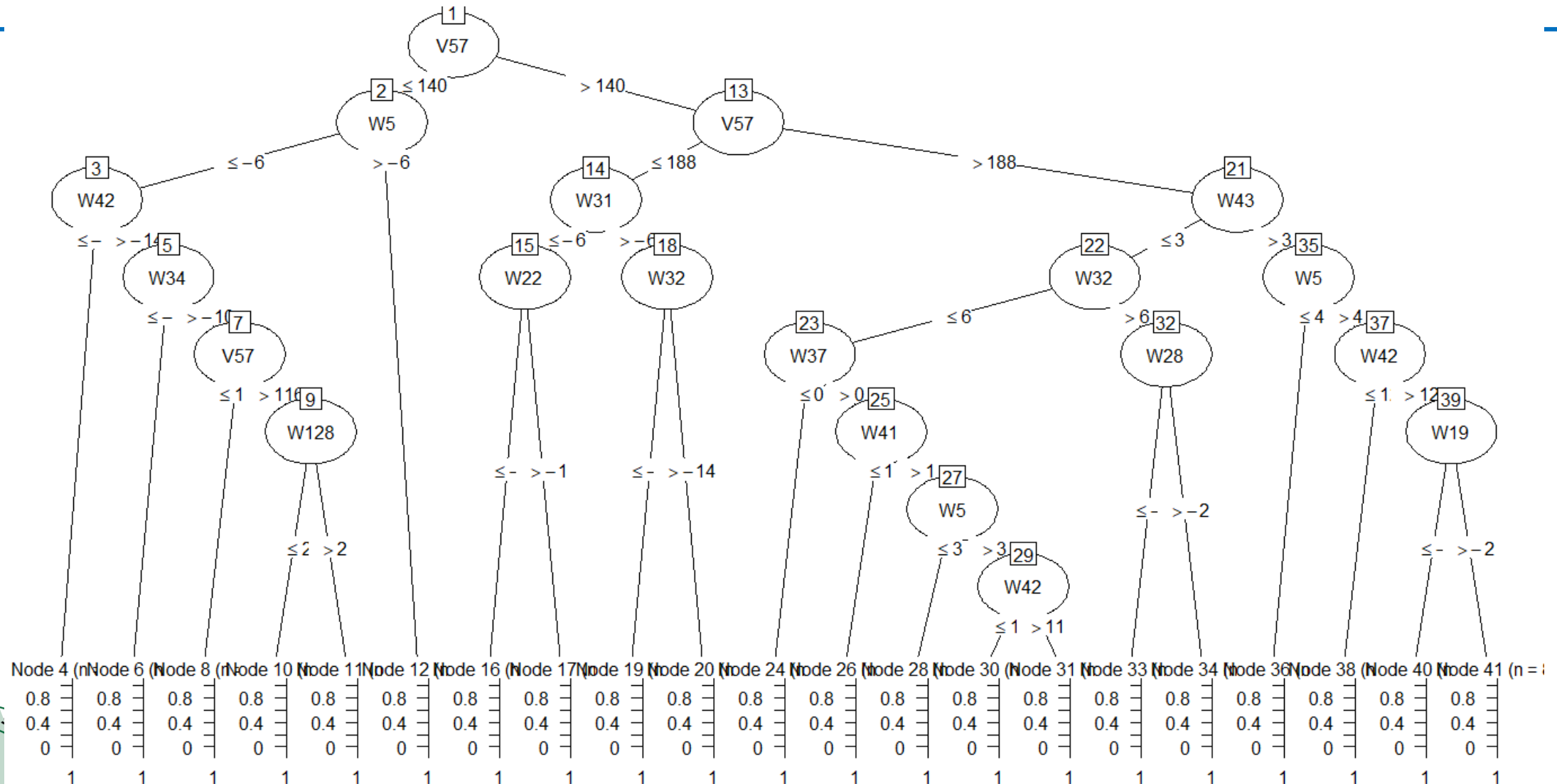
Number of inner nodes: 14
Number of terminal nodes: 15

R Code: Discrete Wavelet Transform (DWT)

- `# Check predicted classes against original class labels`
- `table(classId, pClassId)`
- `# Accuracy`
- `(sum(classId==pClassId)) / nrow(wtSc)`
- `#Plot Tree`
- `plot(ct, ip_args=list(pval=FALSE), ep_args=list(digits=0))`



R Code: Discrete Wavelet Transform (DWT)



Final Note

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

- A free software environment for statistical computing and graphics
- <http://cran.r-project.org/web/views/TimeSeries.html>

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.