Introduction to

Time Series Analysis

Contents

1. What is Time Series
2. Components of Time Series
3. Application Areas
4. Time Series Analysis in R
   i. Plotting a Time Series
   ii. Subsetting a Time Series
5. Irregular Time Series

What is Time Series ?

Time Series is a sequence of values observed over time

Types of Time Scale	
Discrete	Value changes after jumping from one time period to other. Example: Dow Jones Index -End of Day Values
Continuous	Value changes within an infinitely short amount of time. Example. Temperature, Dow Jones Index Tracked Real Time

Components of Time Series

Trend	Long-term increase or decrease in the time series. There may be increase/decrease in short term but overall trend in the long term can be increasing or decreasing.
Seasonality	**Predictable and recurring trends and patterns** over a period of time, normally a year. An example of a seasonal time series is retail data, which sees spikes in sales during holiday seasons like Christmas. Seasonality is reflected only when data is available for more than one year
Cyclic Pattern	Exists when data exhibit rises and falls that are not of fixed period. The duration of these fluctuations is usually of at least 2 years

Application Areas

Industry	Model/Predict	Based on Information such as:	Purpose
Finance	Price of a Stock	• Recent price movement of the stock	Forecasting
Economics	Inflation Rates	• Trend and seasonality in inflation rates	Forecasting
Retail /FMCG	Monthly Sales	• Location, marketing expenses on TV, print and online media	Predictive and Optimization

R packages

R has robust infrastructure for handling and managing time series data. The base R has function “ts” to handle time series data. However, there are many other packages in R to work with time series data.

Package	Features
zoo	• R package for regularly and irregularly spaced time series
xts	• Based on zoo and provides uniform handling of R's different time-based data classes
timeSeries	• Recommended for managing financial time series objects

Case Study

Background

- Annual Sales for a specific company from year 1961 to 2017

Objective

- To plot a time series object

Available Information

- Number of cases: 57
- Variables: Year, sales(in 10's GBP)

Data Snapshot

turnover_annual data

Variables

ons on Discrete Time Scale

Year	sales
1961	224786
1962	230034
1963	236562
1964	250960
1965	261615
1966	268316
1967	283589
1968	280160
1969	301422
1970	308018
1974	364834
1975	392503

Columns	Description	Type	Measurement	Possible values
Year	Financial Year	Numeric	-	-
sales	sales(in 10's GBP)	Numeric	In British Pound	Positive values

Time Series in R

```
# Import turnover_annual Data
turnover_annual <- read.csv("turnover_annual.csv",header=TRUE)
```

- Performing any time series related operations require the data type to be time-based.
- We will use the most basic function **ts()** for converting our data frame object into time series object.

```
#Creating a Time Series Object
salesseries<-ts(turnover_annual$sales,start=1961,end=2017)
```

ts() from package **stats** in R, is a very important function and is used to create time series object. We use **ts()** to convert a column from our data frame **turnover_annual** to time series object named **salesseries.**
The **start=** and **end=** arguments specify first and last period
When the time series has seasonal component, argument **frequency =** can be included in **ts()**. It denotes number of observations per unit of time. Ex. If data is quarterly: **frequency = 4**, if data is monthly: **frequency = 12**.
Frequency argument is not required for annual data.

Plotting Time Series in R

```
# Plotting a Time Series Object
plot(salesseries,col="red",main="Sales Time Series (Simple Plot)")
```

plot() generates a simple line chart.

```
# Output
```

Interpretation :

- The time-series clearly shows upward trend.

Subsetting Time Series in R

- Large volumes of data are required for most real world analytics, time series is no exception.
- Subsetting is an important tool as it facilitates partitioning the data within R for micro-level specific analysis.

```
# Subsetting a Time Series Object
salesseries2<-window(salesseries,start=1990,end=2017)
```

window() is a generic function which extracts the subset of the object x observed between the times **start** and **end**.

Subsetting Time Series in R

```
plot(salesseries2, col="red", main="Sales Time Series (Subset)")
```

```
# Output
```

Interpretation:

- Upward trend is observed for period 1990 to 2017

Irregular Time Series

- The time series considered so far is "Regular" time series. It is observed after regular time interval of one year.
- However, not all time series are evenly spaced. For instance, ATM withdrawal data of a customer is a time series, but the transactions may not happen after uniform intervals like daily, weekly, monthly.
- Irregular time series can be found in a number of industries and domains like ecology (natural disasters), finance (stock prices), clinical research (patient's reactions), astronomy (movement of celestial objects), to name a few.
- But with advancements in computer technology, such uneven time series can be analyzed in their original form.

Handling Irregular Time Series in R :

- It is impossible to coerce irregular time series into `ts()`, because the arguments of start, end and frequency are irrelevant.
- R has two highly effective packages for handling unevenly spaced time series, zoo and xts.

Quick Recap

Time Series	• Sequence of values measured over time • Time scale can be discrete or continuous
Time Series in Analytics	• Used for simple forecasting or building predictive model on time series data
Time Series in R	• R has different packages and functions for handling different types of time series • **`ts()`** in the basic stats package is the most commonly used function • A time series object can also be subsetted using **`window()`** from stats package
Irregular Time Series	• Time Series which are not evenly spaced. • R has two **highly effective packages** for handling unevenly spaced time series, **zoo** and **xts**.

Time Series

Decomposition

Contents

1. Components of Time Series
2. Understanding Moving Averages
3. Time Series Decomposition
   i. Simple Method based on moving averages
   ii. Decomposition Using Local Regression Method

Components of Time Series

- As we know, Trend and Seasonality are main components of Time Series.
- If we assume an additive model, we can write

$$Y_t = S_t + T_t + R_t$$

where,

Y_t : Time series value at period t

S_t : Seasonal component at period t

T_t : Trend-cycle component at period t

R_t : Remainder (or irregular or error) component at period t

- Alternatively, a multiplicative model would be written as

$$Y_t = S_t * T_t * R_t$$

* Additive model is most appropriate if the magnitude of the seasonal fluctuations or the variation around the trend-cycle does not vary with the level of the time series.

Understanding Moving Averages

- Moving Averages are **averages calculated for consecutive data from overlapping subgroups of fixed length**
- Moving averages smoothen a time series by filtering out random fluctuations

Day	End of Day Sales
1	1500
2	2100
3	1750
4	1900
5	1650

Moving Average of Period 3
NA
NA
(1500+2100+1750)/3=1783.33
(2100+1750+1900)/3=1916.67
(1750+1900+1650)/3=1766.67

The first 2 MA values for length 3 are not calculated

- Period of the moving average depends on type of data
- **Non-seasonal data**: Shorter length (Typically 3 period or 5 period MA is considered)
- **Seasonal data**: Typical period is 12 for monthly data and 4 for quarterly data

Time Series Decomposition – Simple Method

Decomposition is a statistical method that deconstructs a time series.

Steps to follow :

Find Trend	Obtain moving averages covering one season – This provides trend component of the time series
Eliminate Trend	Eliminate trend component from original time series. Calculate $Y_t - T_t$
Estimate Seasonality	To estimate the seasonal component for a given time period, simply average the de-trended values for that time period. These seasonal indexes are then adjusted to ensure that they add to zero

The remainder component is calculated by subtracting the estimated seasonal and trend-cycle components

Time Series Decomposition – Simple Method

Suppose we have monthly time series data, for three years 2014, 2015 and 2016:

Step 1 □ Calculate Moving Averages

…	Dec 14	Jan 15	Feb 15	Mar 15	Apr 15	May 15	Jun 15	Jul 15	Aug 15	Sep 15	Oct 15	Nov 15	Dec 15	Jan 16	…

(Consider moving average period of 13 - previous 6 months, next 6 months and current month to calculate moving average of current month)

This gives the trend component T_t

Step 2 □ Eliminating Trend

Remove T_t from the original time series Y_t

Step 3 □ Estimate Seasonal Component

The seasonal index for July is the average of all the de-trended July values in the data i.e. Average of De-trended July 2014, July 2015 and July 2016

Case Study

Background

- Monthly Sales Data for 3 Years (2013, 2014, 2015)

Objective

- To decompose time series into its components and study each component separately.

Available Information

- Sample size is 36
- Variables: Year, Month, Sales

Data Snapshot

Sales Data for 3 Years

Variables

Monthly Observations

Year	Month	Sales
2013	Jan	123
2013	Feb	142
2013	Mar	164
2013	Apr	173
2013	May	183
2013	Jun	192
2013	Jul	199
2013	Aug	203
2013	Sep	207
2013	Oct	209
2013	Nov	214
2013	Dec	255
2014	Jan	223

Columns	Description	Type	Measurement	Possible values
Year	Year	Numeric	2013, 2014, 2015	3
Month	Month	Character	Jan - Dec	12
Sales	Sales in USD Million	numeric	USD Million	Positive values

Time Series Decomposition in R – Simple Method

```
# Time Series Decomposition
salesdata<-read.csv("Sales Data for 3 Years.csv",header=TRUE)
salesseries<-ts(salesdata$Sales,start=c(2013,1), end=c(2015,12),
frequency=12)
```

- **ts()** converts a column from a data frame to a simple time series object.
- **start=** and **end=** arguments specify the x-axis scale. (Year and month in this case).
- **frequency=12** tells that we have monthly data

```
decomp<-decompose(salesseries)
plot(decomp)
```

- **decompose()** performs classical seasonal decomposition through moving averages.
- **plot()** of decompose object gives a 4-level visual representation.

Time Series Decomposition in R – Simple Method

```
# Output
```

Interpretation :

- Trend not calculated for first/last few values
- Seasonal Component repeats from year to year

* The decompose() function takes time series as additive by default. If you wish to create a multiplicative time series, simply add type="multiplicative" in the function.

Time Series Decomposition in R – Simple Method

```
#Analysing the decompose() object. Each component can be separately
viewed by using the $ operator
```

```
decomp$seasonal
```

```
           Jan        Feb        Mar        Apr        May        Jun        Jul        Aug        Sep        Oct
2013 -5.0069444 -1.1819444 -3.7361111 -1.5881944 -2.8319444 -2.2319444  0.2347222 -1.5256944 -4.0944444 -6.4465278
2014 -5.0069444 -1.1819444 -3.7361111 -1.5881944 -2.8319444 -2.2319444  0.2347222 -1.5256944 -4.0944444 -6.4465278
2015 -5.0069444 -1.1819444 -3.7361111 -1.5881944 -2.8319444 -2.2319444  0.2347222 -1.5256944 -4.0944444 -6.4465278
           Nov        Dec
2013 -7.1277778 35.5368056
2014 -7.1277778 35.5368056
2015 -7.1277778 35.5368056
```

```
decomp$trend
```

```
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
2013       NA       NA       NA       NA       NA       NA 192.8333 200.7708 207.4500 213.1958 218.4083 223.0167
2014 227.1667 230.9625 234.5500 237.9292 241.1000 244.5167 247.8917 250.5750 252.9333 254.9917 257.0417 259.1042
2015 261.0417 262.9958 264.9167 266.8417 268.7583 270.8417       NA       NA       NA       NA       NA       NA
```

```
decomp$random
```

```
             Jan         Feb         Mar         Apr         May         Jun         Jul         Aug         Sep
2013          NA          NA          NA          NA          NA          NA  5.93194444  3.75486111  3.64444444
2014  0.84027778  2.71944444  2.98611111  4.75902778  1.73194444  3.31527778 -3.12638889 -0.94930556 -0.83888889
2015  1.96527778  0.08611111 -0.18055556 -1.95347222  1.07361111 -0.50972222          NA          NA          NA
             Oct         Nov         Dec
2013  2.25069444  2.71944444 -3.55347222
2014  0.55486111  0.08611111  6.35902778
2015          NA          NA          NA
```

?

Why are NAs getting generated?

This is because trend not estimated for first/last few values. Consequently, the same will be reflected in random component as well.

Time Series Decomposition in R – Simple Method

```
# Doing Seasonal Adjustment
seasadj <- salesseries - decomp$seasonal
plot(seasadj)
```

```
# Output
```

Interpretation :

- This plot shows seasonally adjusted time series

Time Series Decomposition – Local Regression Method (LOESS)

Loess is the short form of Local Regression method

- Time series decomposition by Loess method not only seasonally adjusts a series but it also helps in modeling the series and understanding all the components of variability in the series.
- It is a non parametric generalisation of ordinary least squares regression.
- The fitting technique does not require a priori specification of the relationship between the dependent and independent variables.

STL is an acronym for "Seasonal and Trend Decomposition using Loess"

- STL works by iterating through smoothing of the seasonal and trend components. STL is a procedure for regular time series, so the design points of the smoothing operation are equally-spaced.

Time Series Decomposition – Local Regression Method (LOESS)

```
#Local Regression Method for Seasonal Decomposition
decomp<-stl(salesseries,s.window="periodic")
```

- **stl()** in used to carry out decomposition by loess method.
- **t.window=** specifies trend window. It takes the span (in lags) of the loess window for trend extraction, which should be odd. If NULL, the default, **nextodd(ceiling((1.5*period) / (1-(1.5/s.window))))** is taken.
- **s.window=** specifies seasonal window. It uses either the character string "periodic" or the span (in lags) of the loess window for seasonal extraction.

```
plot(decomp)
```

Time Series Decomposition – Local Regression Method (LOESS)

```
# Output
```

Interpretation:

- The data shows trend and seasonality.
- Note that trend values are estimated for all periods (unlike decompose function)

Quick Recap

What is Decomposition	• A time series is made up of multiple components such as seasonality, trend, randomness. • Sometimes, studying these components separately provides a more comprehensive insight about the series.
LOESS Method	• Loess is a method for estimating nonlinear relationships while STL is an acronym for Seasonal and Trend decomposition using Loess.
Decomposition in R	• **`decompose()`** carries out simple seasonal decomposition • **`stl()`** is used for doing decomposition by LOESS method

Time Series Analysis

Stationarity of Time Series - I

Contents

1. What is Stationarity ?
2. Stationary Time Series – White Noise Process
3. Non-Stationary Time Series – Random Walk
4. Importance of Stationary Time Series
5. Identifying Stationary Time Series
6. Concept of Autocorrelation & Correlograms

What is Stationarity of Time Series ?

Time series process is called **Stationary if statistical properties of the process remain unchanged over time.**

If Y_t is a **stationary** time series where t=1,2,3,...

then,

$E(Y_t) = \mu_t = \mu$ (constant)

$Var(Y_t) = \sigma_t^2 = \sigma^2$ (constant)

$cov(Y_t, Y_{t+s})$ depends only on **s** (lag), and is independent of **t** (time)

Stationary vs. Non-Stationary Time Series

Fig. 1: Stationary Time Series

Fig. 2: Non-Stationary Time Series

Interpretation :

- A stationary time series has a constant long term mean and variance.
- The first diagram shows a stationary time series whereas the second shows a non-stationary series.

Stationary Time Series - White Noise Process

- **White noise** is the simplest example of stationary time series.
- White Noise time series has **zero mean, constant variance and zero covariance with lagged time series.**

Fig. 3: White Noise

Non Stationary Time Series - Random Walk

- **Random Walk** is the simplest case of Non-stationary time series. It is of the form

$$Y_t = Y_{t-1} + U_t$$

$$t=1,2,3.....$$

- We assume that U_t is a random series with,
  - Constant mean μ
  - Constant variance σ^2
  - Serially uncorrelated

Value of Y at time t is equal to its value at time (t-1) plus a random shock

Why Random Walk is Non Stationary

Let $Y_t = 0$ at time $t = 0$

$$Y_1 = U_1$$

$$Y_2 = Y_1 + U_2 = U_1 + U_2$$

$$Y_3 = Y_2 + U_3 = U_1 + U_2 + U_3$$

$$\text{So, } Y_t = \sum U_t \qquad t = 1, 2, 3 \ldots$$

$Yt = \sum Ut$

$E(Yt) = E(\sum Ut) = t \cdot \mu$

i.e. Mean is not constant

$Var(Yt) = Var(\sum Ut) = t \cdot \sigma 2$

i.e. Variance is not constant

Therefore, Random Walk is a non-stationary time series

Importance of Stationary Time Series

- **Calibration** (estimation of model parameters using historical data) is an important concept in the **forecasting** of time series values.

- In the calibration of time series models we need a stationary time series.

- With a non stationary time series we get into **spurious** regression which badly affects forecasting.

How to Make a Non Stationary Time Series Stationary?

Two Methods for Making Time Series Stationary

Differencing

$$Y_t = Y_{t-1} + U_t \quad ;t=1,2,3.....$$

U_t is a random series with **Constant mean** μ, **Constant variance** σ^2 , and is **serially uncorrelated i.e** (Ut is stationary). Hence, Y_t is differenced:

$$Y_t - Y_{t-1} = \Delta Y = U_t$$

Differencing can be well applied in case of stochastic time series

De-trending

$$Y_t = \beta_1 + \beta_2 t + U_t \quad ;t=1,2,3.....$$

U_t is a stationary with **zero mean** and **constant variance** σ^2 . When Trend element ($\beta_1 + \beta_2 t$) is subtracted , the result is a stationary process :

$$Y_t - (\beta_1 + \beta_2 t) = U_t$$

De-trending is useful when trend is deterministic

Identifying Stationary Time Series

& Concept of
Autocorrelation

Autocorrelation of Lag 1

Data

Year	Sales(m)
Jan	110
Feb	113
Mar	121
Apr	123
May	126
Jun	120
Jul	119
Aug	127
Sep	129
Oct	131
Nov	130
Dec	132

First Order Lag

Year	Yt	Yt-1
Jan	110	
Feb	113	110
Mar	121	113
Apr	123	121
May	126	123
Jun	120	126
Jul	119	120
Aug	127	119
Sep	129	127
Oct	131	129
Nov	130	131
Dec	132	130

Autocorrelation of Lag 1, where $\overline{Y}$ = **Avearge of sales**

$$\sum_{t=2}^{n} \frac{(Y_t - \overline{Y})(Y_{t-1} - \overline{Y})}{\sum_{t=1}^{n}(Y_t - \overline{Y})^2}$$

- **Autocorrelation** is a correlation between a time series (Y_t) and another time series representing lagged values of the same time series.(Y_{t-k})

Autocorrelation Function & Correlogram

- Plot of the **sample autocorrelation function against lag is called Correlogram.**
- ACF (autocorrelation function) is a general expression for lag k autocorrelation.
- Correlogram is mainly used in checking stationarity of a series.

r_K (Sample autocorrelation of lag K)

$$= \sum_{t=k+1}^{n} \frac{(Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2}$$

Correlogram for Checking Stationarity

- For a non-stationary time series showing trend, **slow decay pattern will be observed.**
- If a time series is characterized by seasonal fluctuations, then the **correlogram would also exhibit oscillations** at the same frequency.

Fig. 4:
Correlogram

Sample correlogram of *LGDPI*

This figure presents the sample correlogram.

Partial Autocorrelation Function

- Partial autocorrelation (PACF) is the

> Autocorrelation between y_t and y_{t-h} after removing any linear dependence on $y_1, y_2, \ldots, y_{t-h+1}$

- PACF gives **partial autocorrelations at various lags.**
- PACF is **mainly used to identify order of moving average present in the process.**

Partial Correlogram

Fig. 5: Partial ACF Plot

- Y-Axis = Partial Autocorrelation Function
- X-Axis = Lag
- Here we see that, after lag 1, the PACF drops dramatically and all PACFs after lag 1 are statistically insignificant.

Quick Recap

What is Stationarity	• Time series is stationary if the statistical properties of the process remain unchanged over time • Time series which has zero mean, constant variance and zero covariance with lagged time series is known as **"White Noise"**
Random Walk	• Is the simplest form of non-stationary time series where $Y_t - Y_{t-1} = U_t$ (U_t is a random error term)
Making a Non-Stationary Time Series Stationary	• Two approaches: **Differencing** (when time series is stochastic) and **De-trending** (when trend is deterministic)
Identifying Stationarity of Time Series	• **Graphical method:** ACF and PACF Correlograms • **Analytical method:** Dickey Fuller Test

Time Series Analysis

Stationarity of Time Series - II

Contents

1. Case Study
2. Creating and Plotting Time Series in R
3. Checking Stationarity in R
4. Dickey Fuller (DF) Test –
   i. DF Test
   ii. Drawbacks
   iii. Augmented DF Test

Case Study

Background

- Annual Sales for a specific company from year 1961 to 2017

Objective

- To assess stationarity of time series

Available Information

- Number of cases: 57
- Variables: Year, sales(in 10's GBP)

Data Snapshot

turnover_annual data

Variables

ons on Discrete Time Scale

Year	sales
1961	224786
1962	230034
1963	236562
1964	250960
1965	261615
1966	268316
1967	283589
1968	280160
1969	301422
1970	308018
1971	330835
1974	364834
1975	392503

Columns	Description	Type	Measurement	Possible values
Year	Financial Year	Numeric	-	-
sales	sales(in 10's GBP)	Numeric	In British Pound	Positive values

Creating and Plotting Time Series in R

```
#Importing the Data
turnover_annual <- read.csv("turnover_annual.csv",header=TRUE)
#Creating and Plotting a Time Series Object
salesseries<-ts(turnover_annual$sales,start=1961,end=2017)
plot(salesseries,col="red",main="Sales Time Series (Simple Plot)")
```

ts() converts a column from a data frame to a simple time series object. The **start=** and **end=** arguments start and end period.

Interpretation :

- The time-series clearly shows a positive trend.

Checking Stationarity – Correlogram

```
# ACF Plot
acf(salesseries,col="blue")

# Output
```

- **acf()** returns correlogram with X axis indicating lag number and Y axis autocorrelation value

Interpretation :

- We can observe that there is a very slow decay which is a sign of Non-stationarity.

Plot of 1st Order Differenced Time Series

```
# Creating and Plotting a Differenced Series
salesdiff<-diff(salesseries,difference=1)

plot(salesdiff,main="Sales Differenced Time Series")
```

- **diff()** returns suitably lagged differences.
- **difference=** an integer indicating the order of the difference. Here order is 1.

```
# Output
```

Interpretation :

- Even after first order differencing, the series looks non-stationary.

Correlogram for 1st Order Differenced Time Series

```
# ACF Plot
acf(salesdiff,col="blue")
# Output
```

Interpretation :

- Intuitive conclusion about the non-stationarity even after differencing is confirmed in this ACF plot as the decay is still slow.
- Stationarity is not achieved with first difference.

Determining Order of Differencing

```
# Install & load package “forecast”
install.packages("forecast")
library(forecast)
ndiffs(salesseries)
[1] 2

salesdiff2<-diff(salesseries,diff
plot(salesdiff2,col="red")
```

- **ndiffs()** by default uses a Unit Root test to estimate the number of differences required.

Interpretation :

- After 2nd order differencing, the series looks **stationary**.

Correlogram for 2nd Order Differenced Time Series

```
# ACF Plot
acf(salesdiff2,col="blue")
```

```
# Output
```

Interpretation :

- Intuitive conclusion about stationarity is confirmed in this ACF plot.
- Stationarity is achieved with 2nd order difference.

Analytical Method – Dickey Fuller (DF) Test

- A linear stochastic process has a unit root if "1" is the root of the process's characteristic equation. Such a process is non-stationary.
- Dickey and Fuller pioneered idea of testing for unit roots for stationarity checking.

Consider X_t (t=1,2,3,...) is a time series of the form

$$X_t = \rho X_{t-1} + U_t \quad \ldots\ldots\ldots\ldots\ldots\ldots(1)$$

If $\rho=1$ then X_t becomes a random walk

- We assume that $U_t \sim IID(0, \sigma^2)$, i.e U_t is a white noise
- Therefore, we are interested in testing for $\rho=1$

* IID means independent and identically distributed

Dickey Fuller (DF) Unit Root Test

Objective	To test the **null hypothesis** that **time series is not stationary**

Null Hypothesis H_0: $\rho=1$

Alternate Hypothesis H_1: $\rho< 1$

$(X_t-X_{t-1}) =\Delta X_t=(\rho-1) X_{t-1}+U_t$.....from (1)

H_0: $\rho^*=0$, H_1: $\rho^*<0$, $\rho^*=(\rho-1)$

Test Statistic	$(\rho^*/SE(\rho^*))$ Test statistic follows DF distribution under null
Decision Criteria	Reject the null hypothesis **tcal < DF table value**

Dickey Fuller Test

```
# Install & load package "urca" for executing Dickey Fuller Test
install.packages("urca")
library(urca)
df<-ur.df(salesseries, lag=0)
summary(df)
```

ur.df() performs a Dickey Fuller unit root test on time series data.

```
# Output
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1)

Residuals:
   Min     1Q Median     3Q    Max
-73514 -19075  -8490   3412  76065

Coefficients:
        Estimate Std. Error t value Pr(>|t|)
z.lag.1 0.064334   0.003338   19.27   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25950 on 55 degrees of freedom
Multiple R-squared:  0.871,     Adjusted R-squared:  0.8687
F-statistic: 371.5 on 1 and 55 DF,  p-value: < 2.2e-16


Value of test-statistic is: 19.2745

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.6 -1.95 -1.61
```

Interpretation :

- Time series is non-stationary as value of test statistic is greater than 5% critical value.

Dickey Fuller Test

```
# checking stationarity for series with difference of order 2
df2 <- ur.df(salesdiff2,lags = 0)
summary(df2)
```

```
# Output
```

```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1)

Residuals:
   Min     1Q Median     3Q    Max
-69843  -8676   7079  22229  84580

Coefficients:
        Estimate Std. Error t value Pr(>|t|)
z.lag.1  -1.4639     0.1229  -11.91   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 27740 on 53 degrees of freedom
Multiple R-squared:  0.7279,    Adjusted R-squared:  0.7228
F-statistic: 141.8 on 1 and 53 DF,  p-value: < 2.2e-16


Value of test-statistic is: -11.9083

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.6 -1.95 -1.61
```

Interpretation :

- Time series is stationary as value of test statistic is less than 5% critical value.

Dickey Fuller Test – Drawbacks

- If a simple AR(1) model is used when in fact X_t follows an AR(p) process then the error term will be autocorrelated.
- Autocorrelated errors will invalidate the use of DF distributions, as it assumes errors is white noise.

Augmented Dickey-Fuller Test is a measure to fix this problem.

- The most widely offered criticism of the DF-test, however, is its tendency to over-reject the null hypothesis when its true and under-reject the null hypothesis when its false

* AR(1) is autoregressive process is the first-order process

Augmented Dickey Fuller Test

$$X_t = \alpha_0 + \rho_1 X_{t-1} + \rho_2 X_{t-2} + \rho_3 X_{t-3} + \ldots + \rho_p X_{t-p} + U_t.$$

$$\Delta X_t = \alpha_0 + \alpha^* X_{t-1} + \phi_1 \Delta X_{t-1} + \phi_2 \Delta X_{t-2} + \ldots + \phi_p \Delta X_{t-p+1} + U_t.$$

$$\phi_1 = -(\rho_2 + \rho_3 + \ldots + \rho_p),\ \alpha^* = (\alpha_1 + \alpha_2 + \ldots + \alpha_p) - 1$$

Null Hypothesis H_0: $\rho^* = 0$
Alternate Hypothesis H_1: $\rho^* < 0$
$\rho^* = (\rho - 1)$

Test Statistic	$(\rho^*/SE(\rho^*))$ Test statistic follows DF distribution under null
Decision Criteria	Reject the null hypothesis if **tcal < DF table value**

Quick Recap

Correlograms	• **`acf()`** and **`pacf()`** functions in base R generate Correlograms
Differencing a Time Series	• Simple numeric function **`diff()`** can be used to difference a time series • **`ndiffs()`** function generates order of differencing by applying the Dickey Fuller test
Dickey Fuller Test	• **`ur.df()`** function from the package **`urca`** performs a Dickey Fuller test • The output gives test statistic and critical values for the test statistic

Time Series Modeling –

ARIMA Model

Contents

1. Box-Jenkins (ARIMA) Models
2. Five Step Iterative Procedure
   i. Stationarity Checking
      - Differencing, Correlograms, and Dickey Fuller Test in R
   ii. Model Identification
   iii. Parameter Estimation
      - Simple and Automated Model Estimation in R
      - Running ARIMA in R
   iv. Diagnostic Checking
      - Box Pierce Test in R
   v. Forecasting
      - Predictions on ARIMA Model in R

Box-Jenkins (ARIMA) Models

- ARIMA (Auto Regressive Integrated Moving Average) models are Regression models that use lagged values of the dependent variable and/or random disturbance term as explanatory variables.

- ARIMA models rely heavily on the autocorrelation pattern in the data.

- ARIMA models can also be developed in the presence of seasonality in the time series.

- ARIMA models thus essentially ignore domain theory (by ignoring “traditional” explanatory variables)

When to Use ARIMA Models

- Little or nothing is known about the dependent variable being forecasted
- The independent variables known to be important cannot be forecasted effectively
- Objective is to obtain short term forecasts

Basic ARIMA Models

1. Autoregressive model of order p (AR(p)):

$$\boxed{y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t}$$

Where y_t depends on its *p* previous values

2. Moving Average model of order q (MA(q))

$$\boxed{y_t = \delta + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}}$$

y_t depends on *q* previous random error terms

3. Autoregressive-Moving Average model of order p and q (ARMA(p,q))

$$\boxed{\begin{gathered} y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} \\ + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \end{gathered}}$$

Five-Step Iterative Procedure

Step 1: Stationarity Checking

Differencing

- Differencing continues until stationarity is achieved

$$\Delta y_t = y_t - y_{t-1}$$

$$\Delta^2 y_t = \Delta(\Delta y_t) = \Delta(y_t - y_{t-1}) = y_t - 2y_{t-1} + y_{t-2}$$

- The differenced series has n-1 values after taking the first-difference, n-2 values after taking the second difference, and so on

- The number of times that the original series must be differenced in order to achieve stationarity is called the order of integration, denoted by d ?

?

How many times should a series be differenced?

In practice, it is not required to go beyond second difference.

Case Study

Background

- Annual Sales for a specific company from year 1961 to 2017

Objective

- To develop time series model and forecast sales for next 3 years

Available Information

- Number of cases: 57
- Variables: Year, sales(in 10's GBP)

Data Snapshot

turnover_annual data

Variables

ons on Discrete Time Scale

Year	sales
1961	224786
1962	230034
1963	236562
1964	250960
1965	261615
1966	268316
1967	283589
1968	280160
1969	301422
1970	308018
1974	364834
1975	392503

Columns	Description	Type	Measurement	Possible values
Year	Financial Year	Numeric	-	-
sales	sales(in 10's GBP)	Numeric	In British Pound	Positive values

Creating and Plotting Time Series in R

```
#Importing the Data
turnover_annual <- read.csv("turnover_annual.csv",header=TRUE)
#Creating and Plotting a Time Series Object
salesseries<-ts(turnover_annual$sales,start=1961,end=2017)
plot(salesseries,col="red",main="Sales Time Series (Simple Plot)")
```

ts() converts a column from a data frame to time series object. The **start=** and **end=** arguments specify start and end period

Interpretation :

- The time-series clearly shows a positive trend.

Dickey Fuller Test For Checking Stationarity

```
# Install & load package "urca" for executing Dickey Fuller Test
install.packages("urca")
library(urca)
df<-ur.df(salesseries, lag=0)
summary(df)
```

ur.df() performs a Dickey Fuller unit root test on time series data.

```
# Output
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1)

Residuals:
   Min     1Q Median     3Q    Max
-73514 -19075  -8490   3412  76065

Coefficients:
        Estimate Std. Error t value Pr(>|t|)
z.lag.1 0.064334   0.003338   19.27   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25950 on 55 degrees of freedom
Multiple R-squared:  0.871,     Adjusted R-squared:  0.8687
F-statistic: 371.5 on 1 and 55 DF,  p-value: < 2.2e-16


Value of test-statistic is: 19.2745

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.6 -1.95 -1.61
```

Interpretation :

- Time series is non-stationary as value of test statistic is greater than 5% critical value.

Determining Order of Differencing

```
# Install & load package “forecast”
install.packages("forecast")
library(forecast)
ndiffs(salesseries)
[1] 2

salesdiff2<-diff(salesseries,differences=2)
plot(salesdiff2,col="red")
```

- **ndiffs()** by default uses a Unit Root test to estimate the number of differences required.

Interpretation :

- After 2nd order differencing, the series looks **stationary**.

Correlogram for 2nd Order Differenced Time Series

```
# ACF Plot
acf(salesdiff2,col="blue")
# Output
```

Interpretation :

- Stationarity is achieved with 2nd order difference.

Step 2: Model Identification

Model Identification

- When the data are confirmed stationary, proceed to tentative identification of models through **visual inspection of correlogram and partial correlogram**

Model	AC	PAC
AR (p) $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t$	Dies down	Cuts off after lag p
MA (q) $y_t = \delta + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}$	Cuts off after lag q	Dies down
ARMA (p,q) $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}$	Dies down	Dies down

Model Identification

- ARIMA model is expressed as arima (p,d,q) where
  - p = no. of autoregressive terms
  - d = order of differencing
  - q = no. of moving average terms

ACF Plot

PACF Plot

- ACF and PACF correlograms will help in determining the MA and AR values respectively.

<u>Indicative Model :</u>
<u>arima(2,2,2)</u>

Step 3: Parameter Estimation

Parameter Estimation in R

```
#Estimation using arima function
salesmodel<-arima(salesseries,order=c(2,2,2))
```

- **arima()** fits a model to a univariate time series.
- **order=** argument specifies values of p,d and q

```
coef(salesmodel)
AIC(salesmodel)
```

coef() and **AIC()** return the model coefficients and AIC value.

```
# Output
> coef(gdpmodel)
       ar1        ar2        ma1        ma2
-1.2355735 -0.6701038  0.7851143  0.3295140
> AIC(gdpmodel)
[1] 1285.984
```

There are two AR coefficients and two MA coefficients.
Smaller the AIC value, better is the model. We need to try out various combinations of AR and MA terms to arrive at final model.

Automatic Estimation of Model Parameters

```
# Install & load package “forecast”
# Automatic Model Identification and Parameter Estimation
```

```
install.packages("forecast")
library(forecast)

auto.arima(salesseries,d=2,max.p=2,max.q=2,trace=TRUE,ic="aic")
```

- **auto.arima()** generates the best order arima model. The function conducts a search over possible models within the order constraints provided.
- **trace=** TRUE returns the list of all models considered.
- **ic=** specifies the information criterion. We have specified it as **“aic”.**

Automatic Estimation of Model Parameters

```
# Output
 ARIMA(2,2,2)                    : 1285.984
 ARIMA(0,2,0)                    : 1294.497
 ARIMA(1,2,0)                    : 1283.644
 ARIMA(0,2,1)                    : 1285.212
 ARIMA(2,2,0)                    : 1285.114
 ARIMA(1,2,1)                    : 1285.361
 ARIMA(2,2,1)                    : 1283.568
 ARIMA(1,2,2)                    : 1286.479

 Best model: ARIMA(2,2,1)

Series: gdpseries
ARIMA(2,2,1)

Coefficients:
          ar1      ar2     ma1
      -1.3949  -0.5100  0.9727
s.e.   0.1171   0.1168  0.1019

sigma^2 estimated as 714703483:  log likelihood=-637.78
AIC=1283.57   AICc=1284.37   BIC=1291.6
```

Interpretation :

- Model with the lowest AIC value is selected as the best model.

arima Function Using BEST Order

```
# Run arima() for cross checking parameters based on model suggested
# by auto.arima
```

```
salesmodel<-arima(salesseries,order=c(2,2,1))
coef(salesmodel)
AIC(salesmodel)
```

```
# Output
```

```
> coef(gdpmodel)
       ar1        ar2        ma1
-1.3948728 -0.5100167  0.9726749
> AIC(gdpmodel)
[1] 1283.568
```

Model Selection Criteria

Akaike Information Criterion (AIC)

$$AIC = -2 \ln(L) + 2k$$

where L = Likelihood function

k = Number of parameters to be estimated

<u>Ideally, AIC should be as small as possible</u>

Step 4: Diagnostic Checking

Residual Analysis

If an ARMA(p,q) model is an adequate representation of the data generating process then the residuals should be 'White Noise'

- White Noise time series has zero mean, constant variance and zero covariance with lagged time series.
- Box-Pierce Test (Q Statistic) is the most recommended method for checking if the residuals are white noise process.
- Ljung-Box test is also used for the same purpose.

Box Pierce Test

Objective	To test the **null hypothesis** that **e_t is a white noise process**

Test Statistic	$$Q_{BP} = T\sum_{\tau=1}^{m}\hat{\rho}^2(\tau) \sim \chi^2(m)$$ for large T (based on autocorrelations upto lag m and T observations in a time series)
Decision Criteria	Reject the null hypothesis **if p-value < 0.05**

Box Pierce Test in R

```
# Box Test
resi<-residuals(salesmodel)
Box.test(resi)
plot(resi,col="red")
```

residuals() calculates residual values.

- **Box.test()** runs can run a variety of diagnostic tests. It computes the Box Pierce statistic by default.
- **Box.test()** also has the provision for calculating Ljung–Box test statistic, by adding the **type = "Ljung-Box"** argument.

```
# Output
```

```
        Box-Pierce test

data:  resi
X-squared = 0.17544, df = 1, p-value = 0.6753
```

Interpretation :

- Do not reject Ho, as p-value is greater than 0.05.
- Errors follow white noise process.

Step 5: Forecasting

Forecasting

```
# Forecast for next 3years
predict(salesmodel,n.ahead=3)

# Output
```

predict() function is used to generate next 3 years sales forecasts

```
$pred
Time Series:
Start = 2018
End = 2020
Frequency = 1
[1] 3072045 3308799 3537673

$se
Time Series:
Start = 2018
End = 2020
Frequency = 1
[1] 26017.21 48562.55 75729.34
```

Next 3 years sales forecasts

Quick Recap

Stationarity Checking	• Use **`ndiffs()`** to determine order of differencing • Plot correlogram using **`acf()`** and validate stationarity using **`ur.df()`**
Model Identification	• Tentative identification of models through visual inspection of correlogram and partial correlogram
Parameter Estimation	• **`auto.arima()`** is recommended for obtaining best ARIMA model • It uses AIC as the model selection criteria
Diagnostic Checking	• **`Box.test()`** performs a Box-Pierce test for checking whether errors follow white noise process
Forecasting	• Use **`predict()`** to generate forecasts

Time Series Modeling

Seasonal ARIMA Model

Contents

1. Seasonal Box-Jenkins (ARIMA) Models
2. Five Step Iterative Procedure
   i. Stationarity Checking and Seasonal Differencing
      - Differencing, Correlograms, and Dickey Fuller Test in R
   ii. Model Identification
   iii. Parameter Estimation
      - Simple and Automated Model Estimation in R
      - Running ARIMA in R
   iv. Diagnostic Checking
      - Box Pierce Test in R
   v. Forecasting
      - Predictions on ARIMA Model in R

Seasonal Box-Jenkins (ARIMA) Models

- ARIMA (Auto Regressive Integrated Moving Average) models are Regression models that use lagged values of the dependent variable and/or random disturbance term as explanatory variables.

- Seasonal ARIMA (Often abbreviated as SARIMA) Model is formed by including seasonal terms in the ARIMA model.

- Several real world time series have a seasonal component. Some examples are: Sales of woolen clothes, demand for fertilizers, electricity consumption, etc.

Seasonal Box-Jenkins (ARIMA) Models

- The **seasonal ARIMA model** incorporates both **non-seasonal and seasonal** factors in a multiplicative model.

- Shorthand notation for the model is,

$$\text{ARIMA}\ (p, d, q) \times (P, D, Q)S,$$

with,
p = non-seasonal AR order,
d = non-seasonal differencing,
q = non-seasonal MA order,
P = seasonal AR order,
D = seasonal differencing,
Q = seasonal MA order, and
S = time span of repeating seasonal pattern.

Five-Step Iterative Procedure

Step 1: Stationarity Checking

Assessing Stationarity of Time Series

- Stationarity of a time series can be assessed using:

Time Series Plot (Time v/s Variable) OR Correlogram OR Dickey-Fuller Test

- If a time series is non-stationary then it can be converted via

Differencing De-trending

Seasonal Differencing

- Seasonal differencing is denoted as ,

$$\Delta_s y_t = y_t - y_{t-s}$$

Where,

s denotes **frequency of season**

s = 12 if data is monthly; s = 4 if data is quarterly and so on

- First and seasonal span differencing for monthly data is,

$$\Delta_1 \Delta_s y_t = \Delta_1 (y_t - y_{t-s}) = y_t - y_{t-1} - y_{t-s} + y_{t-s-1}$$

Case Study

Background

- Sales Data for 3 Years (2013, 2014, 2015)

Objective

- To develop seasonal ARIMA Model for generating forecasts

Available Information

- Sample size is 36
- Variables: Year, Month, Sales

Data Snapshot

Sales Data for 3 Years

Variables

Monthly Observations

Year	Month	Sales
2013	Jan	123
2013	Feb	142
2013	Mar	164
2013	Apr	173
2013	May	183
2013	Jun	192
2013	Jul	199
2013	Aug	203
2013	Sep	207
2013	Oct	209
2013	Nov	214
2014	Jul	245

Columns	Description	Type	Measurement	Possible values
Year	Year	nemeric	2013, 2014, 2015	3
Month	Month	factor	Jan - Dec	12
Sales	Sales in USD Million	numeric	USD Million	Positive values

Plotting a Time Series in R

```
# Importing the Data
salesdata<-read.csv("Sales Data for 3 Years.csv",header=TRUE)

#Creating and Plotting a Time Series Object
salesseries<-ts(salesdata$Sales,start=c(2013,1),end=c(2015,12),
                frequency=12)

plot(salesseries,col="red",main="Sales Time Series (Simple Plot)")
```

```
# Output
```

Interpretation :

- The time series shows periodic peaks, indicative of seasonality.

Correlogram

```
acf(salesseries,col="blue")
```

```
# Output
```

Interpretation :

- ACF plot shows a slow decay indicating non-stationarity.

Determining the Order of Differencing and Dickey Fuller Test For Original Series

```
# Better Way of Determining the Order of Differencing
```

```
ndiffs(salesseries)
[1] 1
```

```
# Dickey Fuller Test
```

```
df<-ur.df(salesseries, lag=0)
summary(df)
```

```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-49.216  -4.668  -1.959   3.965  49.363

Coefficients:
        Estimate Std. Error t value Pr(>|t|)
z.lag.1  0.02065    0.01274   1.621    0.114

Residual standard error: 17.82 on 34 degrees of freedom
Multiple R-squared:  0.07176,	Adjusted R-squared:  0.04446
F-statistic: 2.628 on 1 and 34 DF,  p-value: 0.1142
```

Interpretation :

- Time series is non-stationary. Value of test statistic is greater than 5% critical value.

* nsdiffs() function in forecast package uses seasonal unit root tests to determine the number of seasonal differences required for time series x to be made stationary. However, in most cases seasonal differencing of the first order is enough and hence not much focus is put on checking order of differencing.

Dickey Fuller Test – Differenced Series

```
# Dickey Fuller Test for Difference Series
salesdiff <- diff(salesseries,differences = 1)
summary(ur.df(salesdiff,lags = 0))
```

```
# Output
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression none


Call:
lm(formula = z.diff ~ z.lag.1 - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-26.753   1.097   2.598   7.343  55.287

Coefficients:
        Estimate Std. Error t value Pr(>|t|)
z.lag.1  -1.3186     0.1913  -6.891 7.18e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.75 on 33 degrees of freedom
Multiple R-squared:   0.59,     Adjusted R-squared:  0.5776
F-statistic: 47.49 on 1 and 33 DF,  p-value: 7.184e-08


Value of test-statistic is: -6.8914

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.62 -1.95 -1.61
```

Interpretation :

- Time series is stationary. Value of test statistic is less than 5% critical value.

Step 2: Model Identification

Model Identification

- When the data are confirmed stationary, proceed to tentative identification of models through **visual inspection of correlogram and partial correlogram**

Model	AC	PAC
AR (p) $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t$	Dies down	Cuts off after lag p
MA (q) $y_t = \delta + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}$	Cuts off after lag q	Dies down
ARMA (p,q) $y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}$	Dies down	Dies down

Model Identification

- Seasonal ARIMA model is expressed as arima(p,d,q) (P,D,Q) where
  - p = no. of autoregressive terms
  - d = order of differencing
  - q = no. of moving average terms
  - (P,D,Q) are seasonal equivalents of autoregressive, difference and moving average terms

ACF Plot

PACF Plot

Indicative Model : arima(2,1,2)(2,1,2)

Step 3: Parameter Estimation

Parameter Estimation

- There are two ways in which parameters of arima models can be estimated

1. Ordinary Least Squares
2. Maximum Likelihood Method – when the model involves MA component

- Given n observations $y_1, y_2, \ldots, y_n$, the likelihood function L is defined as - the probability of obtaining the data actually observed
- The maximum likelihood estimators (MLE) are those values of the parameters for which the data actually observed are most likely, that is, the values that maximize the likelihood function L.

Parameter Estimation in R

```
# Install and load package “forecast”
# Automatic Model Identification and Parameter Estimation
```

```r
install.packages("forecast")
library(forecast)

auto.arima(salesseries,d=1,D=1,max.p=2,max.q=2,max.P=2,max.Q=2,
           trace=TRUE,ic="aic")
```

- **auto.arima()** generates the best order arima model. The function conducts a search over possible model within the order constraints provided.
- Seasonal model requires **max.D,max.P** and **max.Q** arguments as well.
- **trace=** TRUE returns the list of all models considered.
- **ic=** specifies the information criterion. We have specified it as **“aic”.**

Automatic Model Identification

Output

```
> auto.arima(salesseries,d=1,D=1,max.p=2,max.q=2,max.P=2,max.Q=2,
+            trace=TRUE,ic="aic")

 ARIMA(2,1,2)(1,1,1)[12]                     : 157.0397
 ARIMA(0,1,0)(0,1,0)[12]                     : 156.1096
 ARIMA(1,1,0)(1,1,0)[12]                     : 159.1271
 ARIMA(0,1,1)(0,1,1)[12]                     : 159.4057
 ARIMA(0,1,0)(1,1,0)[12]                     : 157.6536
 ARIMA(0,1,0)(0,1,1)[12]                     : 157.6536
 ARIMA(0,1,0)(1,1,1)[12]                     : 159.6536
 ARIMA(1,1,0)(0,1,0)[12]                     : 157.1806
 ARIMA(0,1,1)(0,1,0)[12]                     : 157.6069
 ARIMA(1,1,1)(0,1,0)[12]                     : 154.6016
 ARIMA(1,1,1)(1,1,0)[12]                     : 161.3531
 ARIMA(1,1,1)(0,1,1)[12]                     : 156.6
 ARIMA(1,1,1)(1,1,1)[12]                     : 163.329
 ARIMA(2,1,1)(0,1,0)[12]                     : 152.5563
 ARIMA(2,1,1)(1,1,0)[12]                     : 153.1015
 ARIMA(2,1,1)(0,1,1)[12]                     : Inf
 ARIMA(2,1,1)(1,1,1)[12]                     : 155.0459
 ARIMA(2,1,0)(0,1,0)[12]                     : 151.6156
 ARIMA(2,1,0)(1,1,0)[12]                     : 151.8961
 ARIMA(2,1,0)(0,1,1)[12]                     : Inf
 ARIMA(2,1,0)(1,1,1)[12]                     : 153.864

 Best model: ARIMA(2,1,0)(0,1,0)[12]

Series: salesseries
ARIMA(2,1,0)(0,1,0)[12]

Coefficients:
         ar1     ar2
      0.1583  0.6353
s.e.  0.1545  0.1856

sigma^2 estimated as 34.14:  log likelihood=-72.81
AIC=151.62   AICc=152.88   BIC=155.02
```

Interpretation :

- Model with the lowest AIC value is selected as the best model.

Using BEST order in arima Function

```
# Obtaining Coefficient
salesmodel<-arima(salesseries,order=c(2,1,0),
seasonal=list(order=c(0,1,0),period=12))

coef(salesmodel)
ar1       ar2
0.1583489 0.6352769
```

Model Selection Criteria

- **Akaike Information Criterion (AIC)**

$$\boxed{AIC = -2 \ln(L) + 2k}$$

where L = Likelihood function

k = Number of parameters to be estimated

<u>Ideally, AIC should be as small as possible</u>

Step 4: Diagnostic Checking

Residual Analysis

If an ARMA(p,q) model is an adequate representation of the data generating process then the residuals should be 'White Noise'

- White Noise time series has **zero mean, constant variance and zero covariance with lagged time series.**

- **Box-Pierce Test (Q Statistic) is the most recommended method for checking if the residuals are white noise process.**

- Ljung-Box test is also used for the same purpose.

Box Pierce Test

Objective	To test the **null hypothesis** that e_t **is a white noise process**

Test Statistic	$$Q_{BP} = T\sum_{\tau=1}^{m}\hat{\rho}^2(\tau) \sim \chi^2(m)$$ for large T (based on autocorrelations upto lag m and T observations in a time series)
Decision Criteria	Reject the null hypothesis **if p-value < 0.05**

Box-Pierce Test in R

```
# Box Test
resi<-residuals(salesmodel)
Box.test(resi)
```

```
# Output
        Box-Pierce test

data:  resi
X-squared = 0.78552, df = 1, p-value = 0.3755
```

```
plot(resi,col="red")
```

Interpretation :

- Do not reject Ho, as p-value is greater than 0.05.
- Errors follow white noise process.

Step 5: Forecasting

Forecasting

```
# Forecast for next 3years
predict(salesmodel,n.ahead=3)
```

predict() function is used to forecast sales for next 3 periods

```
# Output
> predict(salesmodel,n.ahead=3)
$pred
          Jan      Feb      Mar
2016 285.6334 292.1748 292.0954

$se
           Jan       Feb       Mar
2016  5.582376  8.542627 13.268502
```

Next 3 period sales forecasts

Quick Recap

Stationarity Checking	• Use **`ndiffs()`** to determine order of differencing • Plot correlogram using **`acf()`** and validate stationarity using **`ur.df()`**
Model Identification	• Tentative identification of models through visual inspection of correlogram and partial correlogram
Parameter Estimation	• **`auto.arima()`** is recommended for obtaining best ARIMA model • It uses AIC as the model selection criteria
Diagnostic Checking	• **`Box.test()`** performs a Box-Pierce test for checking whether errors follow white noise process
Forecasting	• Use **`predict()`** to generate forecasts

Time Series Regression

Multiple Linear Regression Analysis on Time Series Data

Contents

1. Linear Regression Model for Time Series Data
2. The Concept of Autocorrelation
3. Autocorrelation in Multiple Linear Regression
4. Detecting Presence of Autocorrelation
   i. Durbin Watson Test
   ii. Durbin Watson Test in R
5. Remedial Measure: Maximum Likelihood Estimation Method

Linear Regression Model for Time Series Data

$$Y_t = b_0 + b_1X_{1t} + b_2X_{2t} + \ldots + b_pX_{pt} + e_t$$

Where,

Y_t	:	Dependent Variable at time t
$X_{1t}, X_{2t}, \ldots, X_{pt}$	:	Independent Variables at time t
$b_0, b_1, \ldots, b_p$	:	Parameters of Model
e_t	:	Random Error Component at time t

Variables can either be **Continuous or Categorical**

The Concept of Autocorrelation

- Autocorrelation is a **correlation between a time series** (Y_t) and another time series representing lagged values of the same time series(Y_{t-k}).

- Autocorrelation refers to the **correlation of a time series with its own past values**

t	Y_t
1	Y_1
2	Y_2
"	"
"	"
10	Y_{10}

t	Y_t	Y_{t-1}
2	Y_2	Y_1
3	Y_3	Y_2
"	"	"
"	"	"
10	Y_{10}	Y_9

- Correlation between Y_t and Y_{t-1} is **"first order autocorrelation"**

Autocorrelation in Multiple Linear Regression

- Autocorrelation is observed only when the data is a time series or panel data
- In regression model, absence **of autocorrelation among errors is one of the key assumptions**

Autocorrelation → **Incorrect Standard Errors of Parameters**

If we obtain the OLS estimators of the regression model without correcting error autocorrelation, the model may yield

- **Underestimation of True Error Variance**
- **Overestimation of R^2**
- **Misleading Results of t & F Tests**

Case Study – Predicting Sales

Background

- A retail store undertakes two-fold marketing campaign, one for print media and the other for digital. The company also collects information on yearly increments (price adjusted) in sales. The company wishes to check if the marketing expenses have any bearing on the sales

Objective

- To predict price adjusted incremental sales based on expenses on marketing campaigns

Available Information

- Yearly time series data, of 11 years
- Independent Variables: Marketing expenses for – Print Media and Online
- Dependent Variable: Price Adjusted Incremental Sales

Data Snapshot

SALES VS MARKETING COSTS

Dependent Variable | **Independent Variables**

year	sales	print	online
2000	65	20	30
2001	62	27	23
2002	70	28	34

Columns0	Description	Type	Measurement	Possible values
Year	Year	Numeric	2000 to 2010	11
Sales	Price Adjusted Incremental Sales	Numeric	in Euro Million	Positive values
Print	Expenses on Print Marketing	Numeric	in Euro Million	Positive values
online	Expenses on Online Marketing	Numeric	in Euro Million	Positive values

Detecting Presence of Autocorrelation – Durbin Watson Test

Objective	To **check the assumption of 'No Autocorrelation'**

Null Hypothesis H_0: Autocorrelation is not present among errors

Alternate Hypothesis H_1: Not H0

Test Statistic	$d \cong 2(1-r)$ Where r is sample autocorrelation based on residuals obtained in regression model. Ideal Value of d = 2 (taking r = 0)
Decision Criteria	Reject the null hypothesis if p-value < 0.05

Durbin Watson Test in R

```
# Importing the Data and Fitting Linear Model
sales<-read.csv("SALES VS MARKETING COSTS.csv",header=TRUE)
salesmodel<-lm(sales~print+online,data=sales)

summary(salesmodel)
```

```
# Output
Call:
lm(formula = sales ~ print + online, data = sales)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.16109 -1.00608  0.07342  0.85923  2.32128 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 30.05815    3.78520   7.941 4.61e-05 ***
print        0.48154    0.14275   3.373  0.00974 ** 
online       0.76633    0.06828  11.224 3.56e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.486 on 8 degrees of freedom
Multiple R-squared:  0.952,	Adjusted R-squared:   0.94 
F-statistic:  79.3 on 2 and 8 DF,  p-value: 5.317e-06
```

Durbin Watson Test in R

```
# Install & Load package "car"
# Durbin Watson Test
```

```r
install.packages("car")
library(car)

durbinWatsonTest(salesmodel)
```

durbinWatsonTest() in package car carries out the Durbin Watson test. Output gives the d-w statistic as well as p-value.

```
# Output
```

```
> durbinWatsonTest(salesmodel)
 lag Autocorrelation D-W Statistic p-value
   1       0.4804249     0.7259509   0.004
 Alternative hypothesis: rho != 0
```

Interpretation :

- Reject H0, p-value<0.05.
- Errors have significant autocorrelation.

Error Process in the Presence of Autocorrelation

Presence of autocorrelation implies that, errors at time t are related to errors at previous time points

$$\boxed{e_t = \rho_1 e_{t-1} + \rho_2 e_{t-2} + \ldots + \rho_k e_{t-k} + u_t}$$

ρ_k is coefficient of autocorrelation of lag k between e_t & e_{t-k}

u_t is such that,

E(u_t)=0

V(u_t)= Constant

Cov(u_t , u_s)=0 s≠0

* Note: Typically in time series regression we consider autocorrelation of order 1 only

Consequences of Ignoring Autocorrelation

If we obtain the OLS estimators of the model, without correcting error autocorrelation, the model may yield

- Underestimation of True Error Variance
- Overestimation of R^2
- Misleading Results of t & F Tests

Autocorrelation – Remedial Measures

Cochrane-Orcutt Method

Praise-Winsten Method

Maximum Likelihood Estimation Method

Maximum Likelihood Estimation Method

- Maximum likelihood estimator (MLE) can be obtained by maximizing the log likelihood function with respect to b, σ^2_u, ρ

- Log likelihood function is given as follows:

$$\ln L = -\frac{\sum_{t=1}^{T} u_t^{\,2}}{2\sigma_u^2} + \frac{1}{2}\ln(1-\rho^2) - \frac{T}{2}\left(\ln 2\pi + \ln \sigma_u^2\right)$$

Parameter Estimation – Maximum Likelihood Estimation Method

```
# Parameter Estimation (Maximum Likelihood Estimation)
salests<-ts(sales$sales,start=2000,end=2010)
```

ts() converts a column from a data frame to a simple time series object.

```
xvar<-subset(sales,select=c("online","print"))
```

subset() creates a subset of all the independent variables

```
salesmodel<-arima(salests,order=c(1,0,0),xreg=xvar)
```

arima() estimates model parameters. Subset of independent variables (which is a vector or matrix of external regressors) is used in **xreg=**

```
coef(salesmodel)
 ar1  intercept     online      print
 0.7774653 32.0288203  0.8117825  0.3621399
```

Parameter Estimation – Maximum Likelihood Estimation Method

```
#Parameter Estimation (Maximum Likelihood Estimation)
install.packages("lmtest")
library(lmtest)

coeftest(salesmodel)
```

arima() does not give p-values, hence **coeftest()** from package **lmtest()** is used. It performs z and (quasi-)t tests of estimated coefficients.

```
# Output
z test of coefficients:

           Estimate Std. Error z value  Pr(>|z|)
ar1        0.777465   0.171698  4.5281 5.952e-06 ***
intercept 32.028820   2.137725 14.9827 < 2.2e-16 ***
online     0.811782   0.033897 23.9486 < 2.2e-16 ***
print      0.362140   0.065133  5.5600 2.697e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Interpretation :

- Both print and online are significant variables in this output as well

Predictions

```
# Predictions (For Next Two Years)
online<-c(40,45)

print<-c(50,55)

xvarnew<-data.frame(online,print)



predict(salesmodel, n.ahead=2, newxreg=xvarnew)
```

Expected Values of online and print marketing expenses in next 2 years.

data.frame() combines the future expenses into a single dataframe object, to be used for predictions.

predict() uses the MLE regression model, **n.ahead=2** ensures next two forecasts are generated, **newxreg=** specifies the new values of independent variables.

```
# Output
$pred
Time Series:
Start = 2011
End = 2012
Frequency = 1
[1] 82.80867 88.63343

$se
Time Series:
Start = 2011
End = 2012
Frequency = 1
[1] 0.8509076 1.0778190
```

Quick Recap

Statistical Model	• $Y_t = b_0 + b_1 x_{1t} + b_2 x_{2t} + - - - - + b_p x_{pt} + e_t$ • Here, t – the time element is added to each term
What is Autocorrelation	• Autocorrelation refers to the correlation of a time series with its own past values
Consequences	• Presence of Autocorrelation results in incorrect standard errors of model parameters
Test	• The Durbin Watson test is used to check autocorrelation • **`durbinwatsonTest()`** in package **car** performs the D-W Test in R
Maximum Likelihood Estimation Method	• Method of correcting autocorrelation problem

Time Series Analysis – Exponential Smoothing Methods for Forecasting

DATA SCIENCE INSTITUTE

Contents

1. Forecasting Using Smoothing Methods
2. Exponential Smoothing in R
   i. Single Exponential Smoothing
   ii. Double Exponential Smoothing
   iii. Triple Exponential Smoothing
3. Exploring Built-In Time Series Data in R

DATA SCIENCE INSTITUTE

Forecasting Using Smoothing Methods

- Random, unexplained variation in a time series can have an undesirable impact on forecasts
- Smoothing can cancel or reduce such impacts
- Smoothing can either be Simple (using Moving Averages) or Exponential

DATA SCIENCE INSTITUTE

Single Exponential Smoothing Model

Mathematical Model :

$$F_{t+1} = \alpha Y_t + (1 - \alpha) F_t$$

Where,

F_{t+1}	:	Forecast value for period t + 1
F_t	:	Forecast value for period t
Y_t	:	Actual value for period t
α	:	Alpha (Smoothing constant)

* Note that here we use the dataset "Sales Data for 3 Years"

DATA SCIENCE INSTITUTE

Single Exponential Smoothing Model

Assume α=0.8

t	yt	Ft	
1	23	-	
2	24	23	
3	26	23.80	=0.8*24+0.2*23
4	23.5	25.56	=0.8*26+0.2*23.8
5	27	23.91	
6	26.1	26.38	
7	28	26.16	
8	27	27.63	
9	29	27.13	
10	29.3	28.63	
11	28.2	29.17	
12	27	28.39	
		27.28	

* 1st future value will always be the previous value & then for rest future values exponential smoothing mathematical formula is applied.

DATA SCIENCE INSTITUTE

Single Exponential Smoothing Model - Smoothing Constant α

Values of α

close to one ✉ have less of a smoothing effect and give greater weight to recent changes in the data

closer to zero ✉ have a greater smoothing effect and are less responsive to recent changes

- There is no formally correct procedure for choosing α. Sometimes the statistician's judgment is used to choose an appropriate factor.

- Alternatively, α can be decided based on statistical measure such as Root Mean Squared Error.

DATA SCIENCE INSTITUTE

Get an Edge!

Why the Name "Exponential"?

- This method gives weights to past observation in exponentially decreasing manner.

$$F_{t+1} = \alpha y_t + \alpha(1-\alpha) y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \alpha(1-\alpha)^3 y_{t-3} \text{-----}$$

$$= \alpha y_t + (1-\alpha)\,[\alpha y_{t-1} + \alpha(1-\alpha) y_{t-2} + \alpha(1-\alpha)^2 y_{t-3} \text{----}]$$

$$= \alpha y_t + (1-\alpha) F_t$$

- Larger alpha gives more weight to recent values.

DATA SCIENCE INSTITUTE

Exponential Smoothing in R

```
#Importing the Data
salesseries<-ts(salesdata$Sales,start=c(2013,1), end=c(2015,12),
frequency=12)
```

```
#Single Exponential Smoothing
fit1<-HoltWinters(salesseries, beta=FALSE, gamma=FALSE)
```

```
predict(fit1,n.ahe
fit1
```

- **HoltWinters()** undertakes exponential smoothing.
- **beta = FALSE** and **gamma = FALSE** ensures single exponential smoothing is performed.

* Note that here we use the dataset "Sales Data for 3 Years"

Exponential Smoothing in R

```
# Output
> predict(fit1,n.ahead=1)
          Jan
2016 314.4432
> fit1
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = salesseries, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.754789
 beta : FALSE
 gamma: FALSE

Coefficients:
      [,1]
a 314.4432
```

Interpretation :
It returns predicted future value & value of alpha.

DATA SCIENCE INSTITUTE

Double (Holt) and Triple(Holt-Winters) Exponential Smoothing Methods

Double exponential smoothing has two equations

First equation is similar to single exponential smoothing method

Second equation updates trend using constant beta.

Double exponential smoothing method is used when there is a trend in the time series.

Triple exponential smoothing has three equations

First 2 equations are similar to double exponential smoothing method

Third equation updates seasonal component using constant gamma.

Triple exponential smoothing method is used when there is trend + seasonality in the time series.

DATA SCIENCE INSTITUTE

Double Exponential Smoothing Model

Mathematical Model :

Where,

F_{t+1}	:	Forecast value for period t +1
F_t	:	Forecast value for period t
T_t	:	Trend component for period t
T_{t+1}	:	Trend component for period t +1
Y_t	:	Actual value for period t
α	:	Alpha (Smoothing constant)
β	:	Beta (Second smoothing constant)

DATA SCIENCE INSTITUTE

Double Exponential Smoothing in R

```
#Double Exponential Smoothing
fit2<-HoltWinters(salesseries, gamma=FALSE)
predict(fit2,n.ahead=1)
fit2
```

gamma = FALSE ensures double exponential smoothing

```
# Output
> predict(fit2,n.ahead=1)
          Jan
2016 306.1702
> fit2
Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
HoltWinters(x = salesseries, gamma = FALSE)

Smoothing parameters:
 alpha: 0.3835632
 beta : 0.4889297
 gamma: FALSE

Coefficients:
       [,1]
a 294.81648
b  11.35368
```

Interpretation :

It returns predicted future value, value of alpha

DATA SCIENCE INSTITUTE

Triple Exponential Smoothing Model

Mathematical Model :

$$F_{t+1} = \alpha \frac{Y_t}{S_{t+1-k}} + (1-\alpha) F_t - T_t$$

where,

S_{t+1-k}	:	Seasonal smoothing value for period t +1
F_t	:	Forecast value for period t
F_{t+1}	:	Forecast value for period t +1
F_t	:	Forecast value for period t
T_t	:	Trend component for period t
T_{t+1}	:	Trend component for period t +1
Y_t	:	Actual value for period t

α : Alpha (Smoothing constant) β : Beta (Second smoothing constant)
: Gamma (Third smoothing constant)

DATA SCIENCE INSTITUTE

Triple Exponential Smoothing in R

```
#Triple Exponential Smoothing
```

```r
fit3<-HoltWinters(salesseries)
predict(fit3,n.ahead=1)
fit3
```

```
# Output
```

```
> predict(fit3,n.ahead=1)
          Jan
2016 295.9492
> fit3
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = salesseries)

Smoothing parameters:
 alpha: 0.911556
 beta : 0
 gamma: 0.8681419

Coefficients:
            [,1]
a    296.725314337
b      4.522435897
s1    -5.298595983
s2    -1.290510241
s3    -3.658258513
s4    -0.005594377
s5    -3.580920942
s6    -2.015183167
s7     2.131638936
s8    -0.258654397
s9    -3.287793819
s10   -6.786550699
s11   -7.502574409
s12   31.125466834
```

Interpretation :

It returns predicted future value & value of

DATA SCIENCE INSTITUTE

Get an Edge!

Always approach time series analysis in a systematic manner

Analyse

- Analyse trend and seasonality present in the data

Decompose

- Decompose time series into its components

Forecast

- Forecast future values of the time series

- For vector time series, investigate connections between two or more time series with the aim of using values of some of the processes to predict those of the others. (Eg. Pairs trading in stock market)

DATA SCIENCE INSTITUTE

Exploring Built-In Time Series Data in R

```
#Fetching the Data
data("AirPassengers")

AirPassengers
```

AirPassengers() is Box-Jenkins data, with monthly totals of international airlines passengers (in thousands) from 1949 to 1960

```
# Output
```

```
> AirPassengers
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```


DATA SCIENCE INSTITUTE

Exploring Built-In Time Series Data in R

```
# Checking Data Features
class(AirPassengers)
[1] "ts"

start(AirPassengers)
[1] 1949 1

end(AirPassengers)
[1] 1960 12

frequency(AirPassengers)
[1] 12
```

class() returns class of the data. AirPassengers data is a time series.

start() and **end()** return the first and last values in the data.

frequency() returns information about frequency of time points in the data, i.e. whether monthly, daily, etc.

Interpretation :

- The data is of class ts.
- The database starts at year 1949, month 1 and ends at year 1960 and month 12.
- Frequency = 12 suggests that the data is monthly.

DATA SCIENCE INSTITUTE

Exploring Built-In Time Series Data in R

```
# Exploring Data
time(AirPassengers)
```

time() shows time at which the time series was sampled.

```
# Output
```

```
> time(AirPassengers)
         Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
1949 1949.000 1949.083 1949.167 1949.250 1949.333 1949.417 1949.500 1949.583 1949.667 1949.750 1949.833 1949.917
1950 1950.000 1950.083 1950.167 1950.250 1950.333 1950.417 1950.500 1950.583 1950.667 1950.750 1950.833 1950.917
1951 1951.000 1951.083 1951.167 1951.250 1951.333 1951.417 1951.500 1951.583 1951.667 1951.750 1951.833 1951.917
1952 1952.000 1952.083 1952.167 1952.250 1952.333 1952.417 1952.500 1952.583 1952.667 1952.750 1952.833 1952.917
1953 1953.000 1953.083 1953.167 1953.250 1953.333 1953.417 1953.500 1953.583 1953.667 1953.750 1953.833 1953.917
1954 1954.000 1954.083 1954.167 1954.250 1954.333 1954.417 1954.500 1954.583 1954.667 1954.750 1954.833 1954.917
1955 1955.000 1955.083 1955.167 1955.250 1955.333 1955.417 1955.500 1955.583 1955.667 1955.750 1955.833 1955.917
1956 1956.000 1956.083 1956.167 1956.250 1956.333 1956.417 1956.500 1956.583 1956.667 1956.750 1956.833 1956.917
1957 1957.000 1957.083 1957.167 1957.250 1957.333 1957.417 1957.500 1957.583 1957.667 1957.750 1957.833 1957.917
1958 1958.000 1958.083 1958.167 1958.250 1958.333 1958.417 1958.500 1958.583 1958.667 1958.750 1958.833 1958.917
1959 1959.000 1959.083 1959.167 1959.250 1959.333 1959.417 1959.500 1959.583 1959.667 1959.750 1959.833 1959.917
1960 1960.000 1960.083 1960.167 1960.250 1960.333 1960.417 1960.500 1960.583 1960.667 1960.750 1960.833 1960.917
```

Interpretation :

- The output shows time for each observation as year followed by time stamp.
- Time stamp for Jan it's 0/12, Feb it's 1/12,..... So on

DATA SCIENCE INSTITUTE

Exploring Built-In Time Series Data in R

```
cycle(AirPassengers)
```

cycle() shows positions of each observation in the cycle.

```
# Output
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	1	2	3	4	5	6	7	8	9	10	11	12
1950	1	2	3	4	5	6	7	8	9	10	11	12
1951	1	2	3	4	5	6	7	8	9	10	11	12
1952	1	2	3	4	5	6	7	8	9	10	11	12
1953	1	2	3	4	5	6	7	8	9	10	11	12
1954	1	2	3	4	5	6	7	8	9	10	11	12
1955	1	2	3	4	5	6	7	8	9	10	11	12
1956	1	2	3	4	5	6	7	8	9	10	11	12
1957	1	2	3	4	5	6	7	8	9	10	11	12
1958	1	2	3	4	5	6	7	8	9	10	11	12
1959	1	2	3	4	5	6	7	8	9	10	11	12
1960	1	2	3	4	5	6	7	8	9	10	11	12

Interpretation :

- The output shows frequency of data points, represented numerically

DATA SCIENCE INSTITUTE

Plotting Data with Trend Line

```
# Plotting Data with Trend Line
plot(AirPassengers)

model<-lm(AirPassengers~time(A

abline(model)
```

plot() returns a simple line plot of the entire data.

lm() fits a linear regression model on the data with observations as dependent variables and time stamps as independent variables.

abline() adds a straight line to the plot.

```
# Output
```

Interpretation :

- The plot shows a positive trend.

DATA SCIENCE INSTITUTE

Box Plot with Cycles

```
#Box Plot for Cycle
boxplot(AirPassengers~cycle(AirPassengers),col="red")
```

boxplot() generates a Box-plot for data cycles.

```
# Output
```

Interpretation :

- The plot gives a clear indication that number of passengers in the months of July and August were higher than the rest.

DATA SCIENCE INSTITUTE

Quick Recap

In this session, we learnt about **time series exponential smoothing**:

Smoothing	• Smoothing gives weights to past observations, in order to give more significance to seasonality and trend components of a time series.
Smoothing in R	• Use `HoltWinters()` to carry out exponential smoothing

DATA SCIENCE INSTITUTE

Time Series Analyisis –

The Concept of Co-integration using R

DATA SCIENCE
INSTITUTE

Contents

1. Concept of Co-integration
2. What is Pairs Trading
3. Co-integration and Pairs Trading
4. Engle-Granger Test
5. EG Test in R

DATA SCIENCE INSTITUTE

Concept of Co-integration

- The stock prices are believed to be **non-stationary** in nature.

- If there exists a linear combination of 2 stock prices , such that errors of the long term relationship are stationary (mean reverting) in nature, then the 2 stocks are said to be co-integrated.

- **Engle - Granger test** is used to identify the co-integrated pairs in a particular sector.

DATA SCIENCE INSTITUTE

Co-Integration

Examples of Co-integrated Series

- Income and Consumption
- Money, National Incomes, Price Rates, Interest Rates
- Price of a Commodity and Taxes Levied on that Commodity, Inflation Rate
- Temperatures, Electricity Consumption
- Prices of Two Stocks

DATA SCIENCE INSTITUTE

Co-Integration-Formal Definition

- **"Order of integration"** tells you the **minimum number of differences** needed to get a stationary series.
- **A series of successive differences, d,** can transform the time series into one with stationarity. The differences are denoted by **I(d)**, where **d is the order of integration**.

- Suppose Y_t and X_t are two time series integrated of order d, then
  - Any linear combination of such two series will also be integrated of order d (denoted as I(d)). This is called 'Integration'
- However,

  If there exists a vector β such that $u_t = Y_t - \beta X_t$ is of a lower order of integration (I(d – b), where b > 0) then Y_t and X_t are defined as Cointegrated of order (d,b)

- The idea of cointegration was introduced by **Engle and Granger** in 1987

* Usually, the order of integration is either I(0) or I(1); It's rare to see values for d that are 2 or more.

DATA SCIENCE INSTITUTE

What is Pairs Trading

- Pairs trading finds its roots in the area of securities trading
- A quest to generate returns, irrespective of market behavior, has led to the evolution of this strategy
- Certain securities, often competitors in the same sector, are associated in their day-to-day price movements. When the association breaks down, Traders take advantage of short term over pricing or under pricing of securities & invest accordingly, betting that the "spread" between the two would eventually converge
- Pairs Trading strategy works best in volatile market conditions

DATA SCIENCE INSTITUTE

Simple Example of Pairs Trading

Fig. 1: Pairs Trading

This is the right opportunity for Pairs Trade

TRADER WILL GAIN PROFIT IF HE SELLS HCLTECH AND BUYS GTL

DATA SCIENCE INSTITUTE

Co-Integration and Pairs Trading

- The **stock prices** are believed to be **non-stationary** in nature.
- **If there exists a linear combination of 2 stock prices ,** such that errors of the long term relationship are stationary (mean reverting) in nature, then the **2 stocks are said to be co-integrated.**
- Pairs trading is in fact an extended application of the theory of co-integration.
- **Engle -Granger test** is used to identify the co-integrated pairs in a particular sector.

DATA SCIENCE INSTITUTE

Engle Granger Test for Stock Prices

Let Y_t : TCS stock price at time t

Let X_t : IBM stock price at time t

Both the stock prices form non-stationary time series over time t.

If there exists a linear combination, such that the errors form stationary time series then TCS and IBM stocks are said to be co-integrated.

$$Y_t = \alpha + \beta X_t + \varepsilon_t$$

α = model intercept

β = regression coefficient

ε_t = error term

DATA SCIENCE INSTITUTE

Engle Granger Test for Stock Prices

Regression coefficient β is estimated using ordinary least square (OLS) method

If $\beta < 0$ then the pair is discarded

Selection of regression equation is done by comparing β

$$Y_t = \alpha_1 + \beta_1 X_t + \varepsilon_t \quad .....1$$

$$X_t = \alpha_2 + \beta_2 Y_t + \varepsilon_t \quad .....2$$

The equation with higher β is selected

In case of TCS and IBM stocks $\beta 1 > \beta 2$ hence equation 1 was selected

DATA SCIENCE INSTITUTE

Engle Granger Test

Engle Granger test is used to check stationarity of error time series.

Consider,

$$\Delta \varepsilon_t = \gamma^* \varepsilon_{t-1} + W_t$$

$\Delta \varepsilon_t = \varepsilon_t - \varepsilon_{t-1}$

ε_{t-1} = errors with lag 1

Objective	To test the **null hypothesis** that **time series is not stationary**

Null Hypothesis H0: Two stocks are not co-integrated

Alternate Hypothesis H1: Two stocks are co-integrated

Test Statistic	(y*/SE(y*)) Test statistic follows DF distribution under null
Decision Criteria	Reject the null hypothesis **if tcal < c(p)**

DATA SCIENCE INSTITUTE

Engle Granger Test in R

```
# Installing and Loading Required Packages
install.packages("quantmod")
install.packages("egcm")
library(quantmod)
library(egcm)
```

- **quantmod** is a very popular financial analysis package.
- **egcm** stands for Engle Granger Cointegration Models

```
#EG Test
getSymbols(c("tcs","ibm"), from =
                              env = .Gl
```

- **getSymbols()** fetches data from various sources. The command requires us to specify start and end dates of the prices required.
- **Symbols** a character vector specifying the names of each symbol to be loaded
- **env** where to create objects.

- **ecgm()** performs a simple two-step EG test on a pair of time series.
- egcm(X,Y) considers Y on the LHS and X on the RHS

DATA SCIENCE INSTITUTE

Engle Granger Test in R

```
# Output
IBM.Close[i] =   0.6258 TCS.Close[i] + 156.1917 + R[i], R[i] =   1.0000 R[i-1] + eps[i], eps ~ N(0,  1.6283^2)
               (0.8238)                (4.1724)                (0.0085)

R[2017-10-31] = -4.3906 (t = -0.364)

WARNING: TCS.Close and IBM.Close do not appear to be cointegrated.

Unit Root Tests of Residuals
                                                    Statistic    p-value
  Augmented Dickey Fuller (ADF)                        -0.952    0.90049
  Phillips-Perron (PP)                                 -2.319    0.91531
  Pantula, Gonzales-Farias and Fuller (PGFF)            0.988    0.80764
  Elliott, Rothenberg and Stock DF-GLS (ERSD)          -0.899    0.61297
  Johansen's Trace Test (JOT)                          -4.604    0.98910
  Schmidt and Phillips Rho (SPR)                       -2.290    0.93539

Variances
  SD(diff(TCS.Close))  =   0.200075
  SD(diff(IBM.Close))  =   1.634547
  SD(diff(residuals))  =   1.628260
  SD(residuals)        =  12.077787
  SD(innovations)      =   1.628260

Half life       = Infinite
R[last]         =  -4.390643 (t=-0.36)
```

Interpretation :

- Stocks TCS & IBM are not cointegrated.

?

What if we want to get data from other sources?

There are different packages using database APIs for data from other indices, eg. Quandl for India's National Stock Exchange (NSE) prices

DATA SCIENCE INSTITUTE

Get an Edge!

There are a number of alternative approaches for testing co-integration, apart from the Engle Granger test, such as

- CRDW (Cointegrating Regression Durbin Watson) test ▬ Simple regression of one variable on the other, and the standard Durbin-Watson test on the residuals.
- Error Correction Test ▬ The tendency of cointegrated variables to revert to common stochastic trends is expressed in terms of error-correction
- Johansen's Multivariate VAR Approach ▬ Examines the number of independent linear combinations (k) for an m time series variables set that yields a stationary process

DATA SCIENCE INSTITUTE

Quick Recap

In this session, we learnt about **co-integration in time series:**

Co-integration	• When the time series is integrated and linear combination of variables in that series is also integrated, but having an order lower than the whole series, then the variables are said to be co-integrated.
Pairs Trading	• When the association between two cointegrated stocks breaks down, Traders take advantage of short term over pricing or under pricing of securities & invest accordingly, betting that the "spread" between the two would eventually converge.
EG Test in R	• A combination of packages **quantmod** and **ecgm** is used to perform Engle Granger test in R, to check if a pair of time series is co-integrated.

DATA SCIENCE INSTITUTE