World Bank – ICAR funded
National Agricultural Higher Education Project
Centre for Advanced Agricultural Science and Technology (CAAST)
On
Genomics Assisted Crop Improvement and Management

Training Manual

Time Series Techniques for Forecasting in Agriculture

December 1 – December 10, 2021

Division of Agricultural Economics
ICAR – Indian Agricultural Research Institute
New Delhi – 110012
www.nahep-caast.iari.res.in

NAHEP Sponsored training programme
On
Time Series Techniques for Forecasting in Agriculture

Course Convenor

Alka Singh
Professor and Head
Division of Agricultural Economics
ICAR-Indian Agricultural Research Institute
Pusa Campus, Delhi – 110012
Email: asingh.eco@gmail.com

Co-Convenors

Girish K Jha
Principal Scientist
Division of Agricultural Economics
ICAR-Indian Agricultural Research Institute
New Delhi 110 012
E-mail: girish.stat@gmail.com

Nithyashree M L
Scientist
Division of Agricultural Economics
ICAR-Indian Agricultural Research Institute
New Delhi 110 012
E-mail: nithya.econ@gmail.com

Asha Devi S S
Scientist
Division of Agricultural Economics
ICAR-Indian Agricultural Research Institute
New Delhi 110 012
E-mail:ash.nibha@gmail.com

Division of Agriculture Economics
ICAR-Indian Agricultural Research Institute
New Delhi- 110 012

About NAHEP-CAAST at IARI, New Delhi

Centre for Advanced Agricultural Science and Technology (CAAST) is a new initiative and student centric subcomponent of World Bank sponsored **National Agricultural Higher Education Project (NAHEP)** granted to the Indian Council of Agricultural Research, New Delhi to provide a platform for strengthening educational and research activities of post graduate and doctoral students. The ICAR-Indian Agricultural Research Institute, New Delhi was selected by the NAHEP-CAAST programme. NAHEP sanctioned Rs 19.99 crores for the project on "**Genomic assisted crop improvement and management**" under CAAST programme. The project at IARI specifically aims at inculcating genomics education and skills among the students and enhancing the expertise of the faculty of IARI in the area of genomics.

Objectives:

1. **To develop online teaching facility and online courses for enhancing the teaching and learning efficiency, and scientific communication skills**
2. **To develop and/or strengthen state-of-the art next-generation genomics and phenomics facilities for producing quality PG and Ph.D.students**
3. **To develop collaborative research programmes with institutes of international repute and industries in the area of genomics and phenomics**
4. **To enhance the skills of faculty and PG students of IARI and NARES**
5. **To generate and analyze big data in genomics and phenomics of crops, microbes and pests for genomics augmentation of crop improvement and management**

IARI's CAAST project is unique as it aimed at providing funding and training support to the M.Sc. and Ph.D. students from different disciplines who are working in the area of genomics. It will organize lectures and training programmes, send IARI students for training at expert laboratories and research institutions abroad, and cover students from several disciplines. It will provide opportunities to the students and faculty to gain international exposure. Further, the project envisages developing a modern lab named as **Discovery Centre** that will serve as a common facility for students' research at IARI.

Core-Team Members:

S.No.	Name of the Faculty	Discipline	Institute
1.	Dr. Ashok K. Singh	Genetics	ICAR-IARI
2.	Dr. Vinod	Genetics	ICAR-IARI
3.	Dr. Gopala Krishnan S	Genetics	ICAR-IARI
4.	Dr. A. Kumar	Plant Pathology	ICAR-IARI
5.	Dr. T.K. Behera	Vegetable Science	ICAR-IARI
6.	Dr. R.N. Sahoo	Agricultural Physics	ICAR-IARI
7.	Dr. Alka Singh	Agricultural Economics	ICAR-IARI
8.	Dr. A.R. Rao	Bioinformatics	ICAR-IASRI
9.	Dr. R.C. Bhattacharya	Molecular Biology & Biotechnology	ICAR-NIPB
10.	Dr. K. Annapurna	Microbiology **Nodal officer, Grievance Redressal, CAAST**	ICAR-IARI
11.	Dr. R. Roy Burman	Agricultural Extension **Nodal officer, Equity Action Plan, CAAST**	ICAR-IARI
12.	Dr. K.M. Manjaiah	Soil Science & Agri. Chemistry **Nodal officer, CAAST**	ICAR-IARI
13.	Dr.Viswanathan Chinnusamy	Plant Physiology **PI, CAAST**	ICAR-IARI

Associate Team

S.No.	Name of the Faculty	Discipline	Institute
14.	Dr. Kumar Durgesh	Genetics	ICAR-IARI
15.	Dr. Ranjith K. Ellur	Genetics	ICAR-IARI
16.	Dr. N. Saini	Genetics	ICAR-IARI
17.	Dr. D. Vijay	Seed Science & Technology	ICAR-IARI
18.	Dr. Kishor Gaikwad	Molecular Biology & Biotechnology	ICAR-NIPB
19.	Dr. Mahesh Rao	Genetics	ICAR-NIPB
20.	Dr. Veena Gupta	Economic Botany	ICAR-NBPGR
21.	Dr. Era V. Malhotra	Molecular Biology & Biotechnology	ICAR-NBPGR
22.	Dr. Sudhir Kumar	Plant Physiology	ICAR-IARI
23.	Dr. Dhandapani R	Plant Physiology	ICAR-IARI
24.	Dr. Lekshmy S	Plant Physiology	ICAR-IARI
25.	Dr. Madan Pal	Plant Physiology	ICAR-IARI
26.	Dr. Shelly Praveen	Biochemistry	ICAR-IARI
27.	Dr. Suresh Kumar	Biochemistry	ICAR-IARI
28.	Dr. Ranjeet R. Kumar	Biochemistry	ICAR-IARI
29.	Dr. S.K. Singh	Fruits & Horticultural Technology	ICAR-IARI
30.	Dr. Manish Srivastava	Fruits & Horticultural Technology	ICAR-IARI
31.	Dr. Amit Kumar Goswami	Fruits & Horticulture Technology	ICAR-IARI
32.	Dr. Srawan Singh	Vegetable Science	ICAR-IARI
33.	Dr. Gograj S Jat	Vegetable Science	ICAR-IARI
34.	D. Praveen Kumar Singh	Vegetable Science	ICAR-IARI
35.	Dr. V.K. Baranwal	Plant Pathology	ICAR-IARI
36.	Dr. Deeba Kamil	Plant Pathology	ICAR-IARI
37.	Dr. Vaibhav K. Singh	Plant Pathology	ICAR-IARI
38.	Dr. Uma Rao	Nematology	ICAR-IARI
39.	Dr. S. Subramanium	Entomology	ICAR-IARI
40.	Dr. M.K. Dhillon	Entomology	ICAR-IARI
41.	Dr. B. Ramakrishnan	Microbiology	ICAR-IARI
42.	Dr. V. Govindasamy	Microbiology	ICAR-IARI
43.	Dr. S.P. Datta	Soil Science & Agricultural Chemistry	ICAR-IARI
44.	Dr. R.N. Padaria	Agricultural Extension	ICAR-IARI
45.	Dr. Satyapriya	Agricultural Extension	ICAR-IARI
46.	Dr. Sudeep Marwaha	Computer Application	ICAR-IASRI
47.	Dr. Seema Jaggi	Agricultural Statistics	ICAR-IASRI
48.	Dr. Anindita Datta	Agricultural Statistics	ICAR-IASRI
49.	Dr. Soumen Pal	Computer Application	ICAR-IASRI
50.	Dr. Sanjeev Kumar	Bioinformatics	ICAR-IASRI
51.	Dr. S.K. Jha	Food Science & Post Harvest Technology	ICAR-IARI
52.	Dr. Shiv Dhar Mishra	Agronomy	ICAR-IARI
53.	Dr. D.K. Singh	Agricultural Engineering	ICAR-IARI
54.	Dr. S. Naresh Kumar	Environmental Sciences; **Nodal officer, Environmental Management Framework**	ICAR-IARI

Foreword

The Division of Agricultural Economics, a constituent of the School of Social Sciences of ICAR-Indian Agricultural Research Institute, was established in 1960. The mandate of the Division is to conduct research in frontier areas and serve as a centre for academic excellence in post-graduate education. Since its inception, the Division has been making contributions in basic and applied research with significant implications for agricultural policy. The Division has achieved excellence in post-graduate education and research as an ICAR-UNDP Centre of Excellence through a faculty exchange program for human resources development and strengthening of infrastructure facilities. Since 1995 it has been functioning as an ICAR Centre of Advanced Faculty Training (CAFT) to strengthen the capacity for agricultural economics and policy research in the national agricultural research system.

The research contributions of the Division have been globally recognized and many of the alumni occupy positions of repute in national and international organizations. The Division has maintained good academic liaison with other divisions at IARI, and other national and international agricultural research institutions. The research focus of the Division has been continuously reoriented to address contemporary development challenges. Current research thrust areas of the division include investment in agriculture, inclusive growth, and poverty alleviation, the impact of agricultural technologies and policies, price forecasting and market outlooks, natural resource use in agriculture and ecosystem services, climate change effects, mitigation and adaptations, and food and nutritional security.

The Division has been in the forefront on the use of the latest research methods and analytical tools in its research activities that ensures quality research and reliable estimates. Considering this strength of the Division in forecasting techniques, a 10 days online training course on **"Time Series Techniques for Forecasting in Agriculture"** was organized by the Division from 1 to 10 December 2021. The objective of this training program, sponsored by the Centre for Advanced Agricultural Science and Technology (CAAST) component of the World Bank-funded National Agricultural Higher Education Project (NAHEP), was to upgrade the research skills of the students of social science discipline. The training program covers a range of topics including web resources and research data management, introduction to R and stata softwares, extraction of large datasets, commodity outlook modelling, application of remote sensing and GIS data in research, univariate and multi-equation time series models, futures trading, Artificial Intelligence (AI), machine learning and LSTM techniques for price forecasting etc. I am sure that the lectures on various forecasting techniques and practical sessions will be of immense help to the participants.

.

Rashmi Aggarwal

Dean and Joint Director(Edn)
ICAR-IARI,New Delhi

Date: 10 .12.2021

Preface

Evidence based social science research is of great importance as it provides for framing policies, estimating impacts and accurate forecasting. It employs robust qualitative and quantitative methods for obtaining reliable estimates. The recent advancements in the methodology and analytical techniques, as well as their applications in the field of social sciences, necessitates the students to get updated on these techniques so as to deliver quality reaserch. This training manual is prepared with the aim of guiding the post-graduate students of social sciences to employ these techniques effectively. The manual is based on the NAHEP-CAAST sponsored online short training course titled "Time Series Techniques for Forecasting in Agriculture" organized by the Division of Agricultural Economics, ICAR-Indian Agricultural Research Institute, New Delhi. Centre for Advanced Agricultural Science and Technology (CAAST) is a new initiative and student-centric sub-component of World Bank sponsored National Agricultural Higher Education Project (NAHEP), granted to IARI to provide a platform for strengthening education and research activities of post-graduate students.

Social science research, particularly in the applied disciplines of Agricultural Economics, Agricultural Extension and Agricultural Statistics, is characterised by a diversity of theoretical perspectives, substantive orientation, methodological strategies, data collection practices and analytical techniques. The students of these disciplines usually have to face challenges in research, since it involves conceptualizing the problems relevant to the stakeholders, collecting and handling large data sets (both primary as well as secondary), choosing appropriate methodology (qualitative and quantitative), executing the analysis using appropriate statistical packages, and interpreting and presenting the results in a meaning and useful format to all: farmers, academia and policymakers.

Recognizing the need for imparting the essential research skills to the social science students, we have taken up the task of conducting the training and preparing this manual on- Time Series Techniques for Forecasting in Agriculture. The various chapters of this manual are contributed by the eminent social scientists of the country, with expertise in forecasting using different methods. In addition to the basic research methods, the manual also covers topics like web resources and research data management, introduction to R and stata softwares, extraction of large datasets, commodity outlook modelling, application of remote sensing and GIS data in research,univariate and multi-equation time series models, futures trading, Artificial Intelligence (AI), machine learning and LSTM techniques for price forecasting etc. We take this opportunity to sincerely acknowledge the contribution of all the authors in the preparation of this manual. This manual can act as a quick and effective reference source for the students in their future research endeavours especially in the area of forecasting.

Alka Singh
Girish K Jha
Nithyashree M L
Asha Devi S S

Date: 10.12.2021

Acknowledgments

1. Secretary DARE and Director General ICAR, New Delhi
2. Deputy Director General (Education), ICAR, New Delhi
3. Assistant Director General (HRD), ICAR, New Delhi
4. National Coordinator, NAHEP, ICAR, New Delhi
5. CAAST Team, ICAR-IARI, New Delhi
6. P.G. School, ICAR-IARI, New Delhi
7. Director, ICAR-IARI, New Delhi
8. Dean & Joint Director (Education), ICAR-IARI, New Delhi
9. Joint Director (Research), ICAR-IARI, New Delhi
10. Head, Division of Agriculture Economics, ICAR-IARI, New Delhi
11. Professor, Division of Agriculture Economics, ICAR-IARI, New Delhi
12. AKMU, ICAR-IARI, New Delhi
13. Staff & Students, Division of Agriculture Economics, ICAR-IARI, New Delhi

Contents

Introduction to R Software

A. Dhandapani

National Academy of Agricultural Research Management, Rajendranagar, Hyderabad

dhandapani@naarm.org.in

Introduction

R is a statistical computing environment having facilities for data manipulation, calculation, graphical display etc. R is also a free implementation of S language, which was developed for statistical computing and graphics by John Chambers of Bell Labs. A commercial version of S is also available as S+® from Insightful Co. R was developed initially by Ross Ihaka and Robert Gentleman. R is developed as open source software and available free for use.

Installation of R

R package can be downloaded from the R project site, www.r-project.org . The current version is 2.15.0. R is also available for wide variety of operating systems such as Microsoft ®Windows®, Mac OS X and Unix X11 (Linux). There are several additional packages available which can also be downloaded for free.

To install R package in MS-Windows ® platform, download the latest version of R from the following location: http://cran.r-project.org/bin/windows/base/. The file is called R-2.15.0-win.exe (~48 MB). Run this file and follow the instruction in the screen to install it. To run the software, go to Start → Programs → R→R-2.15.0 and click. The default R screen should appear with the title R GUI/R console with a > symbol in Red colour and red blinking cursor. Interaction with R software can be carried out with this command prompt.

Getting Help

Complete help files in HTML and PDF forms are available. There is an excellent introduction to R package can be found in Help → Manuals (in pdf) "A introduction to R". Several other documentations are also given at the end of this lecture note. To get help on a particular command/function etc., type help (command name). For example, to get help on function 'mean', type in the command prompt(>)

```
>help(mean)
```

This will open the help file with the page containing the description of the function mean. In this lecture note, all R commands and corresponding outputs are given as shaded text to differentiate the normal texts. It should be noted that R is case-sensitive, i.e. typing Help(mean), would get an error message,

```
> Help(mean)
Error: could not find function "Help"
>
```

Packages

The strength of R comes from the optional packages available (again freely). The complete list is available at R-Project website. Packages can be installed either by using the Install Package option within R or download the packages from R-Project website and using the option, install

from local Zip files. Packages are to be loaded before using through Load Package option. For more details, refer R Help system.

Using R

R can also be used as a calculator. For example,

```
> 2+(3*2)
[1] 8
```

To compute, say roots of the equation, $x^2 - 3x + 2$, the following commands can be used:

```
> (3+sqrt(-3*-3-(4*1*2)))/(2*1)
[1] 2
> (3-sqrt(-3*-3-(4*1*2)))/(2*1)
 [1] 1
```

In the above example, a built-in function sqrt() is used to evaluate the square root of a number. Try to compute the square root of a negative number,

```
> sqrt(-4)
[1] NaN
Warning message:
In sqrt(-4) : NaNs produced
```

R produces a warning message saying that the result is not a number (NaN).

Workspace and History

To exist R at the end, type q(). A warning message will be displayed, whether to save the workspace or not. If workspace is saved, R remembers all the objects/variables etc used /created during the particular session. The saved workspace can be loaded into R next time and one can continue working. Thus, one need not recreate again all the objects/variables. Also, the commands entered during a session can be saved as history. This will be useful to store all the relevant commands to run a particular problem in a file and can be run directly with out typing the commands again and again.

Data Types

Vector

In any programming environment, the foremost thing is to learn the various data types supported. Also, in statistics, data comes in various forms such as numeric, categorical, multivariate, time series etc. R supports various data types such as numeric vectors/matrices; character vectors; categorical, logical etc. Simplest structure is a numeric vector, which stores ordered collection of numbers of any specific dimension. To begin with, create a vector, say **a** with 4 entries 1,0,-1 & 3**,** i.e.

$\mathbf{a} = \begin{pmatrix} 1 & 0 & -1 & 3 \end{pmatrix}'$, in R use

```
a<-c(1,0,-1,3)
a
```

Output:
[1] 1 0 -1 3

In this example, a new notation is introduced, namely **"<-"**(< sign followed by -). The statement is an assignment statement using a function c(). The assignment symbol is <- rather than the usual "=". The function c() can take arbitrary number of arguments whose values are used to create the vector in this case. If an expression is typed with out assigning to any variable, then its value is printed. For example, type

```
-a
```

Output:
[1] -1 0 1 -3

On the other hand, type

```
b<- -a
b
```

Output:
[1] -1 0 1 -3

Now, **b** is also a vector.

Further, the following statement,

```
c<- c(a, -a, b,1,2)
c
```

Output:
[1] 1 0 -1 3 -1 0 1 -3 -1 0 1 -3 1 2

produces a vector c of dimension 14.

The other data types such as frame is discussed in "Reading Data" section.

Functions in R

R provides commonly used functions in mathematical and statistical world, such as trigonometric, mean, median etc. More over, several special functions are available. In R, the same function can be used for more than one purpose and may have arbitrary number of arguments. Some functions may have default arguments which need not be specified. For example, a simple call to the function mean with a vector, produces the average of the arguments of the vector. To get trimmed mean (i.e. discard 5 per cent values both the sides) use an extra parameter with value, trim =0.05. If trim is not specified, it is taken as 0 (default value). The function mean also has another argument, called na.rm which is a logical value, taking either true or false. This argument specifies whether the values which are marked as "Not Available" to be taken into consideration while computing averages or not (rm stands for remove). By default, it is set to be fault. Hence, it is important to understand different arguments

of the functions and their roles before using them. However, most of the time, the default values are only required.

Reading Data

The data given in Example 1 will be used here. The 4 parameters can be read into R as 4 vectors say NrFruits, FruitWt, SeedYld, SeedlingLen using the following commands:
(Note: For brevity, only few values are included. In actual code, include all the values)

```
NrFruits <- c(7.0,7.0,,…., 7.0,7.3)
FruitWt <- c(1.85,1.86, …., 2.45,2.44)
SeedYld <- c(147.70, 136.86,….199.87,214.30)
SeedlingLen <- c(16.86,16.77,….,19.31,19.36)
```

Once the 4 vectors are created, it can be used in several ways. Now, to get a 1st item in NrFruits and all except say 3rd item, use the following commands:

```
NrFruits[1]
NrFruits[-3]
```

To view items 1 to 5, use:

```
NrFruits[1:5]
```

For the Group column, one can use c() function by specifying all the 20 values. But, this can be easily generated using ":" notation and rep() function such as:

```
Group <- rep(1:2,each=10)
```

The statement rep(1:2,each=10) repeats the values 1 and 2 for ten times to form the Group Vector.

All the variables NrFruits, FruitWt, SeedYld, SeedlingLen and Group can be made as one single variable using cbind() function:

```
FruitData<-cbind (Group, NrFruits, FruitWt, SeedYld, SeedlingLen)
FruitData
```

The part of the output is shown below:

```
     Group NrFruits FruitWt SeedYld SeedlingLen
[1,]     1      7.0    1.85  147.70       16.86
[2,]     1      7.0    1.86  136.86       16.77
[3,]     1      6.0    1.83  149.97       16.35
[4,]     1      7.0    1.89  172.33       18.26
[5,]     1      7.0    1.80  144.46       17.90
```

Reading from files

Though, the function c() can be easy to assign a data, it would be difficult to enter data in that way for even a small number of observations. It is possible to read from different data sources

such as simple text files, comma separated files etc to sophisticated data sources such as databases, excel files, web etc. To start with, to read from a simple text file, say, data.txt in which the columns are separated by a comma and the first line contains the names of the variables.

```
Fruitdata<– read.table (“data.txt”,header=TRUE, sep= “,”)
```

This assumes that data.txt is in the current path. Otherwise you have to issue the full path where data.txt is stored. For example, to read data.text from the folder, “C:\Users\user\Documents\R Lecture” in Windows machines, use the following command:

```
FruitData<-read.table("C://Users/user/Documents/R Lecture/data.txt",
header=TRUE,sep= “,”)
```

The syntax is very simple, as it uses a function read.table; the first argument is the name of the file, the second argument says that there is a header (the first line of the file) and the values are separated by a comma.

Data Frames

The function read.table returns the results to a Data Frame. A Data Frame in R is nothing but a collection of columns of equal length. Data frames are useful in many ways while doing any statistical analysis. For example, if you want to fit a linear regression between two variables, it is better to build a data frame consisting of dependent as well as independent variables in a single data frame. Individual columns are not directly accessible by their names. For example, the column Group in FruitData can be accessible using:

```
FruitData$Group
```

Reading from Excel

The recommended options for reading data from MS-Excel are first save the file either as TAB-delimited or CSV (Comma Separated Values) and use read.table() function. This will make sure that the data is read properly. It is also possible to read from MS-Excel files directly using ODBC or using R package, xlsReadWrite.

Descriptive Statistics using R

The descriptive statistics such as mean, median, etc. can be easily calculated in R. Using the data frame, FruitData, consider the following R statements:

```
#Descriptive Statistics
#Print Mean of NrFruits
mean(FruitData$NrFruits)
# Print Median of NrFruits
median(FruitData$NrFruits)
#use of attach to make every column available directly
attach(FruitData)
#find Mean of Number of Fruits
mean(NrFruits)
```

Some of the other functions which can calculate descriptive statistics are mean(), median(), max(), min, sd(),var() etc. The following table provides R syntax for various descriptive statistics.

Name of the Statistics	**R Function**	**Usage**
Mean	mean()	mean(NrFruits)
Median	median()	median(NrFruits)
Quartiles	quantile()	quantile(NrFruits)
Percentiles	quantile()	quantile(NrFruits,c(0.01,0.30)) To get 1 and 30 percentiles
Standard Deviation	sd()	sd(NrFruits)
Variance	var()	variance(NrFruits)
Range	max() – min()	max(NrFruits) – min(NrFruits)
Inter Quartile Range	IQR()	IQR(NrFruits)
Skewness	skewness()	skewness(NrFruits)
Kurtosis	kurtosis()	kurtosis(NrFruits)

Function tapply()

Most of the time, one would like to calculate descriptive statistics for several columns together. Moreover, consider the calculation of mean of Number of Fruits for the 2 groups. The two group means can be calculated by the following two statements:

```
mean(NrFruits[Group==1])
mean(NrFruits[Group==2])
```

In case, if the number of groups is large, clearly the above approach would be time consuming. In such cases, one can use tapply() function. To use this function, three arguments may be specified: (i) the variable (column) name for which some summary statistics is to be calculated; (ii) the grouping variable and (iii) summary function. Thus, to get the same result as that of the above two R statements, use:

```
tapply(NrFruits,Group,mean)
```

Summary Function

The function, summary(<variable name/dataframe name>) provides the following information for the variable(s): mean, minimum, maximum, Median and 1st and 3rd quartiles.

Plotting the Data

Before starting of any statistical analysis, it is usually advised to create different types of plots to understand the data. For example, before proceeding to find out the correlation or fit a regression model between two variables, scatter plots may be produced to know the relationship between the variables. In R, several functions are available to produce several plots such as Histogram, Box Plot, XY Plot etc. It is also possible to produce graphs for several variables or same graph for each group etc. in one single panel.

Histograms

Histograms show the frequency distribution of a quantitative variable as vertical bars with area of the bar denotes the frequency of items found in each class interval. Histograms are useful to assess the distribution of the variable. To generate a histogram, hist() function is used in R. For example, to shown histogram of Number of Fruits (NrFruits), use the following code:

```
hist(NrFruits)
```

To produce a histogram with a single color or with different colours, use the following syntax:

```
hist(NrFruits,col = “red” )
multi<- c("blue","red","brown","yellow")
hist(NrFruits,col=multi)
```

To produce a normal curve within the histogram, use the following code:

```
h<-hist(NrFruits,col="red")
x<-seq(min(NrFruits),max(NrFruits),length=50)
y<-dnorm(x,mean=mean(NrFruits),sd=sd(NrFruits))
y<-y*diff(h$mids[1:2])*length(NrFruits)
lines(x,y,col="blue",lwd=4)
```

Box Plots

Box plots are useful to show multiple descriptive statistics of a variable in a single graph. The box plots, also called Box and Whisker plots, can be used for comparing different groups. To produce box plot, use boxplot() function in R.

```
boxplot(NrFruits)
```

To produce a box plot for two groups together, use

```
boxplot(NrFruits~Group)
```

In R, various functions/methods are available to carry out various statistical analyses, such as ANOVA, MANOVA, Principal Components, Time Series Analysis etc. For a complete details of each of the method, can be obtained from R-Reference Manual supplied with the R package. For illustration few examples are provided.

T-test

In example 1, if the interests are to:

1. Test whether the mean of the population of Seed yield/plant (g) is 200 or not.
2. Test whether the natural pollination and hand pollination under open field conditions are equally effective or are significantly different.

To answer the question 1 of the example, enter the following R command:

```
t.test(SeedYld,mu=200)
```

The output is shown below:

```
        One Sample t-test

data:  SeedYld
t = -2.3009, df = 19, p-value = 0.03289
alternative hypothesis: true mean is not equal to 200
95 percent confidence interval:
 163.3414 198.2656
sample estimates:
mean of x
 180.8035
```

Similarly, for the question 2 of the example, the relevant R statements are:

```
t.test(NrFruits[Group==1],NrFruits[Group==2])
t.test(FruitWt[Group==1], FruitWt [Group==2])
t.test(SeedYld [Group==1], SeedYld [Group==2])
t.test(SeedlingLen [Group==1], SeedlingLen [Group==2])
```

The general syntax of t-test in R is as follows:

t.test (x,….)
The arguments (only few are given, for complete list, see R documentation) of t.test are as follows:

x(required)	A non-empty numeric vector of data values
Optional arguments	
Y	A non-empty numeric vector of data values; required for two-sample t-test
Alternative	Takes the following values: “two.sided”, “greater” and “less”. The default value is “two.sided”
Mu	Default value is zero. Specify true value of the mean to be tested in case of single sample t-test and difference in means to be tested in case of in case of two sample t-test.
Paired	Logical value (TRUE/FALSE). TRUE for paired t-test
var.equal	Logical value (TRUE/FALSE). If TRUE, variances of two samples are assumed to be equal and pooled variance is used for estimating the common variance. Otherwise Welsh approximation for d.f. is used.
conf.level	Confidence level of the interval

It may be appreciated that even though there are of lot of arguments, only few are enough to run the command.

Example dataset

The following example for t-test is available at Design Resource Server of IASRI. (http://www.iasri.res.in/design/Analysis%20of%20data/ttest1.html)

Example: An experiment was conducted to study the hybrid seed production of bottle gourd **(***Lagenaria siceraria (Mol) Standl)* Cv. Pusa hybrid-3 under open field conditions during Kharif-2005 at Indian Agricultural Research Institute, New Delhi. The main aim of the investigation was to compare natural pollination and hand pollination under field conditions. The data were collected on 10 randomly selected plants from each of natural pollination and hand pollination. The data were collected on number of fruit set for the period of 45 days, fruit weight (kg), seed yield per plant (g) and seedling length (cm). The data obtained is as given below: {Here 1 denotes natural pollination and 2 denotes the hand pollination}

Group	No. of fruitSet(45days)	Fruit weight (kg)	Seed yield/plant (g)	Seedling length (cm)
1	7.0	1.85	147.70	16.86
1	7.0	1.86	136.86	16.77
1	6.0	1.83	149.97	16.35
1	7.0	1.89	172.33	18.26
1	7.0	1.80	144.46	17.90
1	6.0	1.88	138.30	16.95
1	7.0	1.89	150.58	18.15
1	7.0	1.79	140.99	18.86
1	6.0	1.85	140.57	18.39
1	7.0	1.84	138.33	18.58
2	6.3	2.58	224.26	18.18
2	6.7	2.74	197.50	18.07
2	7.3	2.58	230.34	19.07
2	8.0	2.62	217.05	19.00
2	8.0	2.68	233.84	18.00
2	8.0	2.56	216.52	18.49
2	7.7	2.34	211.93	17.45
2	7.7	2.67	210.37	18.97
2	7.0	2.45	199.87	19.31
2	7.3	2.44	214.30	19.36

An introduction to STATA Software and extraction and handling of unit-level data sets of NSSO and ASI

Nithyashree M L

Division of Agricultural Economics, ICAR-IARI, New Delhi 12
nithya.econ@gmail.com

An Introduction to STATA

This chapter describes an overview and basic commands used in the STATA software which will help the beginners to get familiar and hold a grip in using the software for further statistical analysis. Also, this chapter is aimed to introduce the different unit-level data sets available and the way of handling them. Those who don't have access to STATA software can avail the short-term student license service by using the link Student short-term license request | Stata. As shown below the main interphase (Figure 1) of the STATA has a menu system that enables the users to perform the task by interactive UI; drop-down menu. Alternatively, there is also a command window with which users can write commands directly for the statistical analysis. For example, to summarize data, by typing summarize variable/variables name, we can get the result and one can also do it by using the drop-down menu by going to current statistics (Figure 2). As a beginner one can start by exploring the possible option available by using the drop-down menu, after getting comfortable with the software, can directly type the command to perform any required analysis which is more efficient and professional, apart from this the executed commands can be copied from the history/review window and saved in do file, which can be easily shared with other researchers and this enables single-click execution. With the right side of the interphase, variables, and properties window variables and their related properties including labelling the variables and assigning variables values can be performed and visualized.

STATA provides flexibility in exploring the various option to the users to learn the available functions and packages with the use of the *help* command. By using this command one can learn and use the applicability of different functions and packages. For example, by typing help summarize in the command window we get detailed information of the command summarize in the form of pdf also by scrolling down detailed information on using the summarize command directly as syntax as well as menu format along with some example data sets a shown below is a great source to learn STATA.

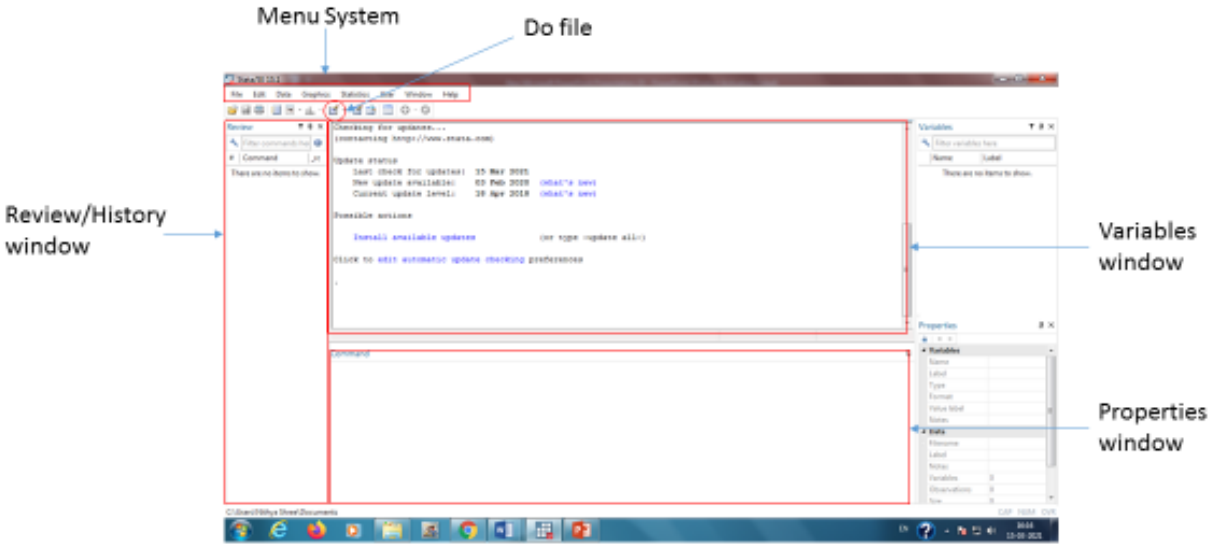


Figure 1. Main interphase of STATA

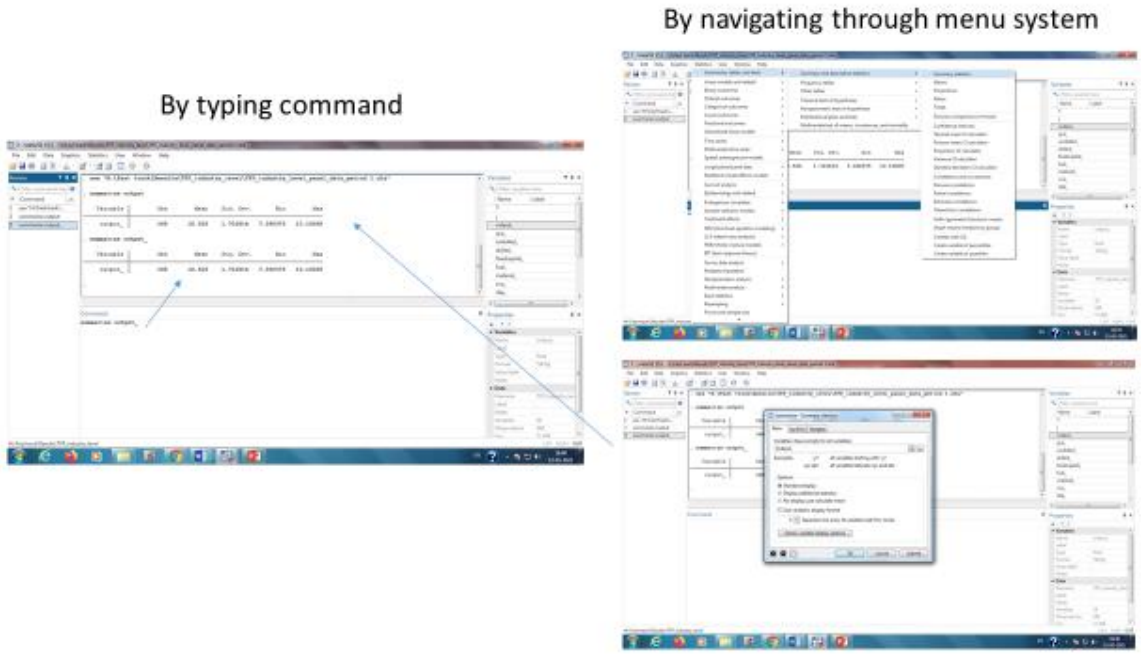
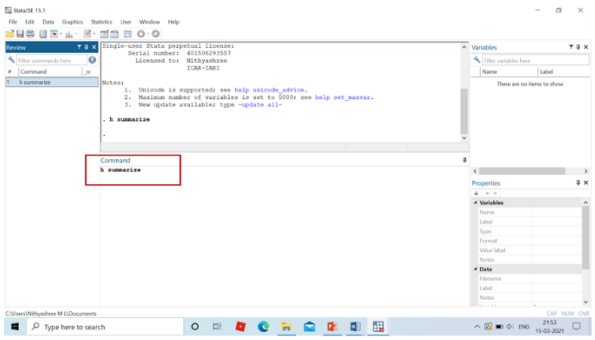
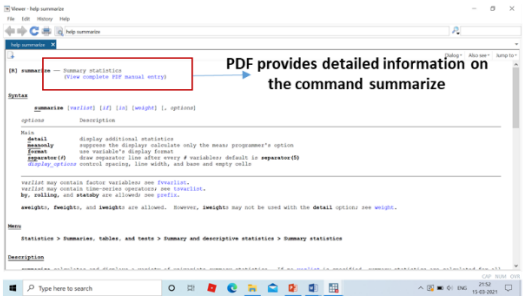
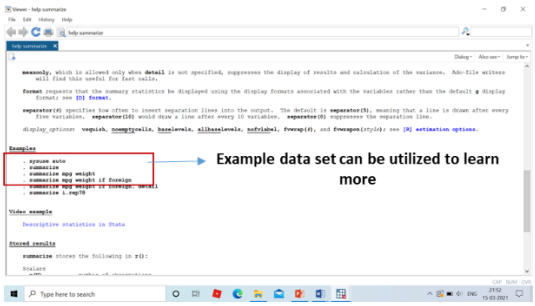


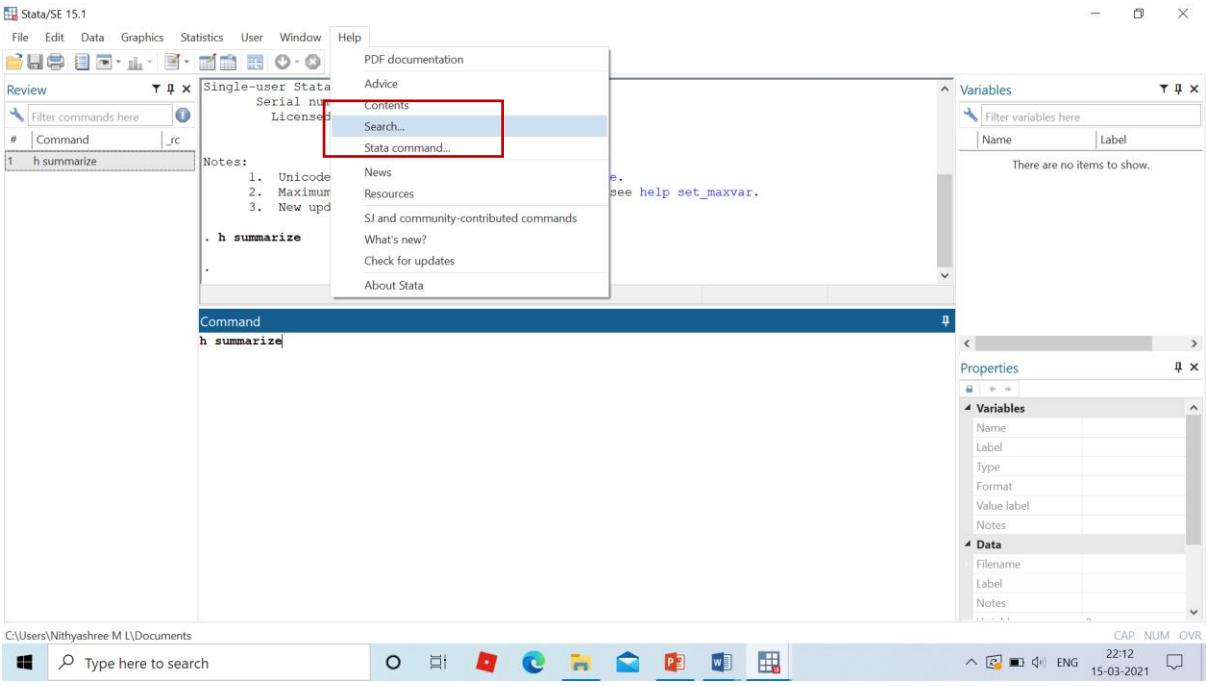
Figure 2. Two ways of performing task in STATA







If the user is not sure about what to type in the command window, then the search option under the help in the menu bar can be explored. Besides there are many user-written commands are available in the STATA software in the form of packages, to use them first we need to install the packages. For this one can use *findit* command and after finding the suitable package install them to make use of those packages and also make sure your system connected to the internet while installing the packages.



It is always good practice to save the results of the analysis, that can be done in STATA by typing *log using filename* in the command window before beginning the analysis, which creates a log file by the given file name and after completing the analysis type *log close* and your analysis will be saved in the smcl format for example filename.smcl, which details all the activity which you carried out during the particular session or analysis. Which you may need at the later stage to communicate to the journals while submitting the research article.

Mathematical and logical operator in STATA are similar to those used in MS excel

a == b if a equals b
a != b if a not equal to b
a > b if a greater than b
a >= b if a greater than or equal to b
a < b if a less than b
a <= b if a less than or equal to b
a & b & refers to and
a | b | refers to or

Handling the data set

1. **Creating a new variable:** gen or egen command is used to create the new variable. These commands can be combined with arithmetic operators or logical operators
   Example: gen grade=1 if marks==2
   egen stdev_age= std(income)
   gen ln_wage = ln(wage)

2. **For labelling the variable:** to create a label to the variable, write the command label variable and type the variable name need to be labelled then write the label with in the invited comma
   Example: label variable ln_wage "Log of hourly wage"

3. **The replace command:** Replace command generally helps in editing value of already existing or generated variable, for this command ***replace*** can be used.
   Example: replace gender=0 if missing(gender)

replace gender=0 if gender==.

4. To sort the data in ascending order: use command ***sort variable name***
5. Command that can convert string to numeric variables: ***Tostring and destring***

Few Commands for Basic Statistical analysis

a. Regression: *reg or probit or logit*
b. Correlation: *corr or pwcorr*
c. Student T test: *ttest*
d. Factor analysis: *factor*
e. Marginal Effects after probit or logit: *mfx*
f. Chisquare test: *tab, all*
g. Principal Component Analysis: *PCA*

Few useful user written commands

h. *tatable2*: Calculate group wise mean value and test the significance
i. *orth_out*: Perform t-test for any number of variables at once
j. *dea*: Data envelopment analysis
k. *acfest*: Production function est. using Ackerberg-Caves-Frazer method
l. *levpet*: Production function est. using Levinsohn and Petrin approach
m. doubleb: Perform Double Bound Contingent Valuation.
n. clustersampsi: perform power calculations for RCT

Exercise- 1

1. Import the dataset “stata_intro”, which has been shared with you already.
2. Summarize the data so that you see the means, standard deviation, min, and max of each variable
3. generate the logarithmic transformation of the variable wage as ln_wage
4. label variable ln_wage, wage, collgrad and union as Log of hourly wage, Hourly wage, College graduate and Union member respectively.
5. define label values for the variable collgrad

6. encode racecat as race
7. obtain mean wage sd wage for the variable union
9. draw histogram for the variable ln_wage and superimpose normal curve
10. generate scatter plot for the variables wage tenure
11. generate scatter plot for the variables wage tenure by union
12. obtain a liner regression equation of ln_wage and tenure
13. obtain a liner regression equation ln_wage and tenure along with by considering any one of the categorical variable and also an interaction component
14. save the commands in do file and results in log file.

Extraction and handling of unit-level data

As a beginner before getting the hands-on unit-level data, it is important to go through the key indicators or summary results. This helps to understand the purpose of the survey and the data coverage in the particular dataset. Besides, this will also help the user to comprehend the sample size, sampling design and estimation procedure which is necessary for further analysis and interpretation of the results. Since the key indicators/summary results, by and large, provides the aggregate estimate, the use of unit-level data enables the researchers to work with the basic unit of survey i.e. household in case of agriculture and firm with regard to the industry sector. The unit-level data sets are generally available in text format, to convert these files into a usable format, some of the steps need to be understood and they are discussed by using two unit-level datasets *viz.*, Key Indicators of Situation of Agricultural Households in India of National Sample Survey Organization (NSSO) and Annual Survey of Industries (ASI). For handling any unit-level data it is essential to use suitable statistical software, here we use STATA (version 15) software for the illustration and the summary of the steps to be followed is shown in Figure 1.

The first and most important step in using unit-level data is to get a though understanding of the supporting documents they mainly comprise the layout which instructs, how the data arranges in the text file and other details such as tabulation programme, code list, concept and definition and schedule, etc. For extracting data from the text file one has to

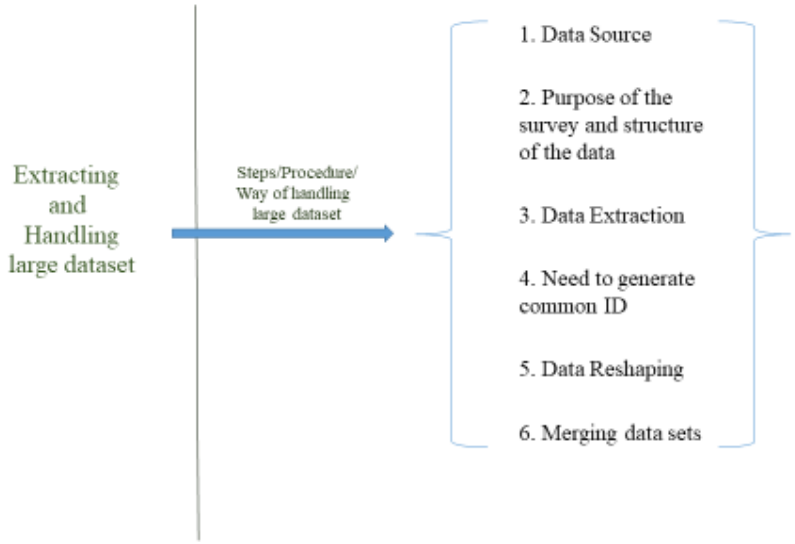


Figure 1. Steps to extract and handle the unit-level data

write the commands in *STATA* by using the information given in the layout. For example, for the NSSO data set it can be written as:

infix Centercode 1-3 FSU 4-8 Round 9-10 Schedule 11-13 Sample 14 Sector 15 NSS_Region 16-18 District 19-20 Stratum 21-22 Sub_Stratum 23-24 Sub_Round 25 Sub_sample 26 FOD_subRegion 27-30 Hamlet 31 SecondStageStratum 32 /// SampleHHno 33-34 VisitNo 35 Level 36-37 Filler 38-40 HouseholdSize 43-44 Religion 45 Social_Group 46 Dwelling 47 StructureType 48 WaterSource 49 HH_OwnLand_YN 50 LandType 51 PossesLandOutsideVillage 52 LandOperated 53 Land_Own_ha 54-62 Land_Leasedin_ha 63-71 Land_neitherLeased 72-80 Land_LeasedOut_ha 81-89 Land_total_possessed 90-98 Cultivation_whetherPerformed 99 Cultivation_incomesource 100 Livestock_whetherPerformed 101 Livestock_incomesource 102 OtherAgr_whetherPerformed 103 OtherAgr_incomesource 104 NonAgr_whetherPerformed 105 NonAgr_incomesource 106 Wage_whetherPerformed 107 Wage_incomesource 108 Pension_whetherPerformed 109 Pension_incomesource 110 Remittance_whetherPerformed 111 Remittance_incomesource 112 Others_whetherPerformed 113 Others_incomesource 114 MGNREGACard_YN 115 RationCard_YN 116 RationCard_Type 117 NSS 127-129 NSC 130-132 MLT 133-142 using "C:\Users\Nithyashree M L\Desktop\Stata_Workshop\Unit level data\AH0233V1.TXT"

where the command infix is written to extract the text file into *STATA* file and at the end data path has been specified. Similarly, for the ASI data it can be written as:

infix year 1-4 Factory 5-9 State 10-11 str C 12 str Block 13 Scheme_code 14 NIC4digit 15-18 NIC5digit 19-23 R_U 24 RO_SRO 25-29 noofunits 30-32 Statusofunit 33-34 Manufacturingdays 35-37 Non_Manufacturingdays 38-40 Total 41-43 Costofproduction 44-57 directlyexported 58-60 Multiplierfactor 61-73 using "C:\Users\Nithyashree M L\Desktop\Stata_Workshop\Unit level data\ASI\OASBLA16.TXT"

After extracting the data files, it is important to create the common ID or unique ID in each data file. This will help to identify the household across the different data sets and also enables to combine the information available in different data files. For creating a common id, the following command in *STATA* is written as:

NSSO

egen id=concat (FSU Hamlet SecondStageStratum SampleHHno)

ASI

egen id=concat(Factory State C)

To combine the information of different data files, the researcher has to ensure that the observations are uniquely identified across the data sets, if not the data needs to be rearranged. By observing the data structure, it can be grouped as wide-format or long format. For example, in Figure 2, students' ids and results are written in the wide-format in Table 1 and the same students' subject-wise grades are arranged in the long format in Table 2. To combine the information of grade and results, Table 2 has to be reshaped/rearranged into a wide format, which can be done by using the command reshape in STATA i.e. reshape wide Grade, i (ID) j(J).

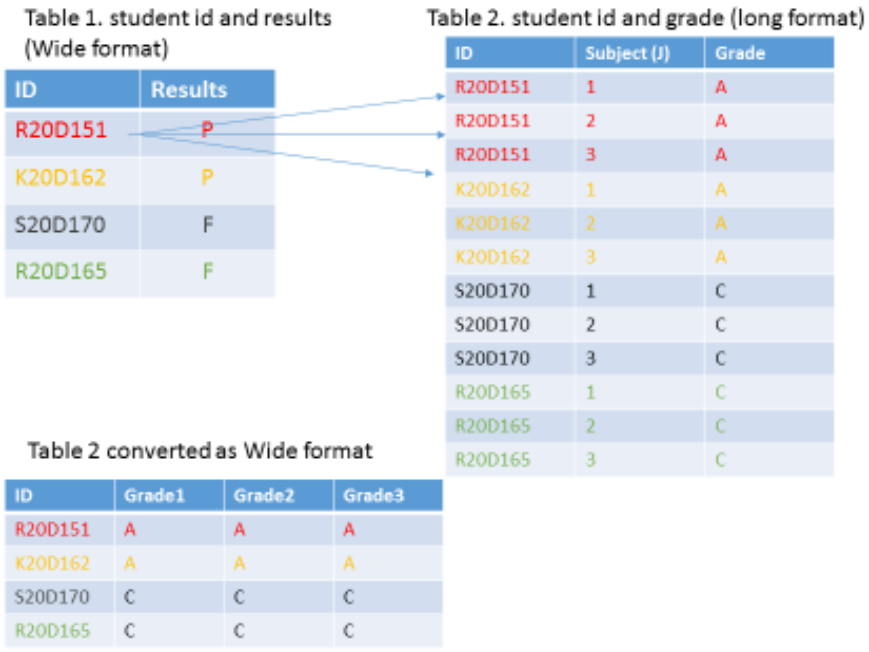


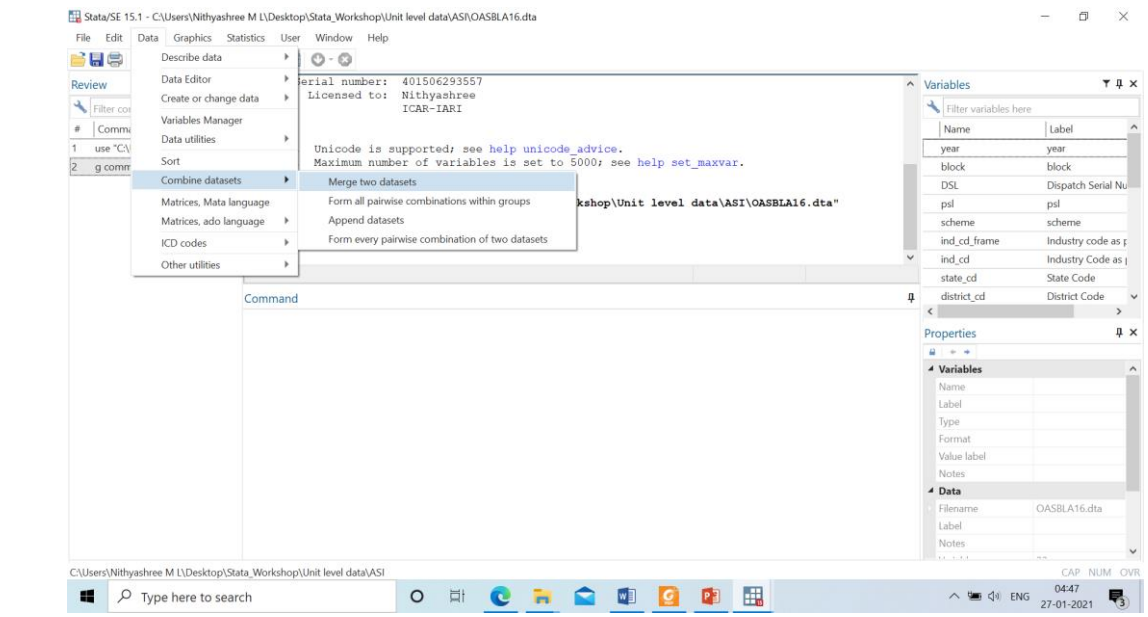
Figure 2. Visualization of wide and long format data

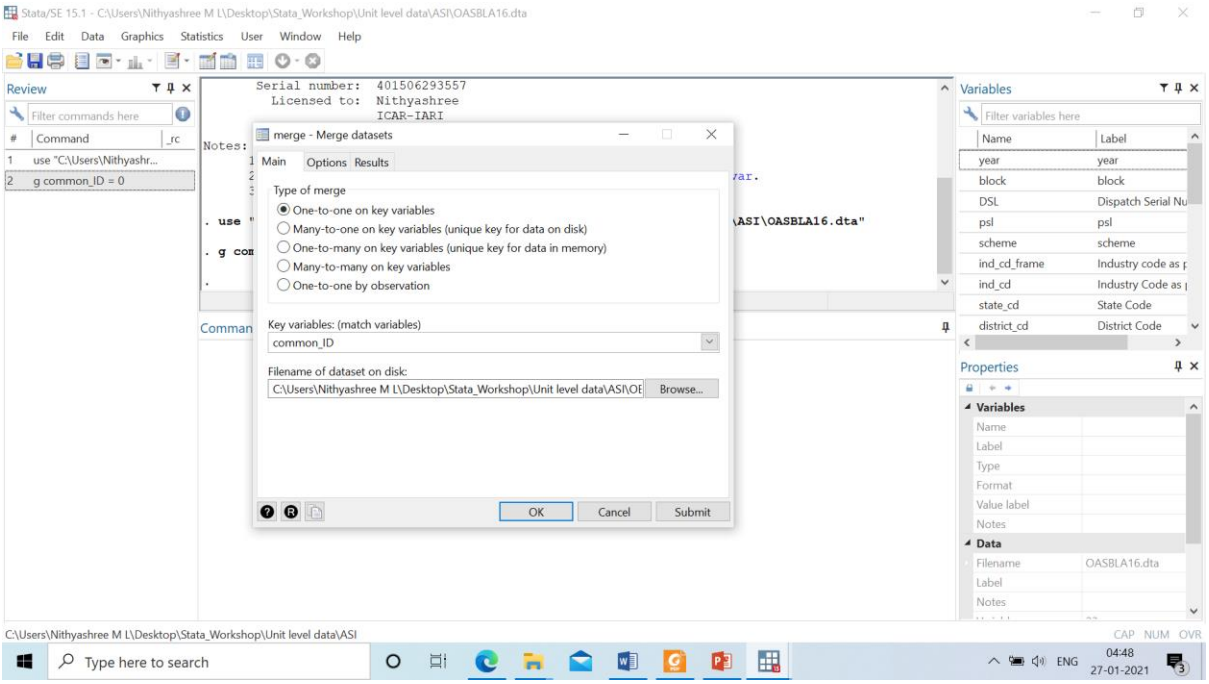
Similarly, to the unit-level data set command can be written as:

NSSO
reshape wide CropCode- PreHarvestSaleValue, i(id) j(SerialNo)

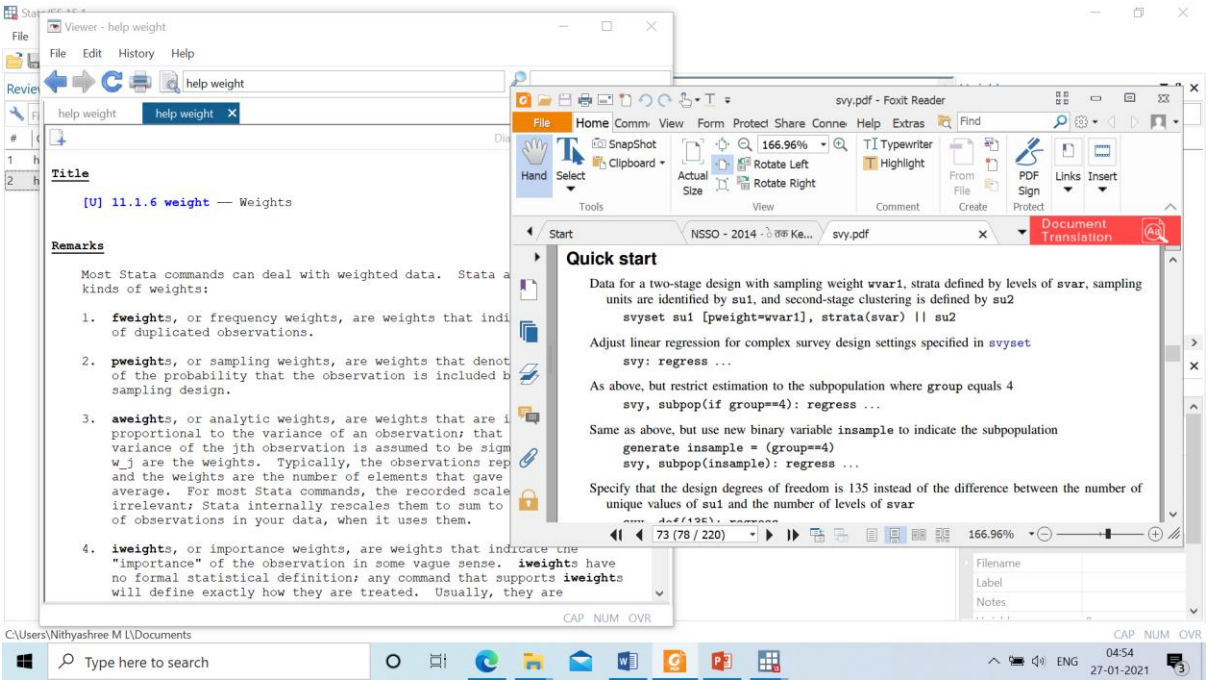
ASI
reshape wide GrossValueOpening - NVC , i(DSL) j(S_no)

In the next step, data files can be merged by using the common ID with the merge option one-to-one on key variable, as shown below





Based on the need for interpretation, multiplier options can be further explored by using the help command as shown below.



The detailed syntax for the ASI survey is given below keeping students in mind, which will be to practice reshaping and merging.

```
      name:  <unnamed>
       log:  F:\MOSPI_ASI_UNIT\2015-16\Data\M_1516.smcl
  log type:  smcl
 opened on:  21 Oct 2019, 12:54:11

. save "F:\MOSPI_ASI_UNIT\2015-16\Data\B.dta"
file F:\MOSPI_ASI_UNIT\2015-16\Data\B.dta saved

. use "F:\MOSPI_ASI_UNIT\2015-16\Data\blkC201516.dta"

. drop Year

. tab S_no
```

S.no	Freq.	Percent	Cum.
1	36,930	9.37	9.37
2	46,763	11.87	21.24
3	52,954	13.44	34.69
4	42,085	10.68	45.37
5	40,209	10.21	55.57
6	3,532	0.90	56.47
7	51,758	13.14	69.61
8	54,993	13.96	83.57
9	9,008	2.29	85.85
10	55,727	14.15	100.00
Total	393,959	100.00	

```
. reshape wide GrossValueOpening Gross_ValueaddduetoRevaluation G_ValueActuala
> ddition G_Valuedepadj G_Valueclose Depuotobeginning Depprovideduringtheyear
> Depadjustment Depuptoyearend N_V_O NVC , i( DSL ) j( S_no )
(note: j = 1 2 3 4 5 6 7 8 9 10)

Data                               long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                   393959   ->   55727
Number of variables                  14   ->     112
j variable (10 values)             S_no   ->   (dropped)
xij variables:
                     GrossValueOpening   ->   GrossValueOpening1 GrossValueOp
> ening2 ... GrossValueOpening10
          Gross_ValueaddduetoRevaluation   ->   Gross_ValueaddduetoRevaluation1
>  Gross_ValueaddduetoRevaluation2 ... Gross_ValueaddduetoRevaluation10
                  G_ValueActualaddition   ->   G_ValueActualaddition1 G_ValueA
> ctualaddition2 ... G_ValueActualaddition10
                        G_Valuedepadj   ->   G_Valuedepadj1 G_Valuedepadj2 .
> .. G_Valuedepadj10
                         G_Valueclose   ->   G_Valueclose1 G_Valueclose2 ...
>  G_Valueclose10
                     Depuotobeginning   ->   Depuotobeginning1 Depuotobeginn
> ing2 ... Depuotobeginning10
                 Depprovideduringtheyear   ->   Depprovideduringtheyear1 Deppro
> videduringtheyear2 ... Depprovideduringtheyear10
                         Depadjustment   ->   Depadjustment1 Depadjustment2 .
> .. Depadjustment10
                        Depuptoyearend   ->   Depuptoyearend1 Depuptoyearend2
>  ... Depuptoyearend10
                                 N_V_O   ->   N_V_O1 N_V_O2 ... N_V_O10
                                   NVC   ->   NVC1 NVC2 ... NVC10
-----------------------------------------------------------------------------

. save "F:\MOSPI_ASI_UNIT\2015-16\Data\C.dta"
\ file F:\MOSPI_ASI_UNIT\2015-16\Data\C.dta saved

. use "F:\MOSPI_ASI_UNIT\2015-16\Data\blkD201516.dta"

. drop Year

. tab S_No
```

S.No	Freq.	Percent	Cum.
1	43,434	6.24	6.24
2	7,693	1.10	7.34
3	21,317	3.06	10.40
4	46,962	6.74	17.14
5	21,202	3.04	20.19
6	34,421	4.94	25.13
7	48,998	7.03	32.16
8	53,187	7.64	39.80
9	48,309	6.94	46.73
10	47,374	6.80	53.54
11	54,010	7.75	61.29
12	47,454	6.81	68.10
13	29,978	4.30	72.41
14	48,651	6.98	79.39
15	51,354	7.37	86.76
16	54,031	7.76	94.52
17	38,157	5.48	100.00
Total	696,532	100.00	

```
. reshape wide OpenungRs ClosingRs , i( DSL ) j( S_No )
(note: j = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17)

Data                                  long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                      696532   ->   54047
Number of variables                      5   ->      36
j variable (17 values)               S_No   ->   (dropped)
xij variables:
                                OpenungRs   ->   OpenungRs1 OpenungRs2 ... Openu
> ngRs17
                                ClosingRs   ->   ClosingRs1 ClosingRs2 ... Closi
> ngRs17
-----------------------------------------------------------------------------

. save "F:\MOSPI_ASI_UNIT\2015-16\Data\D.dta"
file F:\MOSPI_ASI_UNIT\2015-16\Data\D.dta saved



. use "F:\MOSPI_ASI_UNIT\2015-16\Data\blkE201516.dta"

. br

. drop Year

. reshape wide MandaysWorkedManuf MandaysWorkedNonManuf MandaysWorkedTotal Ave
> NumberPersonwork NoofMandayspaid WagessalariesRs , i( DSL ) j( S_No )
(note: j = 1 2 3 4 5 6 7 8 9)

Data                                  long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                      338475   ->   54346
Number of variables                      9   ->      56
j variable (9 values)                S_No   ->   (dropped)
xij variables:
                     MandaysWorkedManuf   ->   MandaysWorkedManuf1 MandaysWork
> edManuf2 ... MandaysWorkedManuf9
                  MandaysWorkedNonManuf   ->   MandaysWorkedNonManuf1 MandaysW
> orkedNonManuf2 ... MandaysWorkedNonManuf9
                     MandaysWorkedTotal   ->   MandaysWorkedTotal1 MandaysWork
> edTotal2 ... MandaysWorkedTotal9
                   AveNumberPersonwork   ->   AveNumberPersonwork1 AveNumberP
> ersonwork2 ... AveNumberPersonwork9
                        NoofMandayspaid   ->   NoofMandayspaid1 NoofMandayspai
> d2 ... NoofMandayspaid9
                        WagessalariesRs   ->   WagessalariesRs1 WagessalariesR
> s2 ... WagessalariesRs9
-----------------------------------------------------------------------------
```

```
. reshape wide ItemCode Unit_Quantity_code QtyCons Purchase_Value Rate_PerUnit
>  , i( DSL ) j( Sno )
(note: j = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 2
> 6 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
>  52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
> 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101
>  102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
>  121 122 123 124 125 126 127 128 129 130 131 132 133)

Data                                  long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                      541009   ->   53406
Number of variables                      9   ->     668
j variable (133 values)                Sno   ->   (dropped)
xij variables:
                                   ItemCode   ->   ItemCode1 ItemCode2 ... ItemCod
> e133
                       Unit_Quantity_code   ->   Unit_Quantity_code1 Unit_Quanti
> ty_code2 ... Unit_Quantity_code133
                                   QtyCons   ->   QtyCons1 QtyCons2 ... QtyCons13
> 3
                            Purchase_Value   ->   Purchase_Value1 Purchase_Value2
>  ... Purchase_Value133
                              Rate_PerUnit   ->   Rate_PerUnit1 Rate_PerUnit2 ...
>  Rate_PerUnit133
-----------------------------------------------------------------------------
```

```
. reshape wide ItemCode Unit_Qty QtyCons Pur_value R_Perunit , i( DSL ) j( Sno
>  )
(note: j = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 2
> 6 27 28 29 30 31 32 33 34 35 36 37 38 39)

Data                                  long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                      29442   ->    8763
Number of variables                     8   ->     197
j variable (39 values)                Sno   ->   (dropped)
xij variables:
                                  ItemCode   ->   ItemCode1 ItemCode2 ... ItemCod
> e39
                                  Unit_Qty   ->   Unit_Qty1 Unit_Qty2 ... Unit_Qt
> y39
                                   QtyCons   ->   QtyCons1 QtyCons2 ... QtyCons39
                                 Pur_value   ->   Pur_value1 Pur_value2 ... Pur_v
> alue39
                                 R_Perunit   ->   R_Perunit1 R_Perunit2 ... R_Per
> unit39
-----------------------------------------------------------------------------
```

```
. reshape wide Item_code Unit_Qty Qty_Manuf Qty_Sold Gross_salevalue Excise_du
> ty Sales_taxVAT Others Subsidy Per_unit_Netsale_value Ex_FactvalOutput , i(
> DSL ) j( Sno )
(note: j = 1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18 19 20 21 22 23 24 25 26 2
> 7 28 29 30 31 32 33 34 35 36 37 38 39)

Data                                  long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                      127906   ->   43768
Number of variables                     15   ->     421
j variable (38 values)                 Sno   ->   (dropped)
xij variables:
                               Item_code   ->   Item_code1 Item_code2 ... Item_
> code39
                                Unit_Qty   ->   Unit_Qty1 Unit_Qty2 ... Unit_Qt
> y39
                               Qty_Manuf   ->   Qty_Manuf1 Qty_Manuf2 ... Qty_M
> anuf39
                                Qty_Sold   ->   Qty_Sold1 Qty_Sold2 ... Qty_Sol
> d39
                           Gross_salevalue   ->   Gross_salevalue1 Gross_salevalu
> e2 ... Gross_salevalue39
                             Excise_duty   ->   Excise_duty1 Excise_duty2 ... E
> xcise_duty39
                            Sales_taxVAT   ->   Sales_taxVAT1 Sales_taxVAT2 ...
>  Sales_taxVAT39
                                  Others   ->   Others1 Others2 ... Others39
                                 Subsidy   ->   Subsidy1 Subsidy2 ... Subsidy39
                  Per_unit_Netsale_value   ->   Per_unit_Netsale_value1 Per_uni
> t_Netsale_value2 ... Per_unit_Netsale_value39
                          Ex_FactvalOutput   ->   Ex_FactvalOutput1 Ex_FactvalOut
> put2 ... Ex_FactvalOutput39
-----------------------------------------------------------------------------

. save "F:\MOSPI_ASI_UNIT\2015-16\Data\J.dta"
file F:\MOSPI_ASI_UNIT\2015-16\Data\J.dta saved

. log close
      name:  <unnamed>
       log:  F:\MOSPI_ASI_UNIT\2015-16\Data\M_1516.smcl
  log type:  smcl
 closed on:  21 Oct 2019, 13:01:11
-----------------------------------------------------------------------------
```

References:

National Sample Survey Organization. (2014). *Key Indicators of Situation of Agricultural Households in India*, January – December 2013. NSS 70th Round. Ministry of Statistics and Programme Implementation, Government of India, New Delhi.

Annual Survey of Industries. (2015-16). *Summary results*, Ministry of Statistics and Programme Implementation, Government of India, New Delhi.

Applications of commodity outlook models for agricultural policy analysis

Shinoj Parappurathu

ICAR-Central Marine Fisheries Research Institute, Kochi
pshinoj@gmail.com

Agricultural Policy Analysis Using Commodity Outlook Models

Commodity Outlook models are partial equilibrium models which have found innumerous applications in the context of economic policy analysis in agriculture. Such models are used under various contexts; agricultural commodity market outlook models like IMPACT model of International Food Policy Research Institute (IFPRI), Rice outlook model of International Rice Research Institute (IRRI), World food model of Food and Agricultural Organization (FAO) are typical partial equilibrium models with the primary objective of generating short and medium term outlook of major food commodities. IMPACT model is a multipurpose model with the capability of simulating important policy variables under various alternative scenarios. Some other models like the World Agricultural Trade Simulation model (WATSim) of Food and Agricultural Policy Research Institute (FAPRI) is specifically designed to model international agricultural trade and related policy simulations. These models could be of different dimensions; some are multi-commodity multi region spatial models while some others are single commodity national models.

Partial Equilibrium Models: Meaning and Definition

A partial equilibrium is a type of economic equilibrium, wherein the clearance on the market of some specific goods is obtained independently from prices and quantities demanded and supplied in other markets. In other words, in such equilibrium, the prices of all substitutes and complements, as well as income levels of consumers are constant. Here, the dynamic process is such that prices adjust until supply equals demand. It is a powerfully simple technique that allows one to study equilibrium, efficiency and comparative statics. The stringency of the simplifying assumptions inherent in this approach make the model considerably more tractable, but may produce results which, while seemingly precise, do not effectively model real world economic phenomena[1]. In partial equilibrium analysis, the effects of policy actions are examined only in the markets that are directly affected. Supply and demand curves are used to depict the price effects of policies. Producer and consumer surplus is used to measure the welfare effects on participants in the market. A partial equilibrium analysis ignores effects on other industries in the economy or assumes that the sector in question is very, very small and therefore has little if any, impact on other sectors of the economy.

Partial Equilibrium *versus* General Equilibrium

While partial equilibrium analysis considers only a particular market or sector and the underlying demand and supply dynamics, general equilibrium seeks to explain the behavior of supply, demand and prices in a whole economy with several or many interacting markets, by seeking to prove that a set of prices exists that will result in an overall equilibrium. As with all

[1] http://en.wikipedia.org

models, this is an abstraction from a real economy; it is proposed as being a useful model, both by considering equilibrium prices as long-term prices and by considering actual prices as deviations from equilibrium. General equilibrium theory both studies economies using the model of equilibrium pricing and seeks to determine in which circumstances the assumptions of general equilibrium will hold. The theory dates to the 1870s, particularly the work of French economist Léon Walras. The distinction between partial and general equilibrium models can be made in terms of (a) *ceteris paribus* assumptions and (b) the variables of interest that are endogenous. At one extreme is the typical model of a commodity market that takes the price and quantity of that commodity as endogenous treating all of other goods as constant and exogenous to the analysis. A the other extreme, are the detailed economy-wide models in which all prices and quantities are endogenous to, and measured in the analysis, so that the extreme *mutatis mutandis* (everything allowed to change) replaces extreme *ceteris paribus*. Most economic analyses fall somewhere in between these two extremes. Another way of distinguishing is in terms of techniques of analysis. For instance, when Marshallian supply-and-demand models are used, the analysis is typically regarded as being a partial equilibrium analysis, whereas when a social accounting matrix (SAM) is involved, it is regarded as general equilibrium analysis (Alston et al, 1998).

Modeling approaches to Partial Equilibrium Analysis

A partial equilibrium modeling problem can be approached from different angles based on the kind of modeling framework used for solving it. A simple partial equilibrium model could consist of linear equations of demand and supply specified by an equilibrium condition as illustrated below;

Building the Model

- Demand equation(behavioral equation): $QD = a - bP$ $(a, b > 0)$
- Supply equation(behavioral equation): $QS = - c + dP$ $(c, d > 0)$
- Equilibrium Condition: $QD = QS$

Method of finding equilibrium

- Solution by eliminating variables

Non-linear models with quadratic terms or higher degree polynomial terms can be used to replace simple linear models when the situation demands detailed examination of the underlying market situation. Still, we have the specific condition for determining the existence of equilibrium with economic implications. In addition to algebraic approach, linear models can be solved using graphical approach also.

Structure of a Typical Agricultural Policy Model

A typical agricultural commodity outlook model under partial equilibrium framework consists of the following sub-components.

1. Producer core system
   (i) Area equation
   (ii) Yield equation
   (iii) Production equation

(iv) Supply equation

2. Consumer core system
   (i) Food demand equation
   (ii) Feed demand equation
   (iii) Other uses demand equation
   (iv) Total demand equation

3. Trade core system
   (i) Export equation
   (ii) Import equation
4. Price linkage equation
5. Model closure

The producer core system depicts the supply side of the commodity under question whereas the consumer core system depicts the demand side. Various demographic and conditional variables like research investment, irrigation, weather parameters like rainfall, temperature, other qualitative variables that determine the choice of consumers etc. can also be incorporated into both producer and consumer core systems as exogenous variables. The trade core system is inserted in the case of an open economy where the goods are traded outside the economy. The price linkage equations link the demand and supply sides with equilibrium conditions. In addition to this, a number of policy variables like tariffs, subsidies, and support prices can also be incorporated into these models exogenously to capture the effects of policy changes. The technical parameters of the various equations have to be estimated based on real data either time series, cross section or pooled. The accuracy of the model output would depend a great deal on how realistic these estimates are. With this basic structure, the model could have various sub-sectoral dimensions which can include crop sector, livestock sector, dairy sector, input sector and spatial dimensions that may vary from regional dimension, national dimension and global dimension depending upon the spatial coverage the modeler intends to incorporate. Both linear and non-linear programming approaches can be applied to derive optimum feasible solutions and various algorithms are available for the purpose of solving such models. Various software packages like SAS, GAMS, Microsoft spread sheet, etc. are enabled with features to construct commodity outlook models and solve them using alternative iterative procedures.

Commodity Outlook for Indian Agriculture: The case of Cereal Outlook Model

The Cereal Outlook Model was developed by the authors for generating commodity outlooks for three major cereals, viz., rice, wheat and maize in India. A detailed account of this model is available in Parappurathu, et al (2014a; 2014b). The model was constructed under a dynamic as well as spatial partial equilibrium modeling framework that incorporate a system of simultaneous equations for effectively depicting the linkages between various economic variables in the balance sheet of major cereals in India. The Model takes cognizance of the key economic variables such as production, demand, stocks, trade, prices and policy variables related to the primary commodities. It has sought to generate medium-and long-term projections, given the past trends in behavior of the variables in question as well as magnitude of technical coefficients which govern their behavior. Technically, the Model derives equilibrium values of the variables based on the econometric linkages established through a set of equations that cuts across commodity as well

as spatial dimensions. It is an open model as it takes into account the trade flows of the commodities with respect to the rest of the world and the endogenous prices are attached to the world market prices. The Model is dynamic in the sense that the current prices and quantities are related to the past prices and quantities and the equilibrium is attained through a dynamic recursive iterative process that continuously adjusts the quantities and prices across time periods till the overall model converges to an equilibrium state. Spatial dimensions have been incorporated by specifying supply side equations separately for six regions in the country.

Model Structure

The Cereal Outlook Model is a typical agricultural-related model that incorporates the major demand and supply side variables, output and input prices, as well as other exogenous variables like income and population; and policy variables like support prices, tariffs, etc. A schematic representation of the linkages in the model is shown in Figure 1.

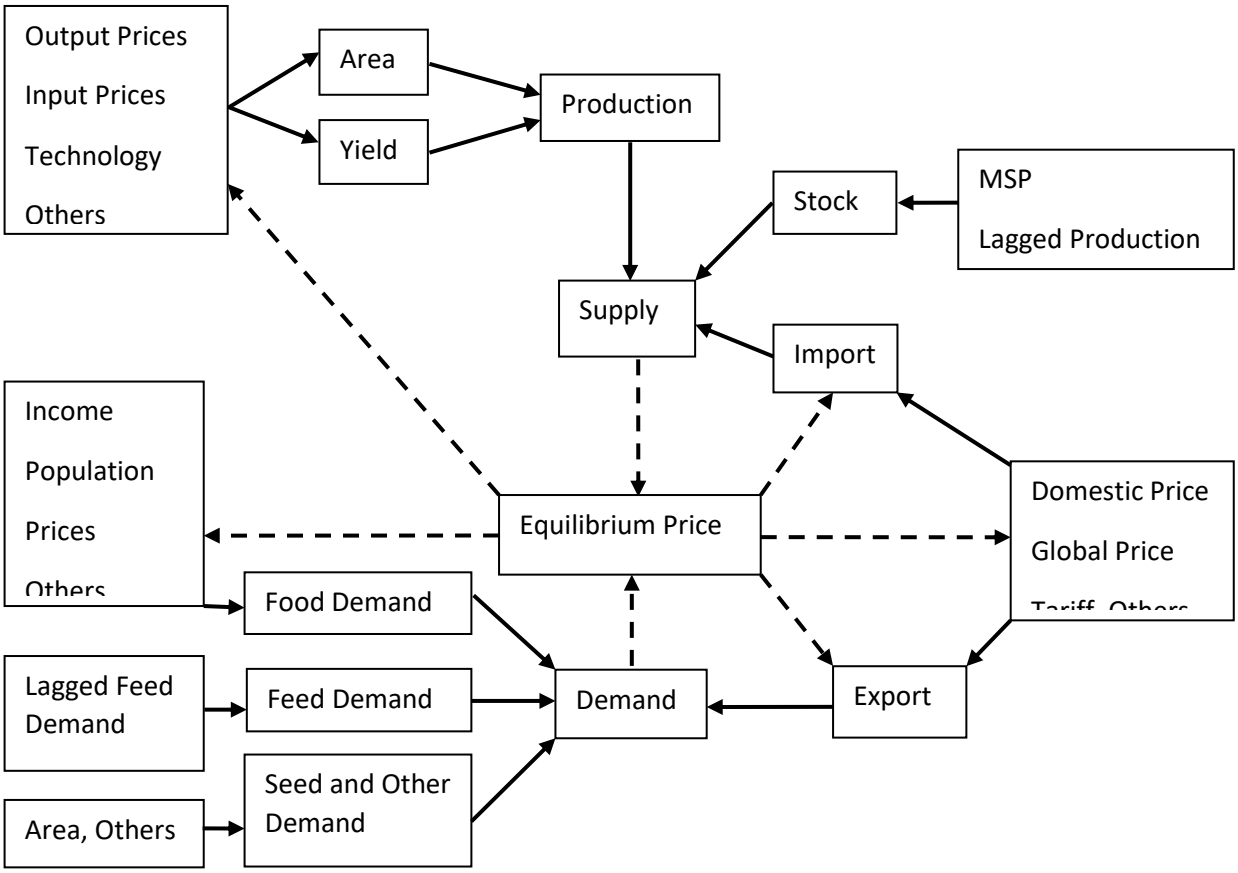


Figure 1. Modeling Framework of *Cereal Outlook Model*: An Illustration
Source: Parappurathu *et al* (2014a)

Broadly, the Model comprises of the following integral components: (i) a producer core system that integrates the linkages between area, yield, production, stock changes and supply of the individual grains; (ii) a consumer core system that includes the equations for food and other uses demand, feed demand and total demand; (iii) a trade core system that incorporates the export and import equations; (iv) a set of price linkages equations for depicting the relationships between producer prices with consumer prices and national and

regional prices; and (v) a model closure equation that links the various cores of the model with certain closure conditions.

However, all such partial equilibrium models would be essentially bound by some basic economic assumptions like perfect competition, constant returns to scale etc. which can be relaxed to some extent depending upon circumstances. The utility of such models varies depending upon the way each of the sub-systems are modeled based on the discretion of the modeler and the purpose for which it is built; whether for forecasting, policy simulations or baseline situation assessment.

References

Alston J.M., Norton G.W. and Pardey P.G. (1998). Science under scarcity. CAB International.

Goletti, F. and Rich, K. 1998. "Policy Simulation for Agricultural Diversification." Report prepared for the UNDP project on "Strengthening Capacity Building for Rural Development in Viet Nam", Washington, D.C.: IFPRI.

Minot, N. (2009). Using GAMS for Agricultural Policy Analysis, Technical Guide, International Food Policy Research Institute, Washington, D.C.

Roningen, V.O. 1997. "Multi-Market, Multi-Region Partial Equilibrium Modeling" In Applied Methods for Trade Policy Analysis: A Handbook, J.F. Francois and K.A. Reinert, eds. (Cambridge: Cambridge University Press), pp. 231-257.

Parappurathu, S., Kumar, A., Kumar, S. and Jain, R. (2014a). A partial equilibrium model for future outlooks on major cereals in India, *Margin-The Journal of Applied Economic Research*, 8 (2): 155-192.

Parappurathu, S., Kumar, A., Kumar, S. and Jain, R. (2014b). Commodity outlook on major cereals in India, Policy Paper No. 28, National Centre for Agricultural Economics and Policy Research, New Delhi.

Sadoulet, E., and deJanvry, A. 1995. Quantitative Development Policy Analysis (Baltimore: Johns Hopkins University Press), chapter 11.

Estimating Demand & Supply Systems: A Methodological Note

Parmod Kumar

Giri Institute of Development Studies,Lucknow

Pkumar@gids.org.in

Food Security exists when all people at all times have physical, social and economic access to sufficient safe and nutritious food which meets their dietary needs and food preferences for an active and healthy life (FAO 2002). Both demand and supply side factors are important for food Security. Three dimensions of food security are;

I. Availability of food – This includes supply side factors such as production/imports/stock
II. Accessibility – It means ensuring food within the reach of people. And it is determined by distribution factors
III. Affordability – It means that the people may have sufficient purchasing power or the food may be affordable for the buyers. It is determined by demand side factors

Food Insecurity may be a temporary/transient phenomenon or it may be persistent/regular in nature.

Recent Policies and programmes for ensuring food security in India

Demand side policies–

- National Food Security Act / Public Distribution System (PDS, RPDS, TPDS)
- Mid-Day Meal Scheme (MDM)
- Integrated Child Development Scheme (ICDS)

Supply side policies-

- National Food Security Mission (NFSM)
- Village Grain Bank Scheme
- Rashtriya Krishi Vikas Yojana (RKVY)
- Employment / Ensuring Purchasing Power
- Food for Work Programme
- Mahatma Gandhi National Rural Employment Guarantee Scheme

Basic properties of a demand model:

Adding-up: Budget shares sum to one, i.e., total expenditure must be exhausted among individual commodities

Homogeneity condition: Demand functions are homogenous of degree zero in prices and income.

Symmetry condition: Slutsky's symmetry condition implies that the compensated cross price elasticities are equal.

Negativity condition: Compensated demand curve is downward sloping or the substitution effect is always negative implying that the 'law of demand' holds.

The Almost Ideal Demand System (AIDS)

The AIDS model developed by Deaton and Muellbauer (1980) derive demand system by use of duality concepts, from a particular cost or expenditure function. The AIDS cost function is given by

$$ln\ e(u,p) = \alpha_0 + \sum_k \alpha_k \ln p_k + \frac{1}{2}\sum_k \cdot \sum_j \cdot \gamma^*_{kj} \ln p_k \ln p_j + u\beta_0 \pi_k p_k^{\beta k}$$

Where α_0 ,α_i , β_i and γij, are parameters, u is utility and pj 's are prices. AIDS model satisfies the axioms of choice exactly, and does not impose additive preferences and under certain conditions, allows consistent aggregation of individual demands to market demands. By applying Shepard's Lemma, that is, by differentiating the cost function, after appropriate substitutions, we obtain the AIDS in budget share form

$$w_i = \alpha_i + \sum_j \cdot \gamma_{ij} \ln p_j + \beta_i \ln \frac{x}{P}$$

Where P is price index defined by

$$ln\ P = \alpha_0 + \sum_k \alpha_k \ln p_k + \frac{1}{2}\sum_k \cdot \sum_j \cdot \gamma^*_{kj} \ln p_k \ln p_j$$

With the above price index, the system becomes non-linear and requires estimation of a large number of parameters that needs a long data series on budget shares, total expenditure and prices. Deaton and Muellbauer (1980a) used Stone Price Index to make the system linear by substituting ln P* = Σ wk ln pk ; where P ~= P*

Parameter Restrictions

Adding up: $\sum_i \alpha_i = 1; \sum_j \gamma_{ij} = 0; and \sum_i \beta_i = 0;$

Homogeneity: Σyij = 0;

Symmetry: Yij = Yji

The fourth restriction (concavity of the expenditure function) has no parametric representation.

Demographic representation:

Following Heien and Pompelli (1988), demographic effects are incorporated in the model by allowing the intercept to be a function of demographic variables as:

$$\alpha_i = \rho_{i0} + \sum_{j=1} \rho_{ij} d_j$$

Where dj is the j th demographic variable, the examples could be household size, region, gender, education level of household head, etc.

Two Stage Budgeting

A two-stage budgeting demand system: It is supposed that the consumers allocate their total budget in two stages. In the first stage, the consumer determines the allocation of his total expenditure between various commodity groupings, e.g., cereals, pulses, edible oils, milk-meat, fruit-vegetables, other food items and non-food items. In the second stage, the expenditure is allocated among different groups, e.g., wheat, rice and coarse grains among the cereal group; fruit, vegetables and nuts among fruit-vegetables group and so on.LA/AIDS model used to estimate the two-stage budgeting demand function.

Elasticity Formulation

Income $\eta_r = 1 + \frac{\beta_r}{w_r}$ elasticity:

Own-price and cross-price elasticities: $\varepsilon_{rs} = \frac{\gamma_{rs} - \beta_r w_s}{w_r} - \delta_{rs}$

Where δ_{rs} is Kronecker's delta which is equal to unity for every s=r, and zero otherwise

Data required

Consumption: All-India average monthly per capita consumer expenditure on food and non-food commodities and the corresponding per capita total consumer expenditure (PCE), separately for rural and urban households. Unit level data of the National Sample Survey Organization (NSSO) for its 43rd to 68th rounds for 'Household Consumption Expenditure'. The time period consists of a continuous time series beginning from 1987-88 (43rd round) up to 2011-12 (68th round). Model estimated at the aggregate and three population groups based on ranking by the level of PCE, viz., the poorest one fourth (bottom 25 percent), the middle half (middle 50 percent) and the upper one fourth (top 25 percent), separately for the rural and urban sector.

Prices: The wholesale price indices published by the OEA. The price indices for the seven broad groups of items as well as for the individual items are prepared at the base year 2004-05. The price indices of all individual items for the whole time period (1987-88 to 2011-12) are first converted into 2004-05 prices (2004-05=100) through splicing. The individual items in the converted monthly price indices are matched with the items within each of the seven broad groups of the consumer expenditure to prepare wholesale price relatives in each group. The monthly wholesale price relatives in each group are then averaged over the months covered by a NSS round to obtain the average price relative for that round in each group. Using the expenditure weights for individual items in a broad group relative to the total expenditure on the broad group, separately for rural and urban areas, the price index relative for individual and broad group items are constructed for both rural and urban sectors. The expenditure weights are worked out on the basis of estimated average consumer expenditure on individual items in each group, separately for rural and urban areas using the NSS 62nd (2004-05) round consumer expenditure per capita data.

Model Estimation

The LA/AIDS model can be estimated using non-linear seemingly unrelated regression method without and with imposing restriction of homogeneity, adding-up and symmetry. The commodity combinations are: cereals - wheat, rice and coarse cereals; pulses edible oils milk - meat - milk and meat; fruits and vegetables - fruits, vegetables and dry-nuts; other food – sugar, spices, beverages and intoxicants non-food.

Assumptions for Demand Projections

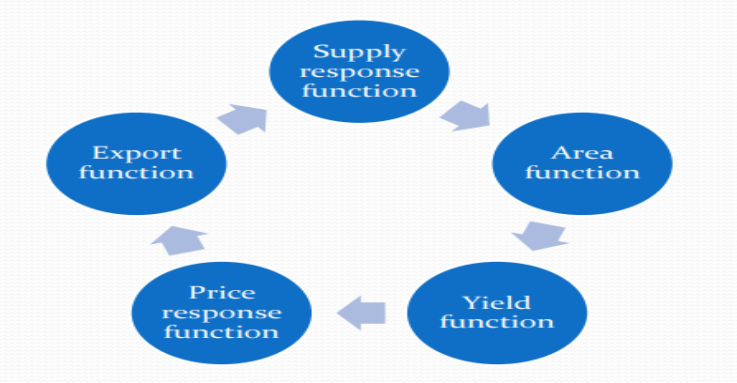
India's decadal population growth rate - 1.64% PA (2001- 2011), projected growth 1.32% PA in 2011-2021 and 1.13% in 2021-2031. Urban share expected to increase from 31.1% in 2011 to 39.1% in 2031. Contribution of rural sector in total GDP expected to come down from 47% in 2011-12 to 44% in 2020-21 and 40% by 2032-33. GDP growth during current year -4.5% and 6% during 2021-22 (World Bank/IMF). From 2022-23 to 2032-33 – Two scenarios (i) 6 % and (ii) 8% per annum Corresponding PCI growth during 2022-23 onwards would be (i) 4.87% and (ii) 6.87%. For current year it would be -5.8% and next year 4.73%.

Demand Projections – Formula

$$Q_{ijt} = q_{ij0} * P_{jt} * (1 + g_{jt} * e_{ij})^t$$

Where Q_{ijt} is household demand for i th commodity for the j th sub group (rural or urban) during the tth time period; q_{ij0} is the annual per capita quantity consumed of i th commodity by the j th sub group during the base year (2004-05); P_{jt} is the projected population of j th sub group in the year t; g_{jt} is annual growth rate in per capita income for the j th sub group during the tth time period; and e_{ij} is the expenditure elasticity of the i th commodity for the j th sub group

Estimation of Supply System: Simultaneous model



Model Specification

Area = f (Pi , Pj , Rain, Irrg, Fert, Trend, Lagged)

Yield = f (Pi , Rain, Irrg, Fert, Trend, Lagged)

Real FHPi = f (MSP, WP, Prod, WI, PD, Trend, Lagged)

Exports = f (Pi , WP, Prod, Open, PD, WT, WI, REER, Trend, Lagged)

Where, Pi is real domestic (farm harvest) price; Pj is real competing crop price; Rain is rainfall - annual, monsoon or winter months as applicable in different cases; Irrg is percentage of area under irrigation, Fert is fertilizer use in kgs per hectare; Fertp is real fertilizer price; MSP is real minimum support price; WP is real world price or real unit value of exports; WI is real world income; PD is policy dummy; Open is openness in terms of share of Indian exports in the world exports commodity wise; WT is volume of world trade in a particular commodity. REER is real effective exchange rate.

Specification of Crops (group of crops)

The specific crops are: rice, wheat, kharif and rabi coarse cereals, kharif and rabi pulses and kharif and rabi oilseeds.

Crop sub-groups are:

Kharif Coarse Cereals: Jowar, Bajra, Ragi and Maize; Rabi Coarse Cereals: Barley; Kharif Pulses: Tur, Moong, Urad; Rabi Pulses: Gram and Masoor (Lentil); Kharif Oilseeds: Groundnut, Soyabeans, Sunflower, Sesamum, Nigerseed; Rabi Oilseeds: Rapeseed & Mustard, Safflower and Linseed

Assumptions for Supply Projections Growth trends of determinants (up to 2020)

		Comp crop price	Fertlizer use	MSP	Irrigation	WPI	Exchange rate	World price $	Fertilizer price	World gdp deflator	World income $	Openness	World trade
Wheat	Per I	0.089	0.048	0.072	0.015	0.077	0.102	-0.010	0.036	0.040	0.031	-0.021	0.005
	Per II	0.050	0.028	0.042	0.008	0.048	0.026	0.002	0.031	0.029	0.029	0.027	0.010
Rice	Per I	0.100	0.060	0.079	0.013	0.088	0.102	-0.013	0.000	0.040	0.031	-0.021	-
	Per II	0.038	0.030	0.043	0.003	0.048	0.019	-0.010	-0.039	0.029	0.029	0.027	-
Kharif coarse cereals	Per I	0.097	0.069	0.069	0.018	0.088	0.102	-0.012	0.038	0.040	0.031	-0.021	-0.004
	Per II	0.038	0.019	0.054	0.002	0.048	0.019	-0.002	0.031	0.029	0.029	0.027	-0.004
Rabi coarse cereals	Per I	0.082	0.052	0.071	0.019	0.077	0.102	0.006	0.036	0.040	0.031	-0.021	-
	Per II	0.045	0.012	0.058	0.013	0.048	0.026	0.010	0.033	0.029	0.029	0.027	-
Kharif pulses	Per I	0.087	0.075	0.097	0.065	0.088	0.102	0.011	0.039	0.040	0.031	-0.021	-0.266
	Per II	0.038	0.027	0.072	0.002	0.048	0.019	0.002	0.029	0.029	0.029	0.027	0.071
Rabi pulses	Per I	0.080	0.063	0.086	0.031	0.077	0.102	0.116	0.038	0.040	0.031	-0.021	-0.266
	Per II	0.042	0.035	0.070	0.034	0.048	0.026	0.005	0.029	0.029	0.029	0.027	0.071
Kharif oilseed	Per I	0.100	0.062	0.090	0.024	0.088	0.102	-0.023	0.039	0.040	0.031	-0.021	0.007
	Per II	0.037	0.018	0.043	0.009	0.048	0.019	0.008	0.029	0.029	0.029	0.027	0.050
Rabi oilseed	Per I	0.081	0.041	0.069	0.028	0.077	0.102	-0.003	0.038	0.040	0.031	-0.021	0.059
	Per II	0.050	0.031	0.076	0.002	0.048	0.026	0.033	0.031	0.029	0.029	0.027	0.024

Expenditure elasticity

Commodity	Rural	Urban
Cereals	-0.13	-0.04
Pulses	0.55	0.36
Edible oils	0.88	0.37
Milk	0.82	0.40
Meat	0.82	0.40
Fruits, Vegetables & Nuts	0.85	0.42

Average demand for SFW and other industrial uses (million tonnes)

- Foodgrains use as SFWI to increase from 34 million tonnes in the early 1990s to around 76 million tonnes up to 2015-16.
- It is expected to cross 95 million tones by the end of 2025-26 and above 120 million tones by 2032-33.
- The ratio of SFW to total foodgrains' demand would increase from less than 25 percent in TE 2012-13 to around 40 percent by 2032-33

Supply Elasticity – Crop Area

	FHP	Competing Price	Rainfall	Irrigation	Time trend / Lag Dep
Wheat	0.10	-0.05	0.15	0.28	0.69
	(1.9)	-(1.6)	(3.7)	(2.7)	(6.7)
Rice	0.10	-0.10	0.19	0.17	0.47
	(1.4)	-(2.3)	(5.8)	(4.1)	(3.5)
CC_kh	0.05	-0.26	0.23	-	0.59
	(0.9)	-(2.7)	(3.4)	-	(5.4)
CC_rb	0.11	-0.22	0.04	-0.50	0.00
	(0.9)	-(2.0)	(1.6)	-(1.8)	-(0.4)
Pul_kh	0.19	-0.05	0.16	0.04	0.74
	(2.2)	-(0.3)	(1.4)	(0.9)	(5.0)
Pul_rb	0.21	-0.04	0.20	-0.40	0.23
	(4.7)	-(0.5)	(2.9)	-(4.9)	(2.1)
Oil_kh	0.14	-0.19	0.12	-0.06	0.71
	(1.6)	-(2.7)	(1.7)	-(0.6)	(6.3)
Oil_rb	0.23	-0.21	0.12	0.18	0.73
	(2.3)	-(2.0)	(2.7)	(1.7)	(8.2)

Supply Elasticity – Yield

	FHP	Rainfall	Irrigation	Fertilizer	Time/Lag dep
Wheat	0.21	0.14	1.59	0.02	0.09
	(3.7)	(2.6)	(4.2)	(0.3)	(0.6)
Rice	0.08	0.27	0.02	0.18	0.01
	(1.0)	(4.3)	(0.1)	(2.9)	(3.1)
CC_kh	0.63	0.49	-0.21	0.53	0.15
	(5.4)	(3.6)	-(2.0)	(6.0)	(1.2)
CC_rb	0.73	0.03	0.57	0.21	0.28
	(3.2)	(0.6)	(1.9)	(1.7)	(1.7)
Pul_kh	0.25	0.55	-	-0.14	-0.04
	(2.9)	(4.2)	-	-(2.2)	-(0.3)
Pul_rb	0.10	0.02	0.04	-0.12	0.00
	(1.5)	(0.6)	(0.3)	-(1.0)	-(0.1)
Oil_kh	0.08	0.92	-0.14	-	0.02
	(0.5)	(6.4)	-(0.6)	-	(7.3)
Oil_rb	0.34	-0.02	0.77	0.24	-0.01
	(4.0)	-(0.8)	(4.0)	(3.3)	-(0.1)

Supply Elasticity – Farm Harvest Price

	MSP	World Price	Domestic Production	Time/Lag dep
Wheat	0.96	-0.03	0.02	-
	(10.0)	-(0.8)	(0.1)	-
Rice	0.77	0.06	-0.09	-
	(10.3)	(1.5)	-(0.6)	-
CC_kh	0.87	0.17	-0.22	0.00
	(8.4)	(2.7)	-(1.7)	(0.9)
CC_rb	0.42	-0.06	0.49	-
	(2.2)	-(0.8)	(4.8)	-
Pul_kh	0.67	0.03	-0.29	-
	(3.7)	(2.5)	-(1.8)	-
Pul_rb	0.91	0.02	-0.29	0.00
	(4.3)	(0.7)	-(1.4)	-(0.5)
Oil_kh	0.12	0.22	-0.31	0.02
	(0.7)	(2.4)	-(2.3)	(3.0)
Oil_rb	0.00	0.03	-0.35	0.01
	(0.1)	(0.4)	-(3.5)	(3.2)

Supply Elasticity – Exports

	World Price / UVE	FHP	Domestic Production	Openness index	Time/Lag dep
Wheat	2.17	-11.93	-1.85	-9.03	0.40
	(4.6)	-(4.0)	-(0.5)	-(3.6)	(3.6)
Rice	0.27	-0.65	0.92	0.29	0.53
	(0.9)	-(1.8)	(0.9)	(0.8)	(4.7)
CC_kh	1.09	-0.13	0.87	0.96	0.67
	(1.7)	-(0.1)	(0.6)	(1.1)	(9.8)
CC_rb	3.16	0.59	2.86	2.10	0.59
	(2.3)	(0.4)	(1.5)	(1.3)	(4.7)
Pul_kh	0.40	-0.99	-0.70	-	0.72
	(2.2)	-(1.1)	-(0.5)	-	(6.9)
Pul_rb	1.19	-3.26	-0.33	-	-0.12
	(3.9)	-(1.7)	-(0.1)	-	-(1.3)
Oil_kh	0.92	-1.40	-0.06	1.28	0.08
	(2.5)	-(3.5)	-(0.1)	(3.0)	(0.6)
Oil_rb	0.91	-4.20	-	0.63	0.24
	(0.6)	-(3.9)	-	(0.4)	(1.5)

Net balances of supply and demand of food grains by 2032-33

Demand Estimates – Food grains: 331-346 Million Tonnes

Milk: 300-355 Million Tonnes

Horticultural commodities: 585-696 MT

Supply Estimates – Food grains: 367-375 Million Tonnes

Milk: 326-33 Million Tonnes

Horticultural commodities: 626-675 MT

Final Estimates of Demand(MT)

	2011-12	2019-20	2020-21	2021-22	2022-23	2027-28	2032-33
Rice	94	107	113	108	108	112	117
Wheat	90	94	101	96	96	102	109
Coarse cereals	38	43	47	45	47	55	65
Cereals	222	244	261	248	252	270	292
Pulses	20	25	21	27	30	35	41
Foodgrain	242	269	282	276	282	305	333
Edible oil (without palm)	10	16	10	19	21	28	36
Oilseeds	34	56	37	67	76	99	130
Milk and products	119	161	131	186	204	258	328
Eggs, meat and fish	12	16	11	19	22	29	41
Vegetables	152	197	147	231	256	327	418
Fruits	88	114	105	128	138	173	217
Nuts	1.64	2.12	1.32	2.55	2.89	3.72	4.80
FVN	242	313	253	361	397	503	641

Final Estimates of Supply (MT)

	2017-18	2018-19	2020-21	2028-29	2032-33
Rice	112.8	116.5	121.3	143.3	156.1
Wheat	99.9	103.6	110.8	119.5	120.1
C. Cereals	47.0	43.1	51.5	57.2	59.6
Pulses	25.4	22.1	23.8	30.5	35.6
Food grains	285.0	285.2	307.5	350.5	371.5
Oilseeds	31.5	31.52	35.5	44.5	49.7
Sugarcane	379.9	400.2	403.8	466.6	501.6
Cotton	32.8	28.7	45.7	83.4	113.9
Jute & mesta	10.0	9.8	11.0	11.9	12.4
Fruit	97.4	98.6	115.3	167.8	202.7
Vegetables	184.4	185.9	211.3	303.0	362.9
Total Horticult	311.7	313.9	365.1	541.4	659.4
Milk	176.3	187.7	194.0	276.3	329.7
Eggs	95.2	103.3	109.9	172.4	215.9
Wool	41.5	40.4	48.7	49.7	50.2
Meat	7.7	8.1	11.9	27.7	42.3
Fish	12.6	13.4	13.7	19.6	23.6

Conclusion

The balance sheet of demand and supply looks quite affirmative for food grains except the case of oilseeds in which case India is already in acute deficit.India is poised for some surplus in rice and wheat while coarse grains will be sufficient to meet the domestic demand whereas pulses will have deficit of around 4 to 6 million tonnes.The overall foodgrains will have quite comfortable position as far as food security is concerned. In oilseeds a massive deficit of more than 50 million tonnes will appear which was worked out without including the imported palm oil and unless India succeeds in achieving nothing less than second Yellow Revolution either through technological breakthrough somewhat similar to BT Cotton, or there is massive expansion in area under oilseeds along with transformation achieved in the yield rate. There is huge possibility of expanding area under oil palm which has much better yield rate compared to oilseed field crops. In other commodities including milk, meat, fruits and vegetables and sugar there appears to be fine balance between demand and supply given the assumptions on supply side turns true.

Univariate Time Series Analysis: ARIMA and G(ARCH) Models

Achal Lama[1], Girish K Jha[2] and Asha Devi[2]

[1]ICAR-Indian Agricultural Statistics Research Institute;[2]ICAR-Indian Agricultural Research Institute-New Delhi

girish.stat@gmail.com

A time series (TS) is a collection of observations on a quantitative characteristic of a phenomenon observed sequentially in time. Here we are interested on those observations which are collected at equally spaced as well as at discrete time intervals, may be collected hourly, daily, weekly, monthly, or yearly, and so on. For example, daily maximum temperatures, weekly agricultural price data, monthly sales, yearly gross national product, annual crop production, etc. A basic assumption in any TS analysis is that some aspects of the past pattern will continue into the future. Hence, dependency through time is used for extrapolation in the future. The notation such as $\{X_t\}$ or $\{Y_t\}$ (t =1,…,T) is used to denote a time series of length T. The goals of time series models include smoothing irregular series, forecasting series into the medium or long term future and causal modelling of variables moving in parallel through time.

Time series methods

Time series methods use different statistical methods to treat the time series data approximately to draw inferences. These models may be univariate, i.e., modelling of single series of data or multivariate that includes multiple series of data containing different variables. Besides, non-linear time series techniques are also popular nowadays. Here it is tacitly assumed that information about the past is available in the form of numerical data. Ideally, at least 50 observations are necessary for performing TS analysis/ modelling, as propounded by Box and Jenkins who were pioneers in TS modelling.

Decomposition models are among the oldest approaches to TS analysis *albeit* a number of theoretical weaknesses from a statistical point of view. These were followed by the crudest form of forecasting methods called the moving averages method. As an improvement over this method which had equal weights, exponential smoothing methods came into being which gave more weights to recent data. Exponential smoothing methods have been proposed initially as just recursive methods without any distributional assumptions about the error structure in them,

and later, they were found to be particular cases of the statistically sound Auto-Regressive Integrated Moving Average (ARIMA) models.

This write-up is intended to provide an overview on both linear and non-linear time series models within the ARMA framework and some frequently used parametric nonlinear models such as Autoregressive conditional heteroscedastic (ARCH) and its generalised form GARCH models. At the end we have given R code for the use of linear and non-linear models on a real data set for better understanding and acceptability.

1. Linear Time Series Models

In a data series containing observations spaced at equal intervals of time often may be correlated. Such correlation between consecutive observations is called *autocorrelation*. When the data is autocorrelated, most of the standard modeling methods based on the assumption of independent observations may become misleading. We therefore need to consider alternative methods that consider the serial dependence in the data which can be achieved by employing time series models such as autoregressive integrated moving average (ARIMA) models.

The most popular class of linear time series models consists of autoregressive moving average (ARMA) models, including purely autoregressive (AR) and purely moving-average (MA) models as special cases. ARMA models are frequently used to model linear dynamic structures, to depict linear relationships among lagged variables, and to serve as vehicles for linear forecasting. A particularly useful class of models contains the so-called autoregressive integrated moving average (ARIMA) models, which includes stationary ARMA - processes as a subclass.

1.1 Autoregressive (AR) Model

A stochastic model that can be extremely useful in the representation of certain practically occurring series is the autoregressive model. In this model, current value of the process is expressed as a finite, linear aggregate of previous values of the process and a shock ε_t. Let us denote the values of a process at equally spaced time epochs $t, t-1, t-2, \ldots$ by $y_t, y_{t-1}, y_{t-2}, \ldots$ then y_t can be described as

$$y_t = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \ldots + \varphi_p y_{t-p} + \varepsilon_t$$

If we define an autoregressive operator of order *p* by

$$\varphi(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p$$

where *B* is the backshift operator such that $By_t = y_{t-1}$, autoregressive model can be written as

$\varphi(B)\, y_t = \varepsilon_t$.

1.2 Moving Average (MA) Model

Another kind of model of great practical importance in the representation of observed time-series is finite moving average process. MA (q) model is defined as

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

If we define a moving average operator of order q by

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q$$

where B is the backshift operator such that $By_t = y_{t-1}$, moving average model can be written as $y_t = \theta(B)\varepsilon_t$.

1.3 Autoregressive Moving Average (ARMA) Model

To achieve greater flexibility in fitting of actual time-series data, it is sometimes advantageous to include both autoregressive and moving average processes. This leads to mixed autoregressive-moving average model

$$y_t = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \varepsilon_t - \theta_1\varepsilon_{t-1} - \theta_2\varepsilon_{t-2} - \cdots \theta_q\varepsilon_{t-q}$$

or

$$\varphi(B)\, y_t = \theta(B)\varepsilon_t$$

and is written as ARMA(p, q). In practice, it is quite often adequate representation of actually occurring stationary time-series can be obtained with autoregressive, moving average, or mixed models, in which p and q are not greater than *2*.

1.4 Autoregressive Integrated Moving Average (ARIMA) Model

A generalization of ARMA models which incorporates a wide class of non-stationary time-series is obtained by introducing the differencing into the model. The simplest example of a non-stationary process which reduces to a stationary one after differencing is Random Walk. A process { y_t } is said to follow an Integrated ARMA model, denoted by ARIMA (p, d, q), if $\nabla^d y_t = (1 - B)^d\, \varepsilon_t$ is ARMA (p, q). The model is written as

$$\varphi(B)(1-B)^d\, y_t = \theta(B)\varepsilon_t$$

ε_t are assumed to be independently and identically distributed with a mean zero and a constant variance of σ^2.

2. Non-linear models: ARCH and GARCH models

After the dominance of the ARIMA model for over two decades, the need of such model was felt which could predict with varying variance of the error term. The solution was provided by Engle (1982) when he developed ARCH model to estimate the mean and variance of the United Kingdom inflation. This model has few interesting characteristics; it models the conditional variance as the square of the function of the previous error term and assumes the unconditional variance to be constant. Along with the ARCH models can model heavy tail data which are common in financial market. Besides these, Bera and Higgins (1993) pointed out that ARCH models are easy and simple to handle, can take care of clustered errors, non-linearity and importantly takes care of changes in the econometrician's ability to forecast.

The ARCH (*q*) model for the series $\{\varepsilon_t\}$ is defined by specifying the conditional distribution of given the information available up to time *t* −1. Let denote this information. ARCH (*q*) model for the series is given by

$$\varepsilon_t|\psi_{t-1} \sim N(0,h_t)$$

$$h_t = a_0 + \sum_{i=1}^{q} a_i\, \varepsilon_{t-i}^2$$

where, $a_0 > 0$, $a_i \geq 0$, for all i and $\sum_{i=1}^{q} a_i < 1$ are required to be satisfied to ensure non-negativity and finite unconditional variance of stationary $\{\varepsilon_t\}$ series. Bollerslev (1986) and Taylor (1986) proposed the Generalized ARCH (GARCH) model independently of each other, in which conditional variance is also a linear function of its own lags and has the following form

$$\varepsilon_t = \xi_t h_t^{1/2} \qquad (1)$$

where ξ_t ~ N (0,1). A sufficient condition for the conditional variance to be positive is

$$a_0 > 0,\ a_i \geq 0,\ i = 1,2,...,q.\ \ b_j \geq 0,\ \ j = 1,2,...,p$$

The GARCH (p, q) process is weakly stationary if and only if

$$. \sum_{i=1}^{q} a_i + \sum_{j=1}^{p} b_j < 1$$

The conditional variance defined by (1) has the property that the unconditional autocorrelation function of ε_t^2 ; if it exists, can decay slowly. For the ARCH family, the decay rate is too rapid compared to what is typically observed in financial time-series, unless the maximum lag *q* is long. As (1) is a more parsimonious model of the conditional variance than a high-order ARCH model, most users prefer it to the simpler ARCH alternative. The most popular GARCH model in applications is the GARCH (*1,1*) model.

Model Building

Step 1: Determine whether the time series is stationary.

The series being analysed must be stationary. A TS is said to be stationary if its underlying generating process is based on a constant mean and constant variance with its autocorrelation function (ACF) essentially constant through time. Thus, if we consider different subsets of a realization (TS 'sample') the different subsets will typically have means, variances and autocorrelation functions that do not differ significantly which means that stationary time series has the property that its statistical properties such as the mean and variance are constant over time. The presence of stationarity in the data can be obtained by simply plotting the raw data or by plotting the autocorrelation and partial autocorrelation function. Statistical tests like Dickey- Fuller test, augmented Dickey-Fuller test, KPSS (Kwiatkowski, Phillips, Schmidt, and Shin) test, Philips-Perron test are also available to test the stationarity.

Step 2: Identify the model.

After the time-series is stationary we go for identifying the mean model for the series. This is done by fitting the simple ARIMA (Autoregressive integrated moving average) model. The ARIMA (p,d,q) is determined by the ACF (Autocorrelation function) and PACF (Partial autocorrelation function) values of the stationary series. The parameter *p* is determined by the ACF value and *q* by the PACF value and *d* refers to order of differencing done to the original series to make it stationary.

Step 3: Estimate the model parameters and diagnostic checking.

Once few tentative models are specified, estimation of the model parameters is straightforward. The parameters are estimated through maximum likelihood function such that an overall measure of errors is minimized or the likelihood function is maximized. This step is basically

to check if the model assumptions about the errors are satisfied. This is achieved by performing portmanteau test. The test is utilized to see whether the model residuals are white noise. The null hypothesis tested is that the current set of residual is white
noise.

The Ljung-Box statistic is given by:

$$Q = n(n+2)\sum_{k=1}^{h}(n-k)^{-1}r_k^{\,2}$$

where, *h* is the maximum lag, *n* is the number of observations, *k* is the number of parameters in the model. If the data are white noise, the Ljung-Box Q statistics has a chi-square distribution with *(h-k)* degrees of freedom.

Step 4: Select the most suitable ARIMA model

The most suitable ARIMA model is selected using the smallest Akaike Information Criterion (AIC) or Schwarz-Bayesian Criterion (SBC). AIC is given by

$$AIC = (-2\log L + 2m)$$

where, $m = p+q$ and L is the likelihood function. SBC is also used as an alternative to AIC which is given by

$$SBC = \log\sigma^2 + (m\log n)/n$$

If the model is not adequate, a new tentative model should be identified, which is again followed by the parameter estimation and model verification. Diagnostic information may help suggest alternative model(s). The steps of model building process are typically repeated several times until a satisfactory mean model is finally selected. The final model can then be used for prediction purposes.

Step 5: Determination of residuals and heteroscedasticity test.

After finding the mean model now the residuals are to be determined. And we create a new variable called ‘rsquare’ by squaring the residuals. Then the ACF and PACF values of the ‘rsquare’ are determined and the lags in which these values are found to be significant are identified. The test for heteroscedasticity is done at identified significant lags. The test employed is the ARCH-LM test.

Step 6: Residuals and diagnostic checking.

The residuals obtained from the mean model used for fitting the different GARCH models were squared and stored in a new variable called 'esquare'. As already mentioned previously, the diagnostic tests are employed to check whether the residuals are white noise or not.

Step 7: Estimation of parameters.

The parameters of the obtained model are estimated using method of maximum likelihood (MLE). And then forecasting is done using the selecting model.

5. Illustration

In this illustration Cotlook A index data is used and was collected from the commodity price bulletin, published by the United Nations Convention of Trade and Development (UNCTAD). The series contains 360 data pints, 346 data points are used for modelling and remaining 14 points for forecasting. At first the ARIMA model was applied to the data set and on unsatisfactory performance of the model, the GARCH model was used.

5.1 Fitting of the Cotlook A index

Various combinations of the ARIMA models were tried, among all, the AR (1) model had minimum AIC and BIC values. The AIC value for fitted GARCH model has been found to be minimum when the mean equation depends on two recent pasts only. Investigating the autocorrelation function (Acf) of squared residuals of AR (2) model, it is found that the Acf and Pacf are maximum at lag 3, which is 0.226 and 0.221 respectively. But if we go for AR (2)-ARCH (3) model, a large number of parameters are needed to be estimated. So, to get a parsimonious model, the AR (2)-GARCH (1, 1) model is selected.

The mean and conditional variance for fitted AR (2)-GARCH (1, 1) model is computed as follows:

$$y_t = 141.9264 \; -1.3905 \; y_{t-1} + 0.4538 \, y_{t-2} + \varepsilon_t$$

$$(3.94) \quad (0.05) \qquad (0.05)$$

where

$$\varepsilon_t = h_t^{1/2} \xi_t \,,$$

and h_t satisfies the variance equation

$$h_t = 8.470 + 0.208\ \varepsilon_{t-1}^2 + 0.215\ h_{t-1}$$

(1.97) (0.09) (0.079)

The values within brackets denote corresponding standard errors of the estimates. The AIC value, for fitted GARCH model is 2288.88.

Table 1. Forecast of the Cotlook A index series

MONTH	ACTUAL VALUE	FORECAST ARIMA(1,1,0)	FORECAST AR(2)-GARCH(1,1)
Feb-11	469.98	408.34(8.30)	389.59(26.46)
Mar-11	506.34	416.47(15.56)	371.55(25.74)
Apr-11	477.56	421.40(22.35)	348.54(25.05)
May-11	364.91	424.53(28.55)	324.69(24.39)
Jun-11	317.75	426.66(34.17)	301.98(23.75)
Jul-11	268.96	428.23(39.29)	281.25(23.13)
Aug-11	251.55	429.49(43.97)	262.76(22.54)
Sep-11	257.63	430.57(48.29)	246.50(21.97)
Oct-11	243.85	431.55(52.30)	232.32(21.42)
Nov-11	230.78	432.48(56.05)	220.01(20.90)
Dec-11	210.43	433.37(59.58)	209.35(20.39)
Jan-12	222.91	434.25(54.45)	200.15(19.91)
Feb-12	222.12	435.12(57.13)	192.21(19.44)
Mar-12	219.36	435.99(59.68)	185.37(19.01)

Table 2. Forecast evaluation of the Cotlook A index series

MODEL	RMSE	RMAPE (%)
ARIMA(1,1,0)	44.03	60.72
AR(2)-GARCH(1,1)	15.38	9.36

6. R code for analysing a time series data

```
library("tseries")
library("forecast")
library("fgarch")
setwd("C:/Users/ACHAL/Desktop") # Setting of the work directory
data<-read.table("data.txt") # Importing data
datats<-ts(data,frequency=12,start=c(1982,4)) # Converting data set into time series
plot.ts(datats) # Plot of the data set
adf.test(datats) # Test for stationarity
diffdatats<-diff(datats,differences=1) # Differencing the series
datatsacf<-acf(datats,lag.max=12) # Obtaining the ACF plot
datapacf<-pacf(datats,lag.max=12) # Obtaining the PACF plot
auto.arima(diffdatats) # Finding the order of ARIMA model
datatsarima<-arima(diffdatats,order=c(1,0,1),include.mean=TRUE) # Fitting of ARIMA model
forearimadatats<-forecast.Arima(datatsarima,h=12) # Forecasting using ARIMA model
plot.forecast(forearimadatats) # Plot of the forecast
residualarima<-resid(datatsarima) # Obtaining residuals
archTest(residualarima,lag=12) # Test for heteroscedascity
# Fitting of AR-GARCH model
garchdatats<-garchFit(formula = ~ arma(2)+garch(1, 1), data = datats, cond.dist = c("norm"), include.mean = TRUE, include.delta = NULL, include.skew = NULL, include.shape = NULL, leverage = NULL, trace = TRUE,algorithm = c("nlminb"))
# Forecasting using AR-GARCH model
forecastgarch<-predict(garchdatats, n.ahead = 12, trace = FALSE, mse = c("uncond"), plot=FALSE, nx=NULL, crit_val=NULL, conf=NULL)
plot.ts(forecastgarch) # Plot of the forecast
```

References:

Bera, A. K., and Higgins, M. L. (1993), ARCH Models: Properties, Estimation and Testing, *Journal of Economic Surv*ey, **7**, 307-366.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics*, **31**, 307-327.

Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (2007). Time-Series Analysis: Forecasting and Control. 3rd edition. *Pearson education*, India.

Engle, R.F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of U.K. inflation. *Econometrica*, **50**, 987-1008.

Fan, J. and Yao, Q. (2003). *Nonlinear time series:nonparametric and parametric methods*. Springer, U.S.A.

Taylor, S. J. (1986). Modeling financial time series. Wiley, New York.

Introduction to multivariate time series model

Achal Lama

ICAR-Indian Agricultural Statistics Research Institute, New Delhi-110012

achal.lama@icar.gov.in

1. Introduction

Modelling and forecasting of major economic phenomenon involve a large number of variables, thus it must be addressed using the multivariate time-series methods. A large number of time-series models have been proposed in literature as alternatives to structural econometric models in economic forecasting applications. But, after the pioneering work of Sims (1980) the VAR models have received much attention. The class of VAR models is a special case of more general Vector Autoregressive Moving Average (VARMA) models. The VAR models were initially used as macroeconomic models, but it has been found as a promising alternative to structural econometric models where simultaneous forecasts are required for a collection of related microeconomic variables.

Let $y_t = (y_{1t}, y_{2t}, ..., y_{nt})'$ denote an (n×1) vector of time series variables. The basic p-lag vector autoregressive VAR (p) model has the form:

$$y_t = A + B_1 y_{t-1} + B_2 y_{t-2} + B_3 y_{t-3} + ... + B_p y_{t-p} + \varepsilon_t$$

where, A is $k\times 1$ vector of intercepts, B_i (i =1, 2, …, p) is $k \times k$ matrices of parameters and

$$\varepsilon_t \sim iidN(0, \Sigma)$$

To have an understanding of the VAR model let us consider a simple two variable case and the lag be one, i.e., p = 1 and k = 2. The VAR model can be represented as:

$$y_t = \begin{bmatrix} y_{1t} \\ y_{2t} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} b_{11}\ b_{12} \\ b_{21}\ b_{22} \end{bmatrix} \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1t} \\ \varepsilon_{2t} \end{bmatrix}$$

$$= A + By_{t-1} + \varepsilon_t$$

Now further defining

$$x_t = (y_{t-1}, ..., y_{t-p})'$$

and,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_T \end{bmatrix}$$

The likelihood function can be derived in two parts as follows:

$$\alpha|\Sigma, y \sim N(\hat{\alpha}, \Sigma \otimes (X'X)^{-1})$$

and

$$\Sigma^{-1}|y \sim W(S^{-1}, T - K - M - 1)$$

where, $\hat{B} = (X'X)^{-1}X'Y$ is the OLS estimates of B and $\hat{\alpha} = vec(\hat{B})$ and $S = (Y - X\hat{B})'(Y - X\hat{B})$.

2. Fitting of VAR model

In this section attempt has been made to summarize broadly the steps followed for modelling a multivariate time series data using VAR model. The steps are as follows:

1. **Determination of Stationarity of the time series**

   The Stationarity of the data sets used is tested. As, it is the basic assumption which needs to be satisfied before proceeding for analysis of any time-series data. Statistical tests like Dickey-Fuller test, Augmented Dickey-Fuller (ADF) test, KPSS (Kwiatkowski, Phillips, Schmidt, and Shin) test, Philips-Perron test are available to test the stationarity. If required, the series needs to be differenced to make it stationary in mean.

2. **Identification the mean model**

   After the time-series is stationary we go for identifying the mean model for the series. This is done by fitting the simple ARIMA (Autoregressive integrated moving average) model. The ARIMA (*p*,*d*,*q*) is determined by the ACF (Autocorrelation function) and PACF (Partial autocorrelation function) values of the stationary series. The parameter *p* is determined by the ACF value and *q* by the PACF value and *d* refers to order of differencing done to the original series to make it stationary. This procedure is useful for univariate case, but in case of multivariate setup we have to use either VAR or Vector Error Correction (VEC) model. The selection of the model depends upon the nature of the series to be modelled. If the series have long run dependency among themselves one need to use the VEC or else VAR model is used. Johansen test for

cointegration is used to have an insight into the dependency relationship with the alternate hypothesis being the presence of cointegration. If we do not find cointegration between the series then we use the VAR model to identify the mean process. The order of the VAR model is identified based on minimum Akaike Information Criterion (AIC) or Schwarz-Bayesian Criterion (SBC). AIC is given by

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

where, k is the number of parameters of the model and L is the likelihood function. SBC is also used as an alternative to AIC which is given by

$$SBC = \log \sigma^2 + (k \log n)/n$$

3. Residual diagnostics

The parameters of the VAR model is estimated through maximum likelihood function such that an overall measure of errors is minimized or the likelihood function is maximized. This step is basically to check if the model assumptions about the errors are satisfied. To achieve this we used the autocorrelation function (ACF) plots for testing the presence of serial correlation in the residual series at various lags, we have also used the Q-Q normal probability plots to check the normality assumption of the residuals. If the residuals are either serially correlated or non-normal, we need to repeat Step 2.

4. Model comparison

The efficiency of the VAR model for forecasting is compared on the basis of following criterion:

Root Mean Squared Error (RMSE) expressed as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(Y_{p,i} - Y_{0,i}\right)^2}$$

Mean absolute error (MAE) expressed as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\left(Y_{p,i} - Y_{0,i}\right)\right|$$

where

Y_0 and Y_p are the observed and forecasted time series

N is the number of data points

The VAR model with minimum RMSE and MAE is selected for forecasting the multivariate time series under consideration.

The above described steps are followed and implemented using R software by calling the package "vars". VAR model can be implemented in various other software like SAS, EViews, etc.

3. Impulse Response (IR)

Apart from modelling and forecasting VAR model is used for studying the impulse of one variable on the other variables of the system. As, VAR models represent the correlations among a set of variables, they are often used to analyse certain aspects of the relationships between the variables of interest. Granger (1969) has defined a concept of causality which, under suitable conditions, is fairly easy to deal within the context of VAR models. Therefore, it has become quite popular in recent years. The idea is that a cause cannot come after the effect. If a variable x affects a variable z, the former should help improving the predictions of the latter variable. Ωt is the information set containing all the relevant information in the universe available upto and including period t. $z_t(h|\Omega t)$ be the optimal (minimum MSE) h-step predictor of the process z_t at origin t, based on the information in Ωt. The corresponding forecast MSE: $\Sigma_t(h|\Omega t)$. The process x_t is said to cause z_t in Granger's sense if

$$\Sigma_t(h|\Omega t) < \Sigma_t(h|\Omega t \ \{x_s : s \leq t\}) \text{ for at least one } h = 1,2,...$$

$\Omega t \ \{x_s : s \leq t\}$ is the set containing all the relevant information in except for the information in the past and present of the x_t process. If z_t can be predicted more efficiently if the information in the x_t process is taken into account in addition to all other information, then x_t is Granger-causal for z_t. Granger-causality may not tell us the complete story about the interactions between the variables of a system. In applied work, it is often of interest to know the response of one variable to an impulse in another variable in a system that involves a number of further variables as well. One would like to investigate the impulse response relationship between two variables in a higher dimensional system. If there is a reaction of one variable to an impulse in another variable we may call the latter causal for the former. We will study this type of causality by tracing out the effect of an exogenous shock or innovation in one of the variables on some or all of the other variables. This kind of impulse response analysis is called multiplier analysis.

4. Data and implementation

For illustration we have taken three series namely the Food Wholesale price index (FWPI), monthly rainfall (mm) data and the Fiscal deficit data to study the BVAR model. The FWPI data was collected from Office of the Economic Adviser, Ministry of Commerce and Industry,

Government of India. The rainfall data was collected from the official website of India Meteorological Department (IMD), Ministry of Earth Sciences, Government of India. The Fiscal deficit data was collected from the official website of Reserve Bank of India (RBI). All three series contain 120 data points from January, 2005 to December, 2014 out of which 114 points were used for model building purpose and the remaining 6 points were kept for validation. The time plot of the data set is depicted by Figure 1 and the descriptive statistics is given in Table 1. Looking at the time plot of the series we can identify the presence of seasonality in the rainfall series, hence the series were seasonally adjusted following standard procedure. From the descriptive statistics we can have an idea that the series under consideration is slightly skewed, has a small amount of kurtosis and also the series is non-normal. VAR model was fitted as per the steps described in Section 2 and hence forecasts were also obtained. Estimates of VAR model is presented in Table 2. The impulse response graph is depicted in Figure 2. From Table 2 we could find the interdependencies among the three series used for analysis. This helps the researcher to describe the movement of the series together. Further, with help of impulse response function we can find exactly how one series is affected by other series independently. This provides a good insight into the transfer of shocks between the variables in the system.

Table 1. Descriptive statistics

	FISCAL	FWPI	RAINFALL
Mean	302.30	167.28	94.12
Median	294.69	164.10	44.90
Maximum	1273.83	265.30	334.10
Minimum	-911.50	97.60	1.70
Std. Dev.	321.03	49.98	95.24
Skewness	-0.14	0.34	0.97
Kurtosis	4.58	1.84	2.48
Jarque-Bera	13.04	9.01	20.45
Probability	< 0.01	<0.01	< 0.01

Table 2. VAR model estimates

	FISCAL_D	FWPI_D	RAINFALL_SA
FISCAL_D(-1)	-0.71	0.0002	0.008
	(0.09)	(0.001)	(0.01)
FISCAL_D(-2)	-0.60	0.002	-0.009
	(0.10)	(0.001)	(0.01)
FISCAL_D(-3)	-0.33	0.004	0.005
	(0.12)	(0.001)	(0.01)
FISCAL_D(-4)	-0.28	0.002	0.001
	(0.11)	(0.001)	(0.01)
FISCAL_D(-5)	-0.35	0.001	0.004
	(0.09)	(0.001)	(0.01)
FWPI_D(-1)	-2.47	0.14	-1.88
	(8.78)	(0.09)	(1.06)
FWPI_D(-2)	-9.49	-0.003	0.61
	(8.64)	(0.09)	(1.05)
FWPI_D(-3)	13.30	0.17	-1.91
	(8.57)	(0.09)	(1.04)
FWPI_D(-4)	-23.09	-0.25	0.25
	(8.97)	(0.09)	(1.09)
FWPI_D(-5)	-14.10	-0.30	-0.41
	(8.85)	(0.09)	(1.07)
RAINFALL_SA(-1)	0.62	0.01	-0.06
	(0.85)	(0.009)	(0.10)
RAINFALL_SA(-2)	0.33	-0.01	-0.09
	(0.87)	(0.009)	(0.10)
RAINFALL_SA(-3)	1.25	-0.01	0.10
	(0.86)	(0.009)	(0.10)
RAINFALL_SA(-4)	1.34	-0.01	0.03
	(0.89)	(0.009)	(0.10)
RAINFALL_SA(-5)	-0.94	0.01	0.07
	(0.80)	(0.008)	(0.09)
C	-197.59	2.34	97.63
	(197.70)	(2.17)	(24.03)

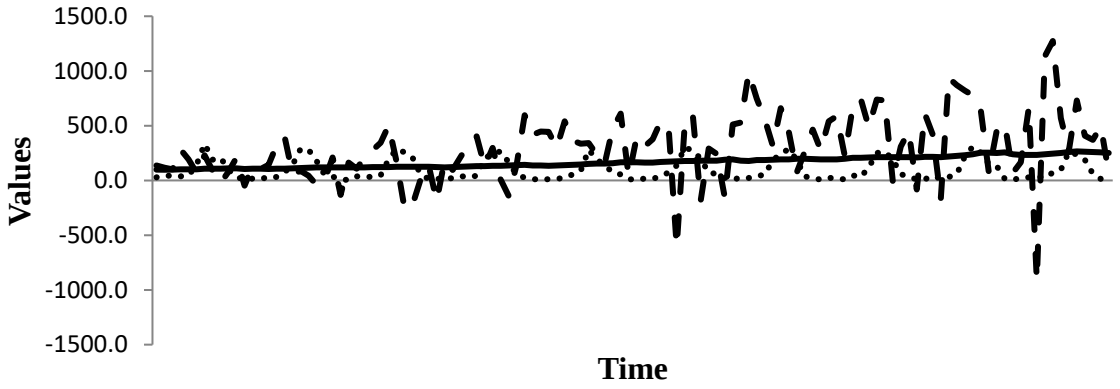


Figure 1. Time plot of Food WPI (solid line), Rainfall (dotted line) and Fiscal (dashed line)

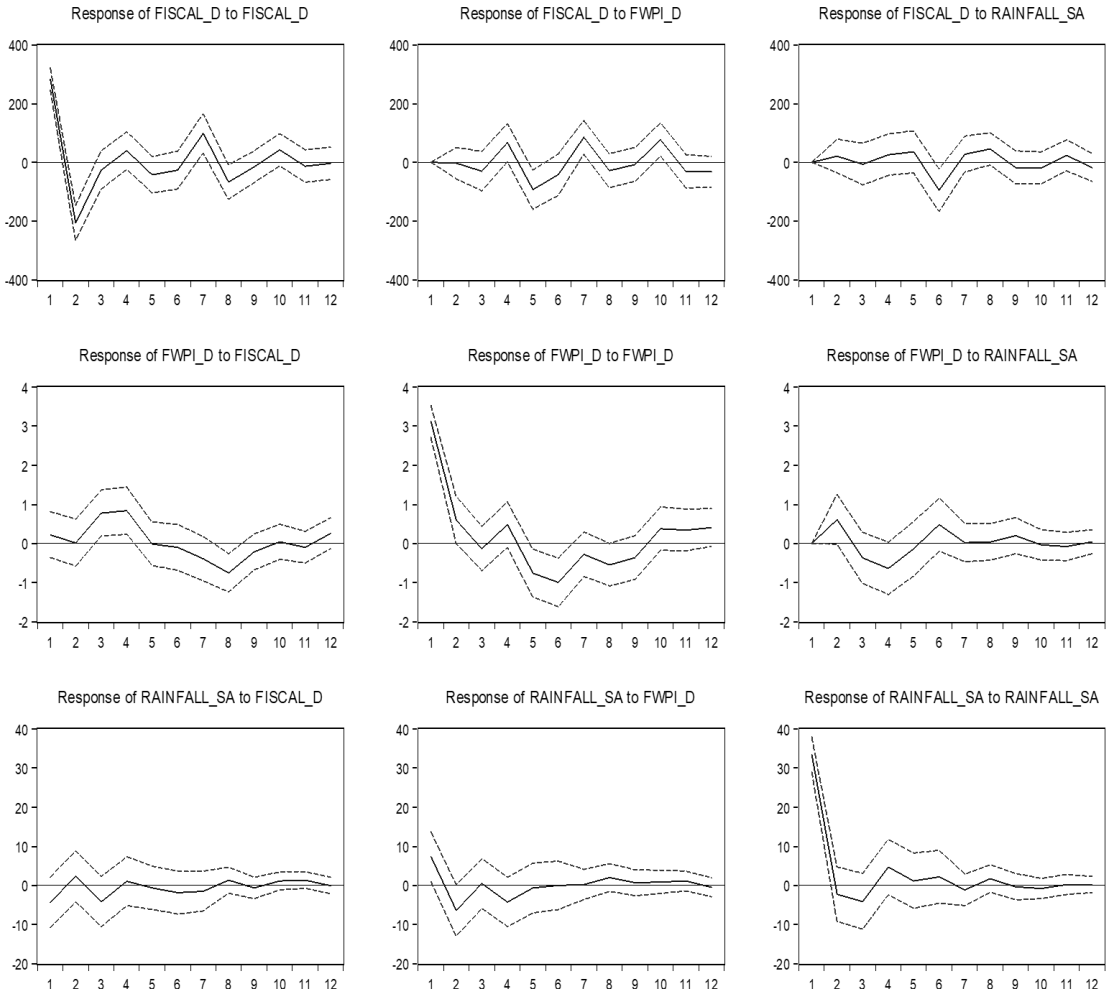


Figure 2. The impulse response graph of VAR model

R code:

 # Loading required pakages

```
library(vars)
library(tseries)
library(rmgarch)
library(fGarch)
library(fBasics)
setwd("C:/Users/BABU/Desktop")# Setting the working directory
data<-read.table("Achal.txt",header=TRUE)# Importing the data set
X<-data[,1]
Y<-data[,2]
adf.test(X)# Testing the stationarity of the data set
adf.test(Y)
X_d=diff(X)# Differencing the data set
```

```
Y_d=diff(Y)
archTest(X) # Testing ARCH effect of the data set
data_d<-data.frame(X_d,Y_d)
var_1=VAR(data,type="const",lag.max=12,ic="AIC") # Fittting VAR model to the data set
irf(var_1, impulse = "X", response = c("Y"), boot = FALSE)
plot(irf(var_1, impulse = "X", response = c("Y"), boot = FALSE)) # Plotting the IRF of the model
# Testing the residuals of the model
normality.test(var_1)
predict(var_1, n.ahead = 6, ci = 0.95)
serial.test(var_1, lags.pt = 12, type = "PT.adjusted")
library(MTS)
MarchTest(data, lag = 12) # Testing for the presence of multivariate ARCH effect
out1<-BEKK11(data, include.mean = T, cond.dist = "normal") #Fitting MGARCH- BEKK model
#Fitting MGARCh-DCC model
garch11.spec = ugarchspec(mean.model = list(armaOrder = c(1,0)),variance.model = list(garchOrder = c(1,1),model = "sGARCH"), distribution.model = "norm")
dcc.garch11.spec = dccspec(uspec = multispec( replicate(2, garch11.spec) ), dccOrder = c(1,1), distribution = "mvnorm")
dcc.fit = dccfit(dcc.garch11.spec, data=data, fit.control=list(scale=TRUE))
print(dcc.fit)
```

Bibliography:

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods, *Econometrica: Journal of the Econometric Society*, 424-438.

Kynclová, P., Filzmoser, P. and Hron, K. (2015). Modeling Compositional Time Series with Vector Autoregressive Models. *Journal of Forecasting*, **34**, 303–314.

Jouini, T. (2015). Efficient Multistep Forecast Procedures for Multivariate Time Series. *Journal of Forecasting*, **34**, 604–618.

Lama, A., Jha, G.K., Gurung, B., Paul,R.K. and Sinha, K. (2016). VAR-MGARCH Models for Volatility Modelling of Pulses Prices: An Application. *Journal of the Indian Society of Agricultural Statistics*, **70**, 145-151.

Lama, A. (2017). Investigations on Bayesian multivariate time-series models. *Unpublished PhD thesis*, PG School, ICAR-IARI.

Litterman, R. (1980). *Techniques for Forecasting with Vector Autoregressions.* University of Minnesota, Ph. D. Dissertation, Minneapolis.

Ramos, F., F., R. (2003) Forecasts of market shares from VAR and BVAR models: a comparison of their accuracy. *International Journal of Forecasting*, **19,** 95-110.

Sims, C. (1980). Macroeconomics and reality. *Econometrica*, **48**, 1-48.

Cointegration and Causality Analysis

[1]Girish K Jha and [2]Rajeev Ranjan Kumar

[1]ICAR-Indian Agricultural Research Institute; [2]ICAR-Indian Agricultural Statistics Research Institute, New delhi-12

girish.stat@gmail.com

1. Introduction

Time series modelling and forecasting is based on the assumption of stationarity that is constancy of parameters like mean, variance and trend over time. Nonstationarity means a variable has no clear tendency to return to a constant value or a linear trend. Most of the macroeconomic time series datasets found to be nonstationary. An important objective of empirical research in macroeconomics is to test hypothesis and estimate relationships among such aggregate variables. The statistical theories applied during the 1980s in building and testing large simultaneous equation models, such as Ordinary Least Squares (OLS), were based on the assumption that the variables were stationary. The problem is that the statistical inference associated with stationary processes is invalid for nonstationary time series. With nonstationary time series, it would be a methodological error to use OLS to estimate their long-run linear relationships because it would lead to spurious regression. Spurious regression is a situation in which there appears to be a statistically significant relationship between variables but the variables are unrelated. The technique of cointegration can better be used to estimate long-run relationships between nonstationary time series data.

Granger (1981) shown that macroeconomic models containing nonstationary stochastic variables can be constructed in such a way that the results are both statistically sound and economically meaningful. His work has also provided the underpinnings for modeling with rich dynamics among interrelated economic variables. Granger has achieved this breakthrough by introducing the concept of cointegrated variables. Cointegration is an econometric concept which mimics the existence of a long-run equilibrium among economic time series. If two or more series are themselves nonstationary, but a linear combination of them is stationary, then they are said to be cointegrated.

2. Stationarity and unit root test

A stochastic process is said to be stationary if its mean and variance are constant over time and the value of the covariance between the two time periods depends only on the distance or gap or lag between the two time periods and not the actual time at which the covariance is

computed. In the time series literature, such stochastic process is known as a weakly stationary. Nonstationarity in a time series occurs when there is no constant mean, no constant variance, or both of these properties. It can originate from various sources but the most important one is the unit root.

2.1 Unit root

Any sequence that contains one or more characteristic roots that are equal to one is called a unit root process. The simplest model that may contain a unit root is the *AR(1)* model. Consider the autoregressive process of order one, *AR(1)*, below:

$$Y_t = \rho Y_{t-1} + X_t^{'}\alpha + e_t \qquad (1)$$

where X_t are optional exogenous regressors which may consists of constant or a constant and trend, ρ and α are paramets to be estimated and the e_t denotes a serially uncorrected white noise error term with a mean of zero and a constant variance. If $\rho = 1$, equation (1) becomes a random walk without drift model, that is, a nonstationary process. When this happens, we face what is known as the unit root problem. This means that, we face with a situation of nonstationarity in the series. If, however, $\rho < 1$, then the series Y_t is stationary.

2.2 The Augmented Dickey-Fuller (ADF) test

The basic idea behind the ADF unit root test for nonstationarity is to simply regress Y_t on its (one period) lagged value Y_{t-1} and find out the estimated ρ is statistically equal to one or not. Equation (1) can be manipulated by subtracting Y_{t-1} from both sides to obtain

$$Y_t - Y_{t-1} = (\rho - 1)Y_{t-1} + X_t^{'}\delta + e_t$$

$$\Delta Y_t = \alpha Y_{t-1} + X_t^{'}\delta + e_t \qquad (2)$$

where $\alpha = \rho - 1$ and Δ is first difference operator.

In practice, instead of estimating equation (1), we shall estimate equation (2) and test for the null hypothesis of $\alpha = 0$ against the alternative of $\alpha \neq 0$. If $\alpha = 0$, then $\rho = 1$, meaning that we have a unit root problem and the series under consideration is nonstationary. The decision to reject or not to reject the null hypothesis of $\alpha = 0$ is based on the Dickey-Fuller (DF) critical

values of the τ (tau) statistic. The DF test is based on an assumption that the error terms e_t are uncorrelated.

However, in practice, the error terms in the DF test usually show evidence of serial correlation. To solve this problem, Dickey and Fuller have developed a test known as the Augmented Dickey-Fuller (ADF) test. In the ADF test, the lags of the first difference are included in the regression equation in order to make the error term e_t white noise and, therefore, the regression equation is presented in the following form:

$$\Delta Y_t = \alpha Y_{t-1} + X_t' \delta \sum_{i=1}^{k} \beta_i \Delta Y_{t-i} + e_t \qquad (3)$$

where *k* denotes the lag length.

3. Testing of Cointegration

3.1 Engle-Granger cointegration

Engle and Granger (1987) solution to spurious regression problem may be illustrated by the following regression equation:

$$Y_t = \beta_0 + \beta_1 X_t + e_t \qquad (4)$$

where Y_t is the dependent variable, X_t the single exogenous regressor, and $\{e_t\}$ a white-noise, mean-zero sequence.

Granger (1981) argues that in order to be meaningful, an equation has to be consistent in the sense that "a simulation of the explanatory right-hand side should produce the major properties of the variable being explained". For example, if Y_t is a seasonal variable, then X_t has to be seasonal, if e_t is to be white noise. To develop the idea further, Granger (1981) defined the concept of degree of integration of a variable. If variable X_t can be made approximately stationary by differencing it *d* times, it is called integrated of order *d*, or *I*(*d*).

Consider equation (4) and assume that both $Y_t \sim I(1)$ and $X_t \sim I(1)$. Then, generally $Y_t - \beta_1 X_t \sim I(1)$ as well. There is, however, one important exception. If $e_t \sim I(0)$, then $Y_t - \beta_1 X_t \sim I(0)$, i.e., the linear combination $Y_t - \beta_1 X_t$ has the same statistical properties as an

I(0) variable. There exists only one such combination so that coefficient β_1 is unique. In this special case, variables X_t and Y_t are called cointegrated. More generally, if a linear combination of a set of *I(1)* variables is *I(0),* then the variables are cointegrated. This concept, introduced by Granger (1981) has turned out to be extremely important in the analysis of nonstationary economic time series. A generalization to *I(d)* variables, where *d* is no longer an integer, is also possible, in which case the linear combination of cointegrated variables has to be $I(d-d_0)$,where $d_0 > 0$.

Engle-Granger method for testing cointegration

To explain the Engle-Granger testing procedure, let's begin with the type of problem likely to be encountered in applied studies. Suppose that two variables Y_t and X_t are believed to be integrated of order 1 and we want to determine whether there exists an equilibrium relationship between the two. Engle and Granger (1987) proposed a four step procedure to determine if two *I*(1) variables are cointegrated of order *CI*(1,1).

Step-1: Pretest the Variables for their Order of Integration

By definition, cointegration necessitates that two variables be integrated of same order. Thus, the first step in the analysis is to pretest each variable to determine its order of integration. The Augmented Dickey-Fuller test can be used to infer the number of unit roots in each of the variables. If both variables are stationary, it would not be necessary to proceed since standard time series methods apply to stationary variables. If the variables are integrated of different orders, it should be concluded that they are not cointegrated.

Step-2: Estimation of long-run equilibrium relationship

If, both variables Y_t and X_t are *I*(1), then estimate the long-run equilibrium relationship in the form

$$Y_t = \beta_0 + \beta_1 X_t + e_t$$

If the variables are cointegrated, an ordinary least squares regression yields a super-consistent estimator of the cointegrating parameters β_0 and β_1.

Step-3: Estimate the Error-Correction Model (ECM)

If the variables are cointegrated, the residuals from the equilibrium regression can be used to estimate the ECM.

Step-4: Assess model adequacy

There are several procedures available that determine whether the estimated error-correction model is appropriate or not.

4.2 Johansen's cointegration

To test the presence of cointegration using a single equation (i.e., Engle-Granger approach) becomes a bit restrictive. Let us consider a situation where we have p>2 variables in the model and p-1 of them are not weakly exogenous, then the single equation approach can be misleading, particularly if there is more than one cointegration relationship present. Thus, when the number of cointegration vectors is unknown and there is a need to allow all variables in the model to be potentially endogenous, the multivariate Vector Autoregressive (VAR) approach developed by Johansen (1988) is efficient. Johansen test of cointegration allow the researcher to test restricted version of the cointegrating vector(s) and speed of adjustment parameters. The first step of Johansen test involves the determination of cointegrating rank, that is, the number of cointegrating relations. Johansen's procedure builds cointegrated variables directly on maximum likelihood estimation instead of relying on Ordinary Least squares (OLS) estimation. Johansen derived the maximum likelihood estimates using sequential tests for determining the number of cointegrating vectors. In fact, Johansen's procedure is nothing more than a multivariate generalisation of the Dickey-Fuller test.

The Johansen cointegration procedure is based upon an unrestricted vector autoregressive (VAR) model specified in error-correction form as follows:

$$\mathbf{Y}_t = A_1\mathbf{Y}_{t-1} + A_2\mathbf{Y}_{t-2} + \ldots + A_k\mathbf{Y}_{t-k} + \mathbf{e}_t$$

$$\Delta\mathbf{Y}_t = \Pi\mathbf{Y}_{t-1} + \sum_{i=1}^{k-1}\Gamma_i\,\Delta\mathbf{Y}_{t-i} + \mathbf{e}_t \qquad (5)$$

where

$$\Pi = -(I - A_1 - A_2 - \ldots - A_k)$$

$$\Gamma_i = (I - A_1 - A_2 - \ldots - A_i), \qquad i = 1,\ldots,k-1$$

$\mathbf{Y}_t$ includes all p variables (for example price indices of crude oil, rice etc.) of the model which are $\sim I(1)$,

Π and Γ_i are parameter matrices to be estimated,

$\mathbf{e}_t$ is a vector of random errors which follow a Gaussian white noise process.

The Johansen test for cointegration evaluates the rank (r) of the matrix Π. If $r = 0$, all variables are $I(1)$ and thus not cointegrated. In case $0 < r < p$, there exist r cointegrating vectors. In the third case, if $r = p$ all the variables are $I(0)$ and thus stationary, and any combination of stationary variables will be stationary. Π represent the long response matrix and is defined as the product of two matrices: θ and β', of dimension $(p \times r)$ and $(r \times p)$, respectively. The β matrix contains the long-run coefficients of the cointegrating vectors, θ is known as the adjustment parameter matrix and is similar to an error correction term.

The Johansen cointegration method estimates the Π matrix through an unrestricted VAR and tests whether one can reject the restriction implied by the reduced rank of Π. Two methods of testing for reduced rank of Π are the trace test and the maximum eigen value test, respectively:

$$\lambda_{trace} = -T \sum_{i=r+1}^{n} \ln(1 - \hat{\lambda}^{2}_{i}) \qquad (6)$$

$$\lambda_{\max}(r, r+1) = -T \ln(1 - \hat{\lambda}_{r+1}) \qquad (7)$$

where, λ_i is the estimated values of the ordered eigen values obtained from the estimated matrix and T is the number of the observations after the lag adjustment.

The trace statistics test the null hypothesis that the number of distinct cointegrating vectors (r) is less than or equal to r against a general alternative. The maximum eigen value tests the null hypothesis that the number of cointegrating vectors is r against the alternative of $r+1$ cointegrating vectors.

5. Causality from vector error correction model (VECM)

The existence of cointegration in the bi-variate relationship implies Granger causality at least in one direction which under certain restrictions can be tested within the framework of Johansen cointegration by the Wald test. If the θ matrix in the cointegration matrix Π has a complete column of zeros, no causal relationship exists since no cointegrating vector appears in that particular block. Pair wise causal relationship can be represented through the following equation:

$$\begin{bmatrix} \Delta Y_{1,t} \\ \Delta Y_{2,t} \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \left(Y_{1,t-1} - \beta Y_{2,t-1} \right) + A_1 \begin{bmatrix} \Delta Y_{1,t-1} \\ \Delta Y_{2,t-1} \end{bmatrix} + \ldots + A_k \begin{bmatrix} \Delta Y_{1,t-k} \\ \Delta Y_{2,t-k} \end{bmatrix} + \begin{bmatrix} e_{1t} \\ e_{2t} \end{bmatrix} \qquad (8)$$

Parameters contained in matrices A_k measure the short-run causality relationship, while β is the cointegrating parameter that characterizes the long-run equilibrium relationship between the series. Through Eq. (8), three possibilities for long-run causality may be identified, (i) $\theta_1 \neq 0, \theta_2 \neq 0$; (ii) $\theta_1 = 0, \theta_2 \neq 0$; and (iii) $\theta_1 \neq 0, \theta_2 = 0$.

The first case indicates bi-directional causality, while the second and third imply uni-directional causality. To analyze for short-run causality we apply the Wald test with the null hypothesis that the joint contribution of the lags of endogenous variables is equal to zero. If the null hypothesis cannot be rejected it implies that the respective endogenous variables can be treated as exogenous in the system. In case of bi-variate models, the Johansen cointegration Eq. (5) can be rewritten as

$$\Delta Y_{1,t} = \mu_1 + \sum_{i=1}^{k1} \beta_i \Delta Y_{1,t-i} + \sum_{j=1}^{k2} \beta_j \Delta Y_{2,t-j} + \alpha_1 ECT_{t-1} + e_{t,1} \quad (9)$$

$$\Delta Y_{2,t} = \mu_2 + \sum_{i=1}^{k1} \beta_i \Delta Y_{1,t-i} + \sum_{j=1}^{k2} \beta_j \Delta Y_{2,t-j} + \alpha_1 ECT_{t-1} + e_{t,2} \quad (10)$$

where, $Y_{1,t}$ and $Y_{2,t}$ are time series (of prices) and ECT is the error correction term. We test the short run causality through Eqs. (9) and (10), by examining the significance of all lagged dynamic terms.

6. Illustration

For the Illustration, monthly potato price of Delhi and Agra markets from Jan-2005 to Dec 2019 has been used, which is obtained from National Horticultural and Development Foundation (NHRDF) website (http://nhrdf.org/en-us/). It has been done using "R" software.

Import Data

To import the data in the 'R' software use the following command. The imported data are saved as "data" variable in this case. To see the first few row of the data use the command 'head(data)'.

```
data <- read.csv(choose.files(), header = TRUE)
head(data)
```

```
##   Month  Agra Delhi Mumbai Bangalore
## 1 5-Jan 194.0   258    394       436
## 2 5-Feb 221.0   245    451       429
## 3 5-Mar 315.5   262    508       473
## 4 5-Apr 410.0   454    664       658
## 5 5-May 396.0   497    712       718
## 6 5-Jun 383.0   659    702       736
```

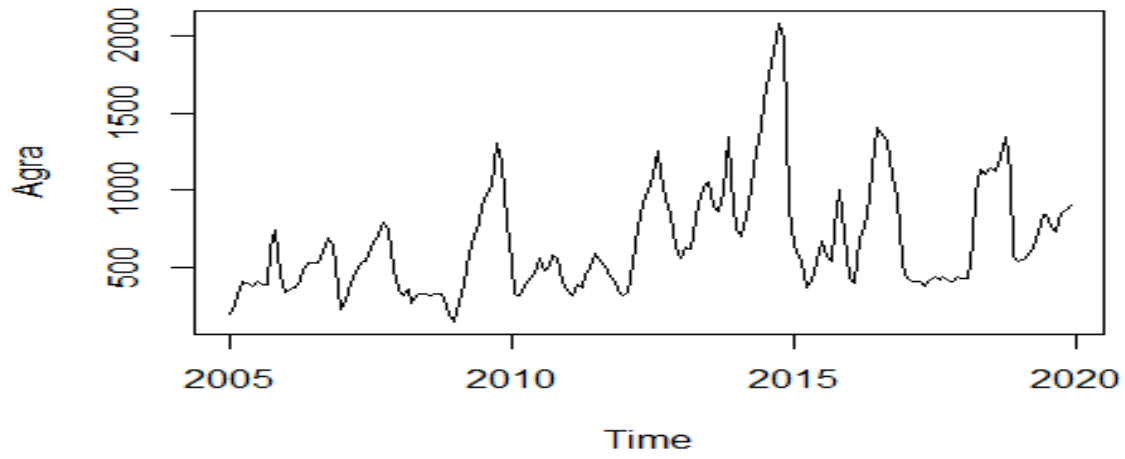
Extract Agra and Delhi Series

The whole data contain potato price series of four markets viz. Agra, Delhi, Mumbai and Bangalore. From the following command we extract the price series of 'Agra' and 'Delhi' markets.

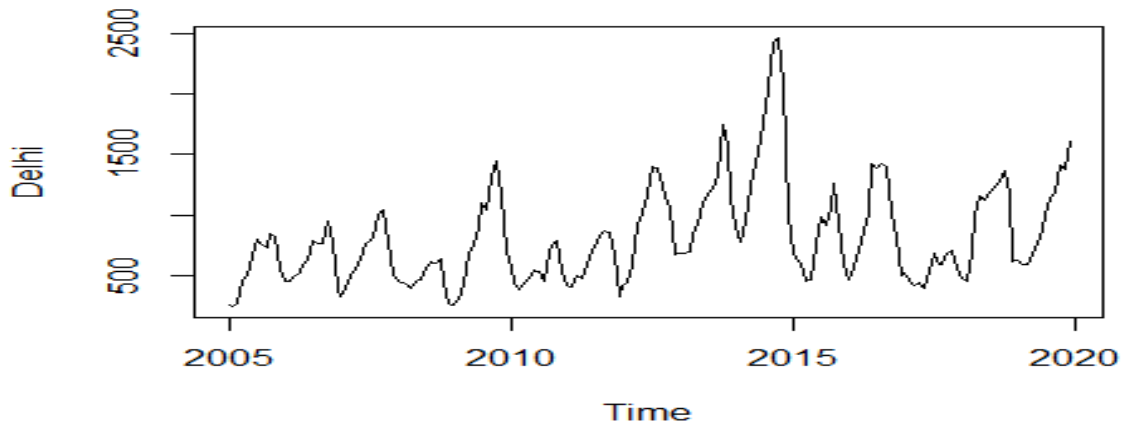
```
Agra <- ts(data[ ,"Agra"], frequency=12,  start=c(2005,1), end=c(2019, 12)
)
Delhi <- ts(data[ ,"Delhi"], frequency=12,  start=c(2005,1), end=c(2019, 1
2))
```

Time Plot of the data series

```
plot(Agra)
```




```
plot(Delhi)
```



Differenced Series

```
Agra_diff <- diff(Agra)
Delhi_diff <- diff(Delhi)
```

Test for Stationarity

The first step for the cointegration analysis is to check the order of integration of the series. The order of integration for the cointegrating series should be same. In order to check order of integration of the series, we used ADF test and the PP test. When we consider 'no drift and no trend' both the price series are nonstationary at level. But, when we applied ADF test and PP test in the differenced series then we found that both the series are stationary. In other words we can say that order of integration for the both series are same i.e. one.

ADF Test

```
adf.test(Agra)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -1.19  0.2517
## [2,]   1 -2.04  0.0419
## [3,]   2 -1.54  0.1262
## [4,]   3 -1.56  0.1181
## [5,]   4 -1.61  0.1020
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -3.22   0.022
## [2,]   1 -5.12   0.010
## [3,]   2 -4.15   0.010
## [4,]   3 -4.32   0.010
## [5,]   4 -4.70   0.010
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -3.42  0.0527
## [2,]   1 -5.59  0.0100
## [3,]   2 -4.58  0.0100
## [4,]   3 -4.85  0.0100
## [5,]   4 -5.35  0.0100
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(Delhi)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -0.839  0.3785
```

```
## [2,]   1 -1.840  0.0664
## [3,]   2 -1.418  0.1709
## [4,]   3 -1.386  0.1822
## [5,]   4 -1.136  0.2720
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -3.05  0.0349
## [2,]   1 -5.42  0.0100
## [3,]   2 -4.78  0.0100
## [4,]   3 -4.80  0.0100
## [5,]   4 -4.52  0.0100
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -3.32  0.0699
## [2,]   1 -5.90  0.0100
## [3,]   2 -5.26  0.0100
## [4,]   3 -5.42  0.0100
## [5,]   4 -5.14  0.0100
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(Agra_diff)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -8.71    0.01
## [2,]   1 -9.30    0.01
## [3,]   2 -7.55    0.01
## [4,]   3 -6.36    0.01
## [5,]   4 -7.16    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -8.69    0.01
## [2,]   1 -9.28    0.01
## [3,]   2 -7.53    0.01
## [4,]   3 -6.34    0.01
## [5,]   4 -7.14    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -8.66    0.01
## [2,]   1 -9.25    0.01
## [3,]   2 -7.51    0.01
## [4,]   3 -6.32    0.01
## [5,]   4 -7.12    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(Delhi_diff)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
```

```
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -8.19    0.01
## [2,]   1 -8.33    0.01
## [3,]   2 -7.24    0.01
## [4,]   3 -7.11    0.01
## [5,]   4 -7.23    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -8.18    0.01
## [2,]   1 -8.32    0.01
## [3,]   2 -7.23    0.01
## [4,]   3 -7.11    0.01
## [5,]   4 -7.22    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -8.16    0.01
## [2,]   1 -8.30    0.01
## [3,]   2 -7.21    0.01
## [4,]   3 -7.09    0.01
## [5,]   4 -7.20    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

PP Test

```
pp.test(Agra)
```

```
## Phillips-Perron Unit Root Test
## alternative: stationary
##
## Type 1: no drift no trend
##  lag Z_rho p.value
##    4 -5.48   0.115
## -----
##  Type 2: with drift no trend
##  lag Z_rho p.value
##    4 -29.6    0.01
## -----
##  Type 3: with drift and trend
##  lag Z_rho p.value
##    4 -34.7    0.01
## ---------------
## Note: p-value = 0.01 means p.value <= 0.01
```

```
pp.test(Delhi)
```

```
## Phillips-Perron Unit Root Test
## alternative: stationary
##
## Type 1: no drift no trend
##  lag Z_rho p.value
##    4  -4.9   0.162
## -----
##  Type 2: with drift no trend
##  lag Z_rho p.value
```

```
##    4 -32.9    0.01
## -----
##  Type 3: with drift and trend
##  lag Z_rho p.value
##    4 -37.8    0.01
## ---------------
## Note: p-value = 0.01 means p.value <= 0.01
```

```
pp.test(Agra_diff)
```

```
## Phillips-Perron Unit Root Test
## alternative: stationary
##
## Type 1: no drift no trend
##  lag Z_rho p.value
##    4 -95.9    0.01
## -----
##  Type 2: with drift no trend
##  lag Z_rho p.value
##    4 -95.9    0.01
## -----
##  Type 3: with drift and trend
##  lag Z_rho p.value
##    4 -95.9    0.01
## ---------------
## Note: p-value = 0.01 means p.value <= 0.01
```

```
pp.test(Delhi_diff)
```

```
## Phillips-Perron Unit Root Test
## alternative: stationary
##
## Type 1: no drift no trend
##  lag Z_rho p.value
##    4   -92    0.01
## -----
##  Type 2: with drift no trend
##  lag Z_rho p.value
##    4 -92.1    0.01
## -----
##  Type 3: with drift and trend
##  lag Z_rho p.value
##    4   -92    0.01
## ---------------
## Note: p-value = 0.01 means p.value <= 0.01
```

Engle-Granger: Long-run relationship of Delhi and Agra Potato market

To test the Engle-Granger long-run relationships for the Agra and Delhi potato price series, first of all both the price series are considered separately as endogenous variables, are simply estimated by ordinary least-squares (OLS). The residuals of these long-run relationships are stored as objects error.Agra and error.Delhi. An augmented Dickey-Fuller (ADF)-type test is applied to the residuals of each equation (coint.Agra and coint.Delhi) for testing whether the

variables are cointegrated or not. The test results indicate that both the error series are stationary.

```
data_EG <- window(cbind(Delhi, Agra), freq=12, start=c(2005, 3))
Agra.eq <- lm(Agra~Delhi)
summary(Agra.eq)

##
## Call:
## lm(formula = Agra ~ Delhi)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -460.44  -50.52    9.03   71.76  285.33
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.90279   20.98948  -0.234    0.816
## Delhi        0.82973    0.02307  35.958   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 124.7 on 178 degrees of freedom
## Multiple R-squared:  0.879,  Adjusted R-squared:  0.8783
## F-statistic:  1293 on 1 and 178 DF,  p-value: < 2.2e-16

Delhi.eq <- lm(Delhi~Agra)
summary(Delhi.eq)

##
## Call:
## lm(formula = Delhi ~ Agra)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -302.58  -92.13  -26.96   53.09  601.34
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 103.87856   22.40629   4.636 6.85e-06 ***
## Agra          1.05937    0.02946  35.958  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 140.9 on 178 degrees of freedom
## Multiple R-squared:  0.879,  Adjusted R-squared:  0.8783
## F-statistic:  1293 on 1 and 178 DF,  p-value: < 2.2e-16

error.Agra <- ts(resid(Agra.eq), start = c(2005, 3), end = c(2019, 12),
                 frequency = 12)
error.Delhi <- ts(resid(Delhi.eq), start = c(2005, 3), end = c(2019, 12),
                  frequency = 12)
coint.Agra <- adf.test(error.Agra)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -6.07    0.01
## [2,]   1 -6.30    0.01
## [3,]   2 -5.05    0.01
## [4,]   3 -4.78    0.01
## [5,]   4 -4.58    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -6.05    0.01
## [2,]   1 -6.28    0.01
## [3,]   2 -5.02    0.01
## [4,]   3 -4.75    0.01
## [5,]   4 -4.55    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -6.18    0.01
## [2,]   1 -6.53    0.01
## [3,]   2 -5.29    0.01
## [4,]   3 -5.06    0.01
## [5,]   4 -4.89    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01

Coint.Delhi <- adf.test(error.Delhi)

## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -6.19    0.01
## [2,]   1 -6.71    0.01
## [3,]   2 -5.78    0.01
## [4,]   3 -5.82    0.01
## [5,]   4 -5.63    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -6.16    0.01
## [2,]   1 -6.69    0.01
## [3,]   2 -5.75    0.01
## [4,]   3 -5.79    0.01
## [5,]   4 -5.60    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -6.15    0.01
## [2,]   1 -6.72    0.01
## [3,]   2 -5.82    0.01
## [4,]   3 -5.89    0.01
## [5,]   4 -5.74    0.01
```

```
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

Engle-Granger: ECM for Agra and Delhi Potato market

In the next step, the error-correction models (ECMs) for the Agra and Delhi potato markets are specified. For this, the necessary first differences of the series and its lagged values are created, as well as the series for the error term lagged by one period.

If two series are cointegrated, then there should be Granger-causation in at least one direction. That is, at least one coefficient of the error term should be significant and with the negative sign. The coefficient of the error-correction term (error.ecm1) in the error correction model (ECM) ('ecm.eq1') is -0.032 and is not significant, which means potato price of Delhi market does not Granger cause to the Agra market. On the contrary, the error-correction term (error.ecm2) in the ECM ('ecm.eq2') is -0.28 and is significant. In other words we can say, potato price of Agra market Granger cause to the Delhi market.

```
data_ECM <- ts(embed(diff(data_EG), dim=2), freq=12, start=c(2005, 5))
colnames(data_ECM) <- c('Delhi.d', 'Agra.d', 'Delhi.d1', 'Agra.d1')
head(data_ECM)
```

```
##      Delhi.d Agra.d Delhi.d1 Agra.d1
## [1,]      43    -14      192    94.5
## [2,]     162    -13       43   -14.0
## [3,]     145     26      162   -13.0
## [4,]     -56    -21      145    26.0
## [5,]     -23      5      -56   -21.0
## [6,]     118    228      -23     5.0
```

```
error.ecm1 <- window(lag(error.Agra, k=-1), start=c(2005, 5), end=c(2019,
12))
error.ecm2 <- window(lag(error.Delhi, k=-1), start=c(2005, 5), end=c(2019,
12))
ecm.eq1 <- lm(Agra.d~error.ecm1+Agra.d1+Delhi.d1, data = data_ECM)
summary(ecm.eq1)
```

```
##
## Call:
## lm(formula = Agra.d ~ error.ecm1 + Agra.d1 + Delhi.d1, data = data_ECM)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -864.93  -60.02   19.25   73.72  250.67
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.62302    9.73472  -0.064   0.9490
## error.ecm1  -0.03292    0.08886  -0.370   0.7115
## Agra.d1     -0.23869    0.10443  -2.286   0.0235 *
## Delhi.d1     0.68941    0.08992   7.667 1.24e-12 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 128.9 on 172 degrees of freedom
## Multiple R-squared:  0.3878, Adjusted R-squared:  0.3772
## F-statistic: 36.32 on 3 and 172 DF,  p-value: < 2.2e-16
```

```
ecm.eq2 <- lm(Delhi.d~error.ecm2+Agra.d1+Delhi.d1, data = data_ECM)
summary(ecm.eq2)
```

```
##
## Call:
## lm(formula = Delhi.d ~ error.ecm2 + Agra.d1 + Delhi.d1, data = data_ECM
)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -843.80  -72.01   19.53   91.84  342.58
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.8644    12.5140   0.149  0.88174
## error.ecm2   -0.2869     0.1055  -2.720  0.00719 **
## Agra.d1      -0.1553     0.1326  -1.171  0.24309
## Delhi.d1      0.4990     0.1187   4.203 4.22e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 165.7 on 172 degrees of freedom
## Multiple R-squared:  0.2493, Adjusted R-squared:  0.2362
## F-statistic: 19.04 on 3 and 172 DF,  p-value: 1.041e-10
```

Johanson Test of Cointegration

Johansen and Juselius start by determining the cointegration rank. Before the results of these tests are discussed, the arguments of the function ca.jo() should be presented briefly. The test type is either eigen or trace for the maximal eigenvalue statistic or the trace statistic, respectively, where the default is the former. Whether no deterministic term, a constant, or a trend should be included in the cointegration relations can be set by the argument 'ecdet'. The decision as to whether the long-run or transitory form of the VECM should be estimated is determined by the argument 'spec'. The default is spec="longrun". The inclusion of centered seasonal dummy variables can be set by providing the corresponding seasonality as an integer; e.g., season = 12 for monthly data.

Trace statistics test results indicate that for the cointegrating rank r=0 test statistics is 39.51 and significant, and for r=1 the test statistics is 8.15 and not significant, which means both the price series are cointegrated. The similar results are found in the eigen value statistics.

```
data_coint <- cbind(Agra, Delhi)
```

Trace Statistics

```
coint_trace <- ca.jo(data_coint, type = 'trace', K=2, # The lag order of t
he series (levels) in the VAR
                     season= 12, # If seasonal dummies should be included,
the data frequency must be set accordingly, i.e '4' for quarterly data
                     ecdet= 'none', # Character, 'none' for no intercept i
n cointegration, 'const' for constant term in cointegration and 'trend' fo
r trend variable in cointegration.
                     spec="longrun")

summary(coint_trace)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.16150106 0.04477755
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  8.15  6.50  8.18 11.65
## r = 0  | 39.51 15.66 17.95 23.52
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##             Agra.l2 Delhi.l2
## Agra.l2   1.0000000  1.00000
## Delhi.l2 -0.8844356 -2.23302
##
## Weights W:
## (This is the loading matrix)
##
##            Agra.l2   Delhi.l2
## Agra.d  -0.4285400 0.02245229
## Delhi.d -0.1575856 0.06047030
```

Eigen value Statistics

```
coint_eigen <- ca.jo(data_coint, type = 'eigen', K=2, # The lag order of t
he series (levels) in the VAR
                     season= 12, # If seasonal dummies should be included,
the data frequency must be set accordingly, i.e '4' for quarterly data
                     ecdet= 'none', # Character, 'none' for no intercept i
n cointegration, 'const' for constant term in cointegration and 'trend' fo
r trend variable in cointegration.
                     spec="longrun")

summary(coint_eigen)
```

```
## ######################
## # Johansen-Procedure #
```

```
## ######################
## Test type: maximal eigenvalue statistic (lambda max) , with linear tren
d
## Eigenvalues (lambda):
## [1] 0.16150106 0.04477755
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  8.15  6.50  8.18 11.65
## r = 0  | 31.35 12.91 14.90 19.19
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##             Agra.l2 Delhi.l2
## Agra.l2   1.0000000  1.00000
## Delhi.l2 -0.8844356 -2.23302
##
## Weights W:
## (This is the loading matrix)
##
##            Agra.l2   Delhi.l2
## Agra.d  -0.4285400 0.02245229
## Delhi.d -0.1575856 0.06047030
```

References

Engle, R. F. and Granger, C. W. J. (1987). Cointegration and error correction: Representation, estimation, and testing. *Econometrica*, **55**, 251-276.

Granger, C. J. (1981). Some properties of time series data and their use in econometric model specification. *Journal of Econometrics*, **16**, 121-130.

Jha, G.K. (2013). Energy growth linkage and strategy for meeting the energy demand in Indian agriculture. *Agricultural Economics Research Review*, **26**, 119-127.

Johansen, S. (1988). Statistical analysis of cointegration vectors. *Journal of Economic Dynamics and Control*, **12**, 231-254.

Kumar, R.R. and Jha, G.K. (2017). Examining the co-movement between energy and agricultural commodity prices in India. *Journal of the Indian Society of Agricultural Statistics*, **70**(3), 241-252.

Kumar, R.R., Jha, G.K., Choudhary, K. and Mishra, D.C. (2020). Spatial integration and price transmission among major potato markets in India. *Indian Journal of Agricultural Sciences*, **90**(3), 581-584.

Pfaff, B. (2008). *Analysis of Integrated and Cointegrated Time Series with R.* Springer Science and Business Media LLC.

Phillip, P. C. B. and Perron, P. (1988). Testing of unit root in time series regression. *Biometrika,***75**, 335-346.

Sundaramoorthy, C., Jha, G.K., Pal, S. and Mathur V.C. (2014). Market integration and volatility in edible oil sector in India. *Journal of the Indian Society of Agricultural Statistics,* **68**(1), 67-76.

Artificial Neural Networks: An Introduction

Girish Kumar Jha
ICAR-Indian Agricultural Research Institute, New Delhi-110 012
girish.stat@gmail.com

Introduction

The seed of the modern era of neural networks was sown with the pioneering work of McCulloch and Pitts in 1943 when they introduced the idea of neural networks as computing machines. The major impetus to the growth and development of research and applications in the area of neural networks was provided by the development of the back-propagation algorithm, a popular learning algorithm for the training of multilayer perceptrons by Rumelhart, Hinton and Williams (1986). However, the recent resurgence of interest in this area is mainly because neural networks are the fundamental building block of deep learning, which is an artificial intelligence function that allows computational models to learn features from raw data with multiple levels of abstraction. Presently, artificial intelligence (AI) is a growing field and has many practical applications due to the availability of massive data, graphics processing units (GPUs) hardware and open source software like python and tensorflow etc.

Artificial neural networks (ANNs) are non-linear data driven self-adaptive approach as opposed to the traditional model-based methods. They are powerful tools for modelling, especially when the underlying data relationship is unknown. ANNs can identify and learn correlated patterns between input data sets and corresponding target values. After training, ANNs can be used to predict the outcome of new independent input data. ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. Thus, they are ideally suited for the modeling of agricultural data which are known to be complex and often non-linear. In recent years neural computing has emerged as a practical technology, with successful applications in many fields as diverse as finance, medicine, engineering, geology, physics and biology. The excitement stems from the fact that these networks are attempts to model the capabilities of the human brain. From a statistical perspective neural networks are interesting because of their potential use in prediction and classification problems.

A very important feature of these networks is their adaptive nature, where “learning by example” replaces “programming” in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is readily available. These

networks are "neural" in the sense that they may have been inspired by neuroscience but not necessarily because they are faithful models of biological neural or cognitive phenomena. In fact, majority of the network are more closely related to traditional mathematical and/or statistical models such as non-parametric pattern classifiers, clustering algorithms, nonlinear filters, and statistical regression models than they are to neurobiology models.

Neural networks (NNs) have been used for a wide variety of applications where statistical methods are traditionally employed. They have been used in classification problems, such as identifying underwater sonar currents, recognizing speech, and predicting the secondary structure of globular proteins. In time-series applications, NNs have been used in predicting stock market performance. As statisticians or users of statistics, these problems are normally solved through classical statistical methods, such as discriminant analysis, logistic regression, Bayes analysis, multiple regression, and ARIMA time-series models. It is, therefore, time to recognize neural networks as a powerful tool for data analysis.

Basics of a neuron

An artificial neural network is a set of simple computational units that are highly interconnected. The units are also called nodes and loosely represent the biological neuron. A graphical presentation of neuron is given in Figure 1. A neuron is an information processing unit that is fundamental to the operation of a neural network. The connections between nodes are unidirectional and are represented by arrows in the figure. These connections model the synaptic connections in the brain. Each connection has a weight called the synaptic weight, denoted as w_{kj}, associated with it. The synaptic weight, w_{kj}, is interpreted as the strength of the connection from the *j*th unit to the *k*th unit. Unlike a synapse in the brain, the synaptic weight of an artificial neuron may lie in a range that includes negative as well as positive values. If a weight is negative, it is termed inhibitory because it decreases the net input. If the weight is positive, the contribution is excitatory because it increases the net input.

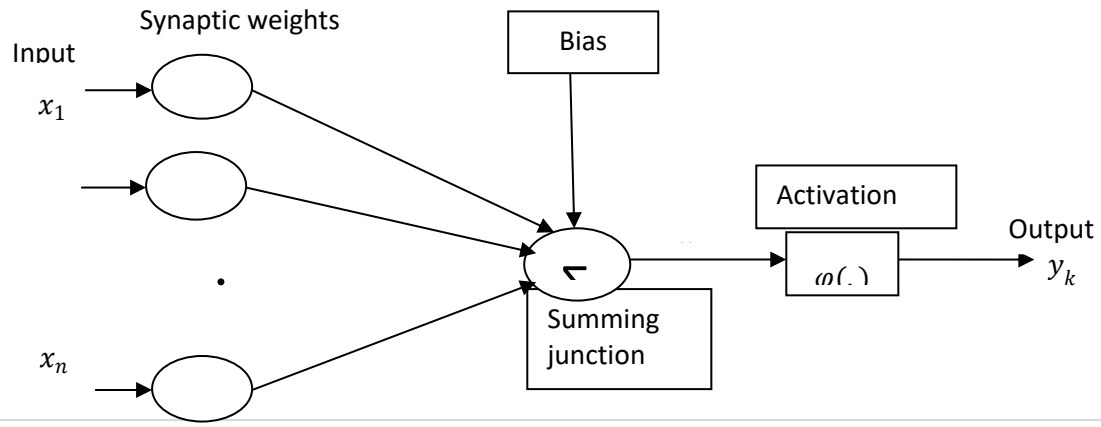


Figure 1: Nonlinear model of a neuron

The input into a node is a weighted sum of the outputs from nodes connected to it. Each unit takes its net input and applies an activation function to it. The neuronal model of Figure 1 also includes an externally applied bias, denoted by b_k. The bias b_k has the effect of increasing or lowering the net input of the activation function depending on whether it is positive or negative respectively. In mathematical terms, we may describe a neuron *k* by the following equations

$$y_k = \varphi(v_k) = \varphi\left(\sum_{j=1}^{n} w_{kj}x_j + b_k\right)$$

where $x_1, x_2, \ldots, x_n$ are the input patterns, $w_{k1}, w_{k2}, \ldots, w_{kn}$ are the synaptic weights of neuron k, b_k is the bias, $\varphi(.)$ is the activation function and y_k is the output of the neuron. The neural networks are built from layers of neurons connected so that one layer receives input from the preceding layer of neurons and passes the output on to the subsequent layer.

Types of activation function

An activation function which is also known as squashing function, squashes or limits the amplitude range of the output of a neuron. It is a mathematical function which converts the input to an output, and adds the magic of neural network processing. The abstraction of the processing of neural networks is mainly achieved through the activation functions. Activation functions give the nonlinearity property to neural networks and make them true universal function approximators. Three commonly used activation functions are described below:

a. **Sigmoid Function**: The term sigmoid means S-shaped and the logistic form of the sigmoid maps the interval (-∞, ∞) onto (0, 1). The main motivation of using this activation function is allowing the outputs to be given a probabilistic interpretation. It is defined by

$$f(x) = \frac{1}{(1 + e^{-ax})}$$

where *a* is the slope parameter of the sigmoid function and is illustrated in Figure 2.

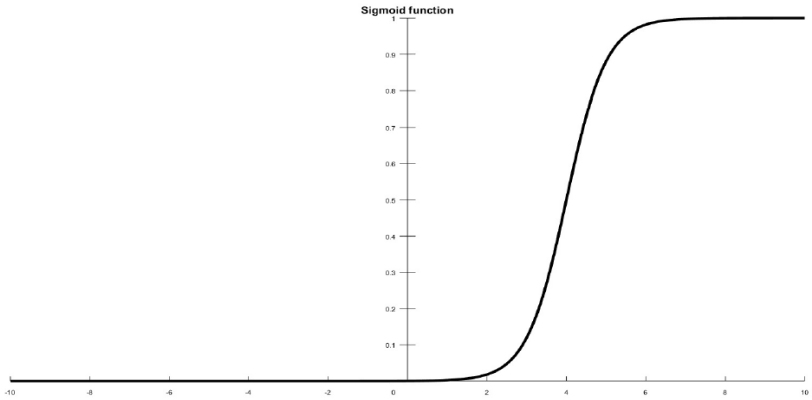


Figure 2: Sigmoid function

b. **Hyperbolic Tangent**: This is a nonlinear function, defined in the range of values (-1, 1) and is plotted in Figure 3. This function is defined by

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

It is established empirically that tanh function provided faster convergence of training algorithms than logistic function. Both the logistic and hyperbolic tangent functions differ only through a linear transformation. These two were the most common form of activation functions used in the construction of neural networks prior to the introduction of rectified linear units.

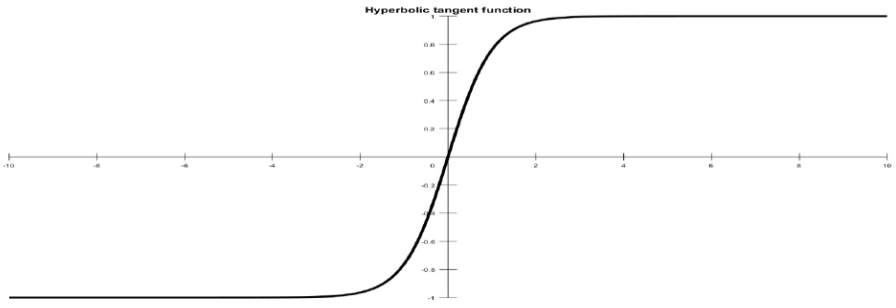


Figure 3: Hyperbolic tangent function

c. **Rectified Linear Unit (ReLU):** It is the most used activation function since 2015. It is a simple condition and has advantages over the other functions. The function is defined by the following formula and is plotted in Figure 4:

$$f(x) = \begin{cases} 0 & when\ x < 0 \\ x & when\ x \geq 0 \end{cases}$$

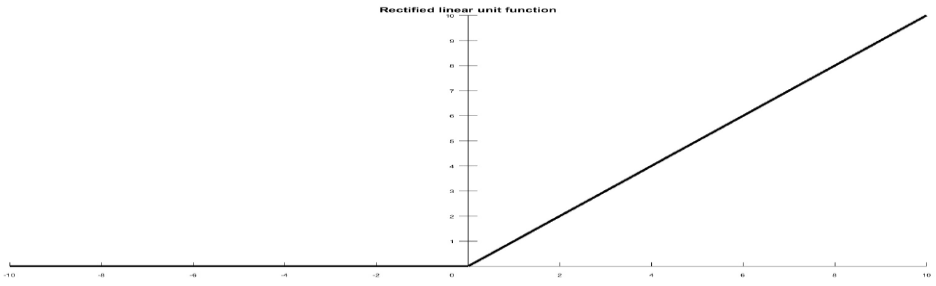


Figure 4: Rectified Linear Unit

Neural networks architectures

An artificial neural network is defined as a data processing system consisting of a large number of simple highly inter connected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex of the brain. There are several types of architecture of neural networks. However, the two most widely used ANNs are discussed below:

Feed forward networks

In a feed forward network, information flows in one direction along connecting pathways, from the input layer via the hidden layers to the final output layer. There is no feedback (loops) i.e., the output of any layer does not affect that same or preceding layer.

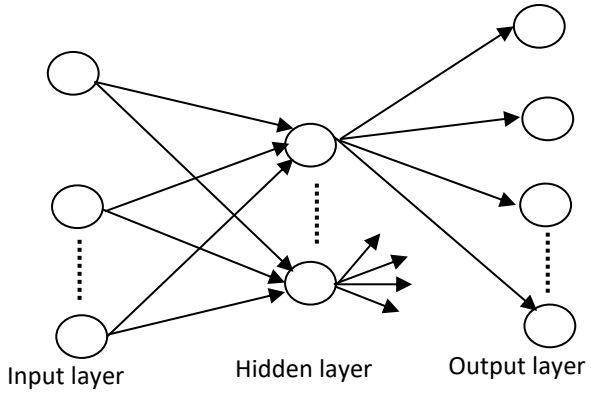


Figure 5: A multi-layer feed forward neural network

Recurrent networks

These networks differ from feed forward network architectures in the sense that there is at least one feedback loop. Thus, in these networks, for example, there could exist one layer with feedback connections as shown in figure below. There could also be neurons with self-feedback

links, i.e. the output of a neuron is fed back into itself as input.

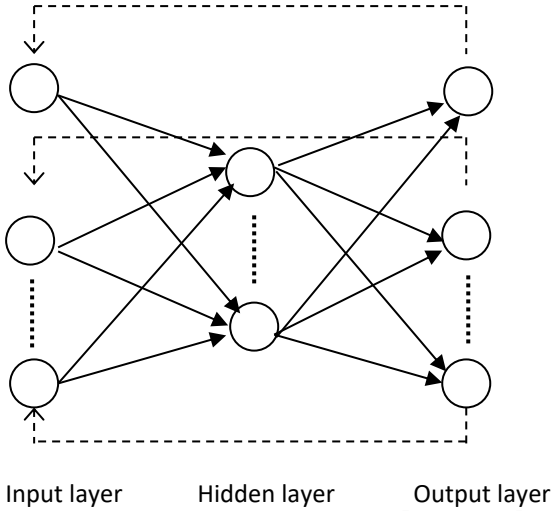


Figure 6: A recurrent neural network

Learning/Training methods

Learning methods in neural networks can be broadly classified into three basic types: supervised, unsupervised and reinforced.

Supervised learning

In this, every input pattern that is used to train the network is associated with an output pattern, which is the target or the desired pattern. A teacher is assumed to be present during the learning process, when a comparison is made between the network's computed output and the correct expected output, to determine the error. The error can then be used to change network parameters, which result in an improvement in performance.

Unsupervised learning

In this learning method, the target output is not presented to the network. It is as if there is no teacher to present the desired patterns and hence, the system learns of its own by discovering and adapting to structural features in the input patterns.

Reinforced learning

In this method, a teacher though available, does not present the expected answer but only indicates if the computed output is correct or incorrect. The information provided helps the network in its learning process. A reward is given for a correct answer computed and a penalty for a wrong answer. Reinforced learning was not one of the popular forms of learning but gaining importance in case of deep learning.

Development of an ANN model

The various steps in developing a neural network model are:

A. Variable selection

The input variables important for modeling variable(s) under study are selected by suitable variable selection procedures.

B. Formation of training, testing and validation sets

The data set is divided into three distinct sets called training, testing and validation sets. The training set is the largest set and is used by neural network to learn patterns present in the data. The testing set is used to evaluate the generalization ability of a supposedly trained network. A final check on the performance of the trained network is made using validation set.

C. Neural network architecture

Neural network architecture defines its structure including number of hidden layers, number of hidden nodes and number of output nodes etc.

- Number of hidden layers: The hidden layer(s) provide the network with its ability to generalize. In theory, a neural network with one hidden layer with a sufficient number of hidden neurons is capable of approximating any continuous function. In practice, neural network with one and occasionally two hidden layers are widely used and have to perform very well.
- Number of hidden nodes: There is no magic formula for selecting the optimum number of hidden neurons. However, some thumb rules are available for calculating number of hidden neurons like for a three layers network with n input and m output neurons, the hidden layer would have *sqrt(n*m)* neurons.
- Number of output nodes: Neural networks with multiple outputs, especially if these outputs are widely spaced, will produce inferior results as compared to a network with a single output.
- Activation function: As mentioned earlier, activation functions are mathematical formulae that determine the output of a processing node. Each unit takes its net input and applies an activation function to it. The purpose of the transfer function is to prevent output from reaching very large value which can 'paralyze' neural networks and thereby inhibit training. Now the default recommendation is to use rectified linear units as

activation function for network learning.

D. Evaluation criteria

For regression problems, the sum of squares error function is commonly used in neural networks. The cross-entropy loss function is commonly used in case of classification problem.

E. Neural network training

Training a neural network to learn patterns in the data involves iteratively presenting it with examples of the correct known answers. The objective of training is to find the set of weights between the neurons that determine the global minimum of error function. This involves decision regarding the number of iterations i.e., when to stop training a neural network and the selection of learning rate (a constant of proportionality which determines the size of the weight adjustments made at each iteration) and momentum values (how past weight changes affect current weight changes).

Conclusion

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. A large number of claims have been made about the modeling capabilities of neural networks, some exaggerated and some justified. Hence, to best utilize ANNs for different problems, it is essential to understand the potential as well as limitations of neural networks. For some tasks, neural networks will never replace conventional methods, but for a growing list of applications, the neural architecture will provide either an alternative or a complement to these existing techniques. Finally, I would like to conclude that the performance of traditional/shallow neural networks depend on the feature of the data provided by the domain experts. This limitation inspired another subset of neural networks called deep neural networks (DNNs). DNNs extend traditional ANNs by adding multiple processing layers between input and output layers into the model that allows hierarchical representation of raw data through several layers of abstraction.

References

Cheng, B. and Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. Statistical Science, **9**, 2-54.

Jha, G.K., Thulasiraman, P. and Thulasiram, R. K. (2009). PSO based neural network for time series forecasting. In proceeding of the IEEE International Joint Conference on Neural Networks, USA, pp 1422-1427.

Jha, Girish Kumar and Sinha, K. (2014). Time-delay neural networks for time series prediction: An application to the monthly wholesale price of oilseeds in India. *Neural Computing and Applications*, 24(3-4): 563-571.

Jha, Girish Kumar and Sinha, K. (2013). Agricultural price forecasting using neural network model: An innovative information delivery system. *Agricultural Economics Research Review*, 26(2): 229-239.

Kaastra, I. and Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, **10**, 215-236.

Kohzadi, N., Boyd, S.M., Kermanshahi, B. and Kaastra, I. (1996). A comparision of artificial neural network and time series models for forecasting commodity prices. *Neurocomputing*, **10**, 169-181.

McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115-133.

Patterson, D. (1996). Artificial Neural Networks. Singapore: Prentice Hall.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage ang organization in the brain. *Psychological review*, **65,** 386-408.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning internal representation by error propagation. In D.E. Rumelhart, J.L. McClelland and the PDP Research Group (Eds.), *Parallel distributed processing: Exploration in microstructure of cognition*, Volume 1: Foundations, pp 318-362. Cambridge, MA: MIT Press.

Simon Haykin (2006). Neural Networks: A comprehensive foundation, Pearson Prentice Hall.

Saanzogni, Louis and Kerr, Don (2001) Milk production estimate using feed forward artificial neural networks. *Computer and Electronics in Agriculture*, **32**, 21-30.

Swanson, N. R., and White, H. (1997). Forecasting economic time series using adaptive versus non-adaptive and linear versus nonlinear economic models. *International Journal of Forecasting*, 13, 439–461.

Warner, B. and Misra, M. (1996). Understanding neural networks as statistical tools. *American Statistician*, **50**, 284-93.

Zhang, G., Patuwo, B. E. and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, **14**, 35-62.

Zhang, G.P. (2007). Avoiding pitfalls in neural network research. IEEE transactions on systems, man and cybernetics-Part C: Applications and reviews, **37**, 3-16.

Artificial Intelligence & Machine Learning for Classification in Agriculture

Manjunath Chikkamath

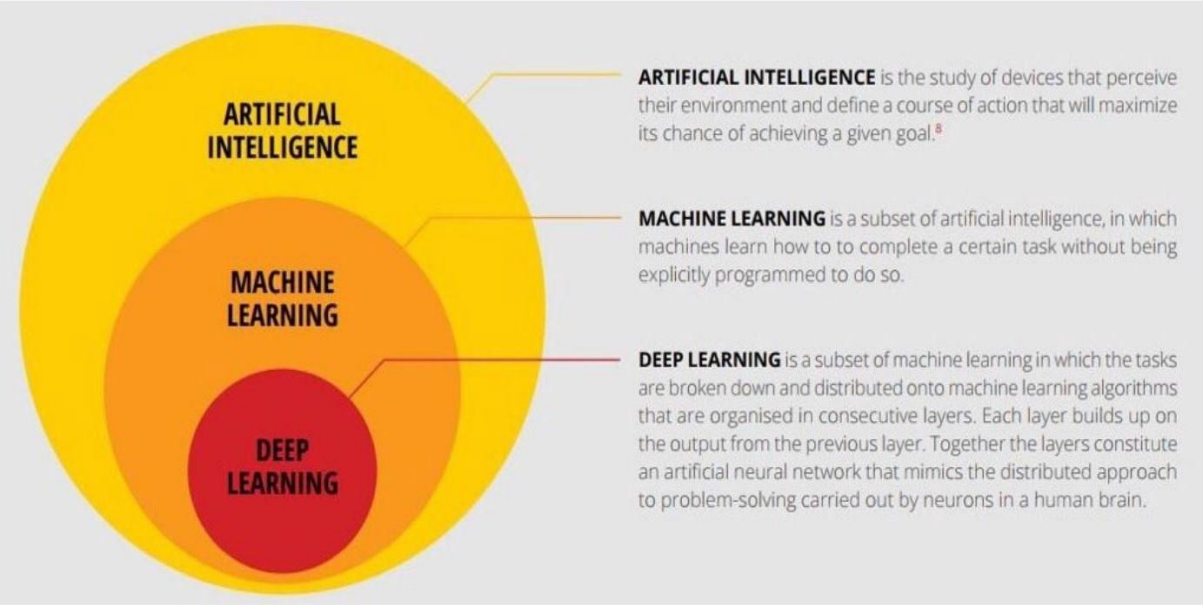
Robert Bosch Engineering and Business Solutions limited

Dr.chikkamath@gmail.com

What is Artificial Intelligence(AI)

Artificial intelligence (AI) is a branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence. These artificial intelligence systems are powered by machine learning, some of them are powered by deep learning and some of them are powered by very boring things like rules. AI initiates common sense, problem-solving and analytical reasoning power in machines, which is much difficult and a tedious job

- An intelligent entity created by humans
- Capable of performing tasks intelligently without being explicitly instructed
- Capable of thinking and acting rationally and humanely



Machine Learning

Artificial Intelligence and Machine Learning are much trending and confused terms nowadays. Artificial intelligence is a set of algorithms and intelligence, to try to mimic human intelligence. Machine learning is one of them, and deep learning is one of those machine learning techniques. A computer program is said to learn from experience (E) with respect to some task

(T) and some performance measure (P), if its performance on T, as measured by P, improves with experience E then the program is called a machine learning program - Tom Mitchell. The ML algorithms use Computer Science and Statistics to predict rational outputs

Algorithms and Learning techniques

Supervised learning techniques

- Linear classifier (numerical functions)
- Parametric (Probabilistic functions)
- Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
- Non-parametric (Instance-based functions)
- K-nearest neighbors, Classification and regression tree (CART), Decision tree
- Aggregation or Ensemble
- Bagging (bootstrap + aggregation), Boosting, Stacking

Unsupervised learning techniques

- Clustering
- K-means clustering
- Spectral clustering
- Density Estimation
- Gaussian mixture model (GMM)
- Graphical models
- Dimensionality reduction
- Principal component analysis (PCA)
- Factor analysis
- Reinforcement learning
- Decision making (robot, chess machine)

Deep Learning

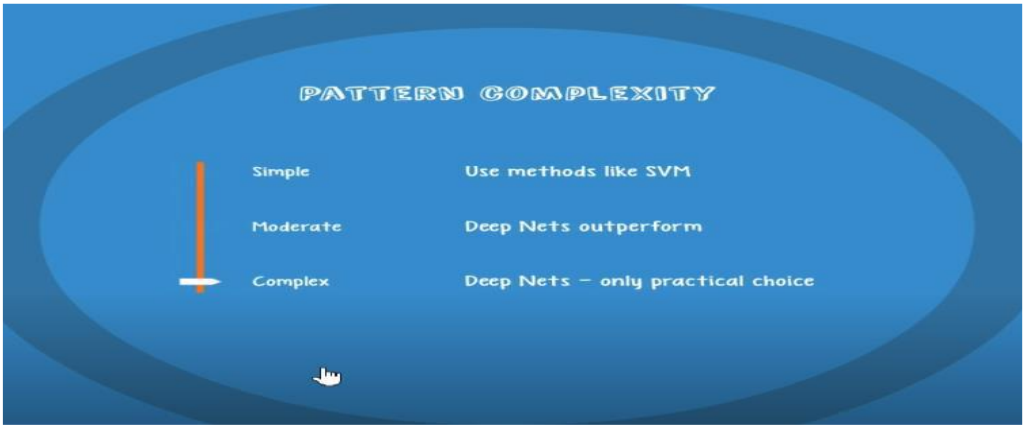
- Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by experience and example. Similarly, the deep learning algorithm would perform a task repeatedly, each time tweaking it a little to improve the outcome
- Deep learning deals with algorithms inspired by the structure and function of the brain

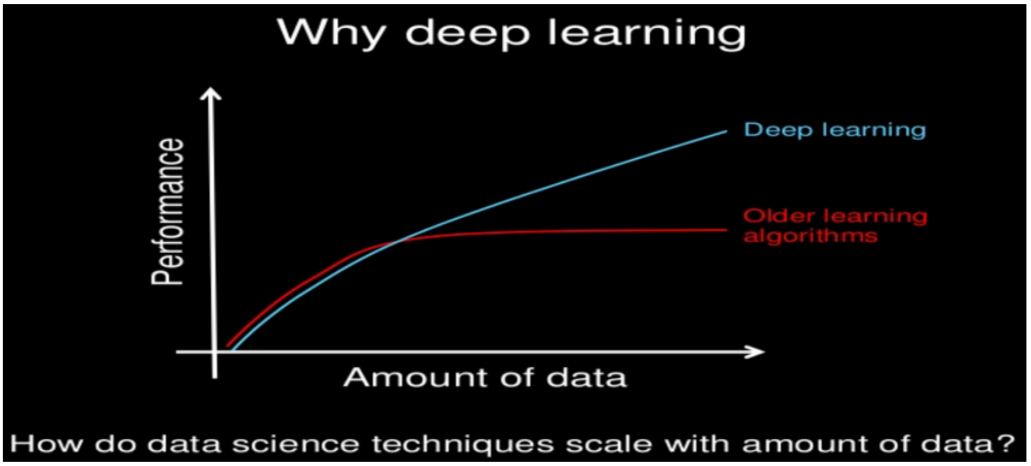
called artificial neural networks. It mirrors the functioning of our brains

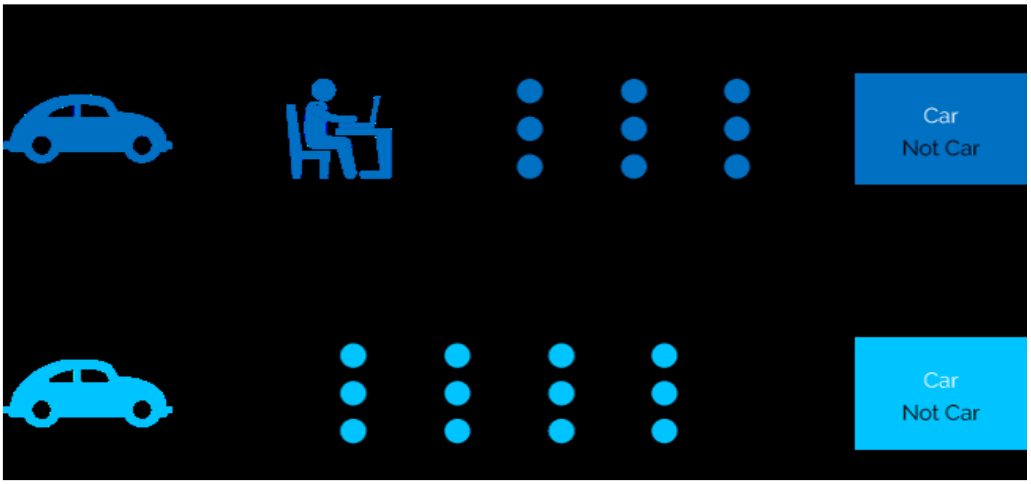
- Deep learning algorithms are like how nervous system structured where each neuron connected each other and passing information
- Deep learning is the name we use for "stacked neural networks"; that is, networks composed of several layers - Deep learning is about neural network
- Deep Learning means using a neural network with several layers of nodes between input and output

Why deep learning ?

- Deep learning achieves accuracy at higher levels than ever before
- Deep learning outperforms humans in some tasks like classifying objects in images
- As the amount of data increases, the performance of traditional learning algorithms, like SVM and logistic regression, does not improve by a whole lot. In fact, it tends to plateau after a certain point. In the case of neural networks, the performance of the model increases with an increase in the data you feed to the model
- Feature extraction is done by human in machine learning whereas deep learning model figure out by itself
- The amount of data we generate every day is staggering—currently estimated at 2.6 quintillion bytes. This increase in data creation is one reason that deep learning capabilities have grown in recent years.
- Stronger computing power that's available today - Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.







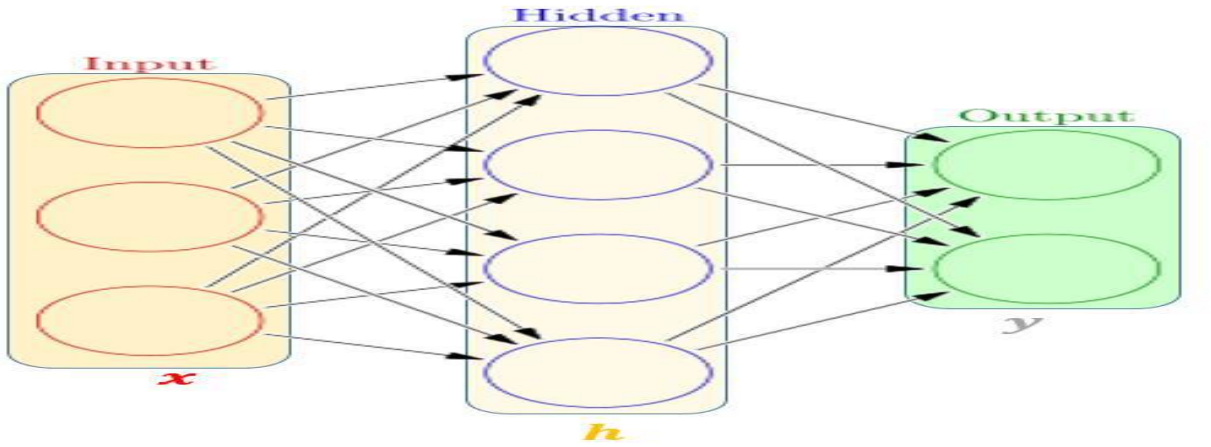
Deep Learning - Neural Network Representation

Weights

$\boldsymbol{h} = \boldsymbol{\sigma}(\boldsymbol{W1}\ \boldsymbol{x} + \boldsymbol{b1})$

$\boldsymbol{y} = \boldsymbol{\sigma}(\boldsymbol{W2h} + \boldsymbol{b2})$

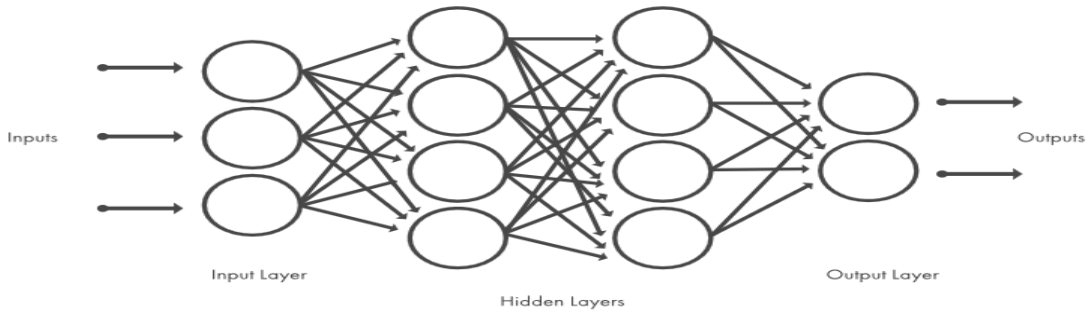
Activation functions

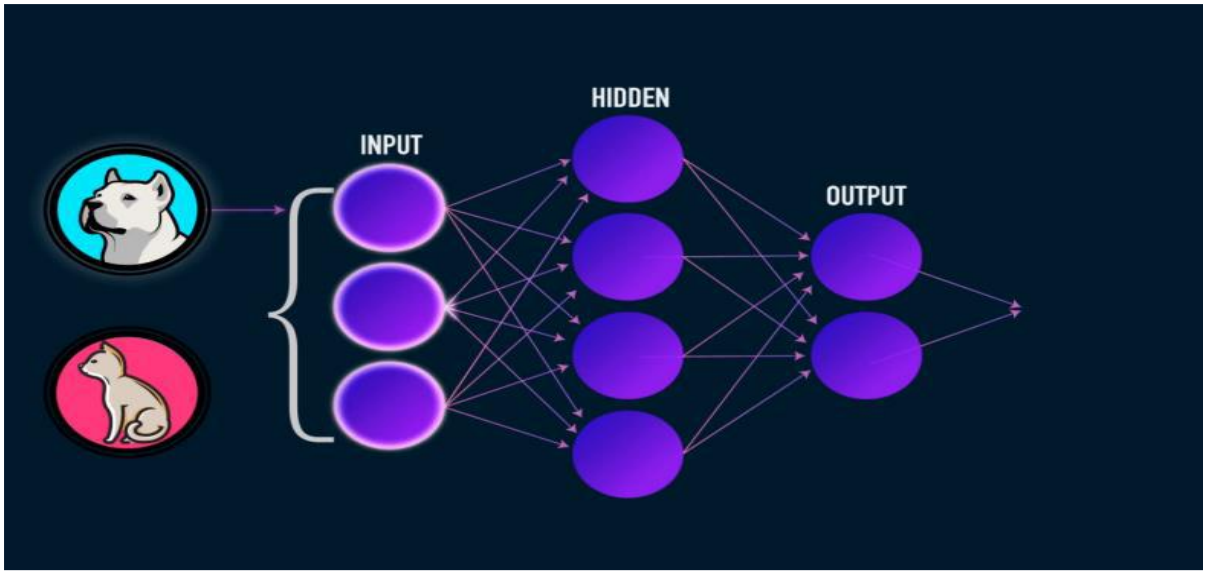


4 + 2 = 6 neurons (not counting inputs)
[3 x 4] + [4 x 2] = 20 weights
4 + 2 = 6 biases
26 learnable parameters

How deep learning works

- A neural network is composed of input, hidden, and output layers—all of which are composed of "nodes". Input layers take in a numerical representation of data (e.g. images with pixel specs), output layers output predictions, while hidden layers are correlated with most of the computation
- The major points to keep note of here are the tunable weight and bias parameters—represented by w and b respectively in the function. **These are essential to the actual "learning" process of a deep learning algorithm**
- After the neural network passes its inputs all the way to its outputs, the network evaluates how good its prediction was (relative to the expected output) through something called a loss function. As an example, the "Mean Squared Error" loss function
- The goal of network is ultimately to minimize this loss by adjusting the weights and biases of the network. In using something called "back propagation" through gradient descent.
- The network backtracks through all its layers to update the weights and biases of every node in the opposite direction of the loss function—in other words, every iteration of back propagation should result in a smaller loss function than before.
- The continuous updates of the weights and biases of the network ultimately turns it into a precise function approximator —one that models the relationship between inputs and expected outputs.

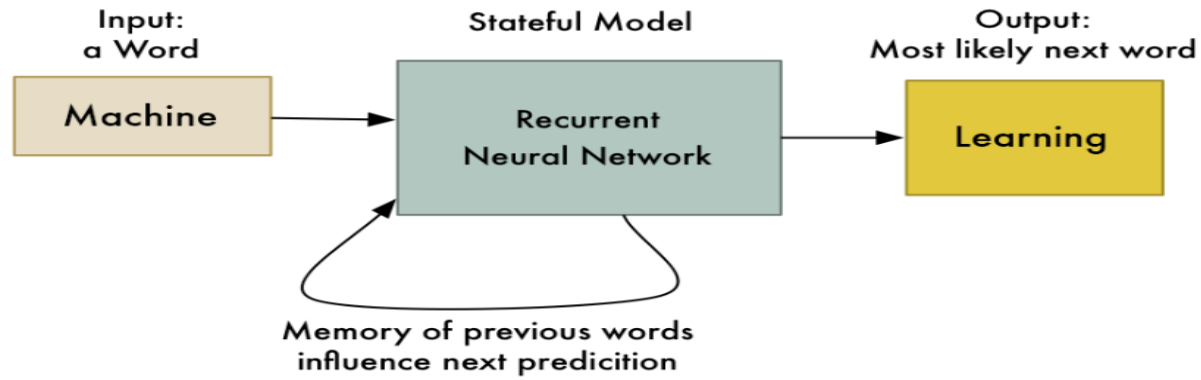




Types of Neural Networks

1. **Feedforward Neural Network** - Here data or the input travels in one direction. The data passes through the input nodes and exit on the output nodes. This neural network may or may not have the hidden layers. Used in computer vision and speech recognition.
2. **Radial Basis Function Neural Network** - Radial basic functions consider the distance of a point with respect to the center
3. **Multilayer Perceptron** - A multilayer perceptron has three or more layers. Every single node in a layer is connected to each node in the following layer. A multilayer perceptron uses a nonlinear activation function (mainly hyperbolic tangent or logistic function)
4. **Modular Neural Network -** A modular neural network has a number of different networks that function independently and perform sub-tasks. A large and complex computational process can be done significantly faster by breaking it down into independent components
5. **Recurrent Neural Network(RNN)** – Long Short Term Memory
    - Works on the principle of saving the output of a layer and feeding this back to the input to help in predicting the outcome of the layer
    - Here, the first layer is formed similar to the feed forward neural network with the product of the sum of the weights and the features
    - The recurrent neural network process starts once this is computed, this means, from one time step to the next each neuron will remember some information it had in the previous time-step. This makes each neuron act like a memory cell in performing computations. In this process, neural network works on the front propagation and remember what information it needs for later use

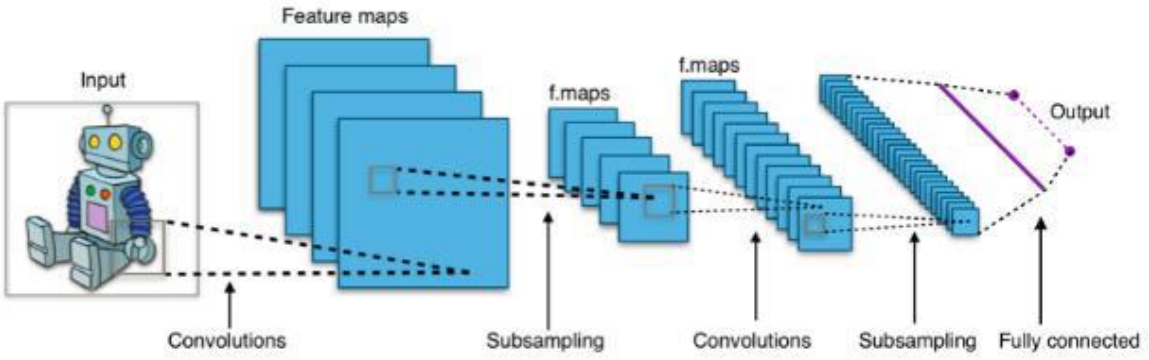
- Here, if the prediction is wrong we use the learning rate or error correction to make small changes so that it will gradually work towards making the right prediction during the back propagation
- It is used in text to speech(TTS) conversion models. Natural language processing – Entity tagging, Time Series Forecasting

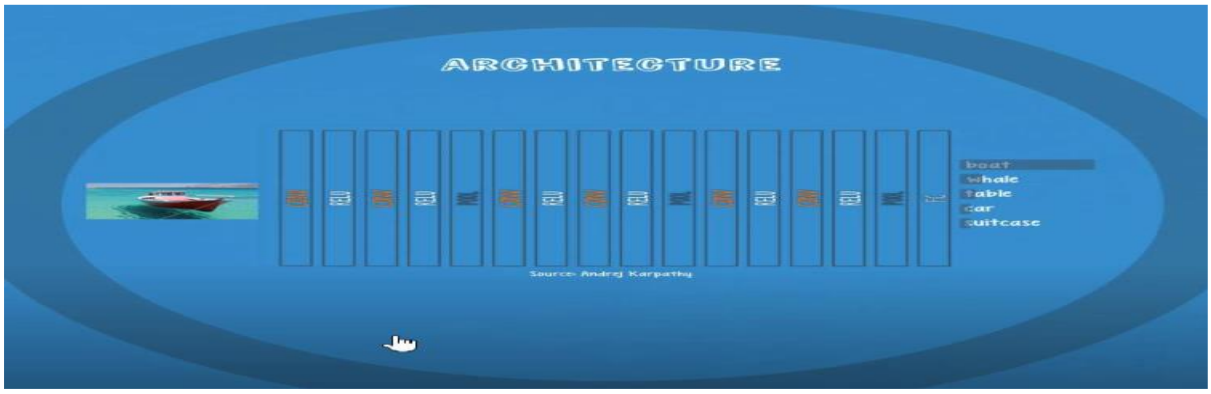


Output so far:
Machine Learning

6. Convolutional Neural Network(CNN)

- In a convolutional neural network the unit connectivity pattern is inspired by the organization of the visual cortex. Units respond to stimuli in a restricted region of space known as the receptive field
- Receptive fields partially overlap, over-covering the entire visual field
- CNN uses a variation of the multilayer perceptrons
- A CNN contains one or more than one convolutional layers. These layers can either be completely interconnected or pooled
- In this neural network, the input features are taken in batch wise like a filter. This will help the network to remember the images in parts and can compute the operations. These computations involve conversion of the image from RGB or HSI scale to Gray-scale. Once we have this, the changes in the pixel value will help detecting the edges and images can be classified into different categories
- Used in image and video recognition and Signal processing





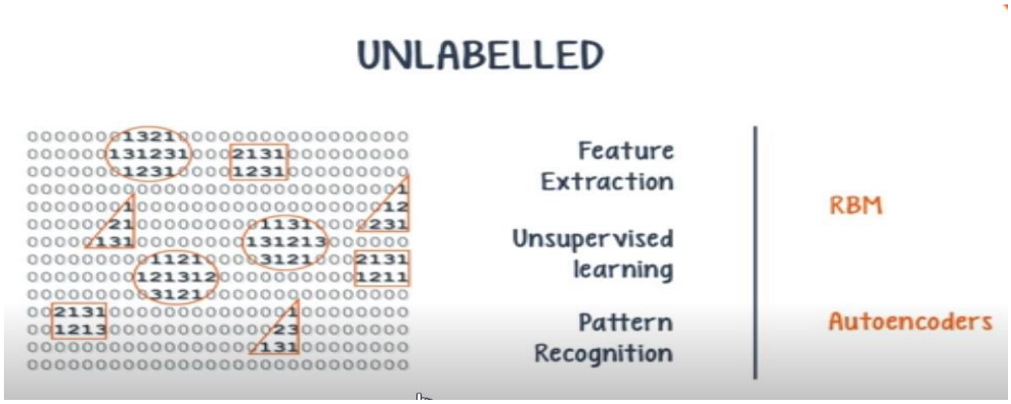
Choice of Deep Neural Network

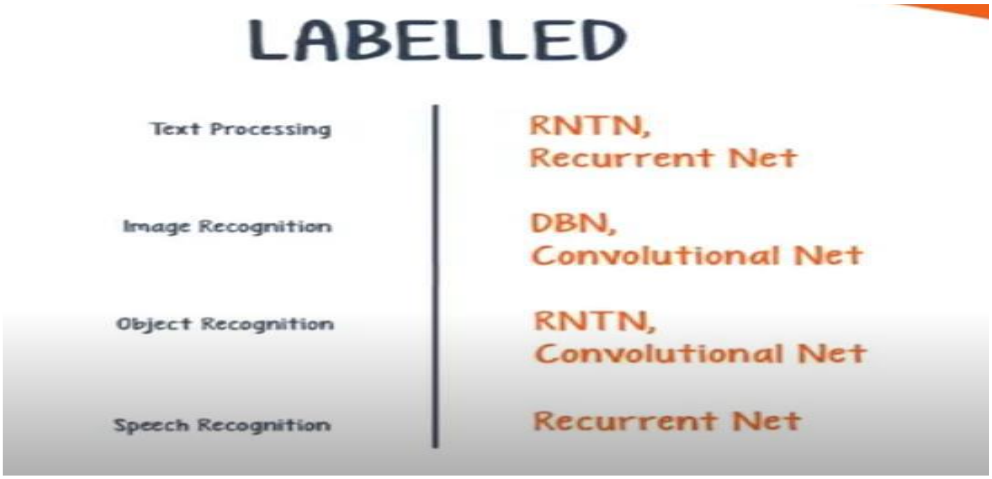
RBM- Restricted Boltzmann Machine

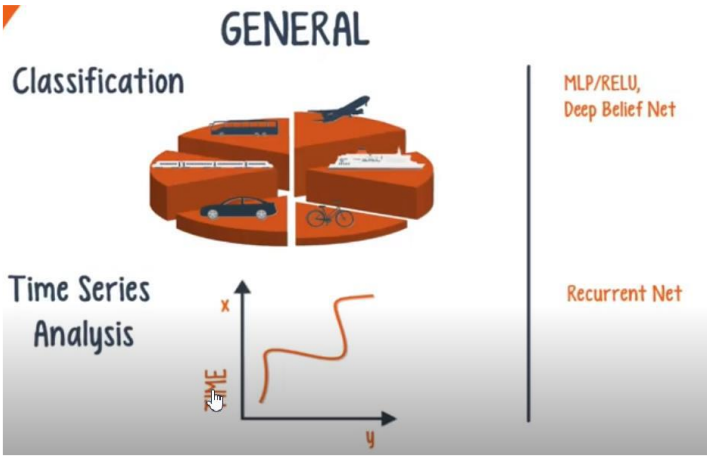
RNTN – Recursive Neural Tensor Network

NER, Sentiment analysis etc

DBN – Deep Belief Network







Deep Learning Platforms and Libraries

Platforms – Lot are available

- Ersatz Labs
- H2o.ai
- Dato Graphlab

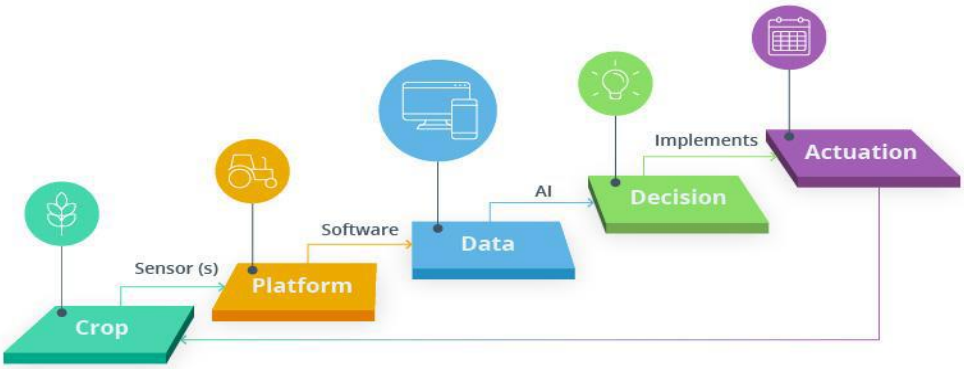
Libraries

- Theono
- Caffe
- Tensorflow
- Torch
- Deeplearning4j

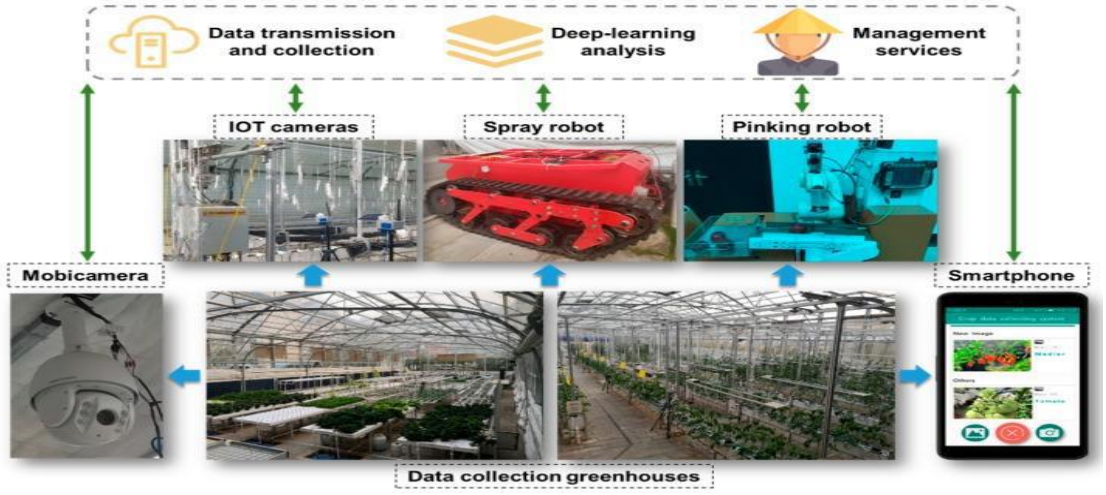
AI in Agriculture

Forms

- Robot
- Drone
- Apps- Mobile apps

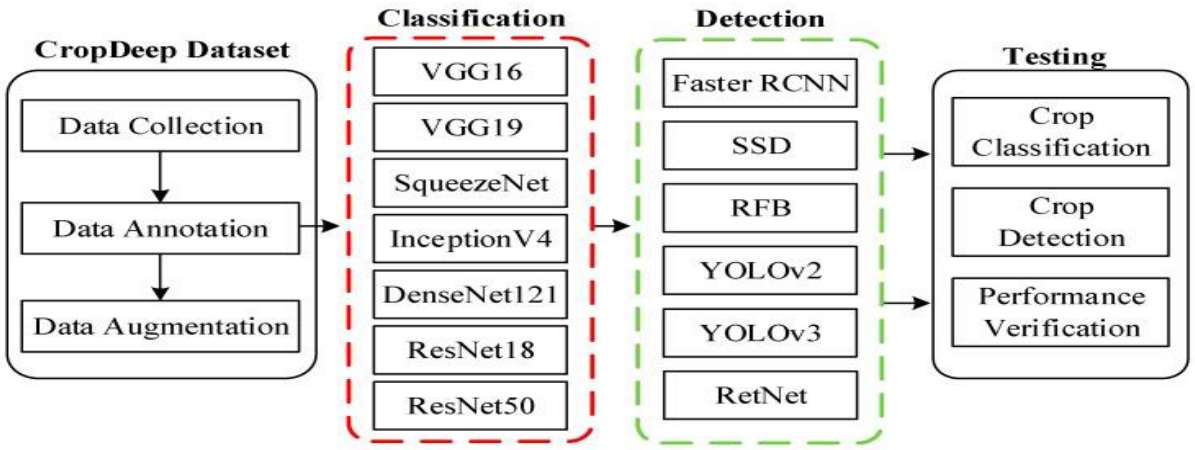


Data Collection



Data Annotation

Training Model and Validation



Model Implementation

Industry Application of AI and ML in Agriculture

- Weed Control - Blue River Technology has developed a robot called See & Spray which reportedly leverages computer vision to monitor and precisely spray weeds on cotton plants.
- The company claims that its precision technology eliminates 80 percent of the volume of chemicals normally sprayed on crops and can reduce herbicide expenditures by 90 percent

Monitoring soil and crop in real-time - An AI-based application called Plantix uses image recognition-based technology that helps farmers identify nutrient deficiencies in soil, plant pests and other diseases. With the help of Plantix, farmers can easily figure out which fertilizer to use in order to improve the quality of the harvest. The app also provides tips and solutions for the detected problems.

Using drones for data collection - PrecisionHawk allows farmers to virtually walk around their fields with the help of drones. Farming operations of all sizes either big or small, are using drones to reduce the time and costs associated with crop data collection

Predicting the weather - IBM weather forecast sends out alerts in case of weather disruptions and provides integrated solutions to help maximize crop yields, minimize environmental impact and reduce costs

AI powered robots - Agrobot robots have the ability to operate 24/7 which increases the efficiency, optimizes the cost of precision to weed, hoe, and harvest. The Argobot E-series with advanced AI system not only picks up the crops but it can also identify the ripeness of the crop in the field.

References

- https://www.ibm.com/weather/industries
- https://plantix.net/en/
- https://www.precisionhawk.com/agriculture/drones
- https://www.agrobot.com/e-series
- https://intellias.com/artificial-intelligence-in-agriculture/
- https://www.sciencedirect.com/science/article/pii/S258972172030012X
- https://emerj.com/ai-sector-overviews/ai-agriculture-present-applications-impact/

Time Series Analysis: Opportunities and Challenges

Lalith Achoth

Visiting Faculty, Department of Agricultural Economics, College of Agriculture, UAS, GKVK, Bengaluru

lalithuas@gmail.com

At the best of times, data science can be complicated, opaque, and dense with jargon, especially for those just starting to learn about it. This, at least, was my experience entering the field as a student intern for Wegaw with relatively limited prior knowledge in the details of how exactly you're supposed to do something with a dataset.

However, with some research, the very helpful guidance of someone more experienced, some understanding of high-school statistics, and a few very useful Python libraries, I was ultimately able to make a couple of reasonably good predictions. Described below is the complete process I went through to get these predictions, in the hope that someone in the position I was in about a month ago will find it useful.

The data of interest in this case is the snow depth in certain measuring sites in British Columbia (BC). Predictions of future snow depth through methods like those described below can be a piece of a larger puzzle of tracking snowfall and snowmelt patterns in the region. That being said, a similar process can be used for almost any kind of data.

Aiken Lake, British Columbia, Canada

Finding the Data

Thankfully, this was the easy part. Like most developed countries with any amount of snowfall, Canada has a government agency that keeps reasonably good records of snowfall over time. In this case, the provincial government of BC has publicly available snow data that can be downloaded here.

This is the part of the article where I have to stress the principle of "Garbage In, Garbage Out". It's absolutely vital when doing this kind of work that relatively clean and reliable data be chosen so that results are accurate. It might therefore be worth going through a couple of different datasets to find ones without significant gaps: problems with the data can be fixed, but you can't make data appear from thin air.

For this project, since I was intending to create a multivariate model, I picked three datasets measuring snow depth, temperature, and snow water content on an hourly basis from a single station near Aiken Lake, BC. The Aiken Lake data was in csv format, which I dealt with using the pandas library in Python.

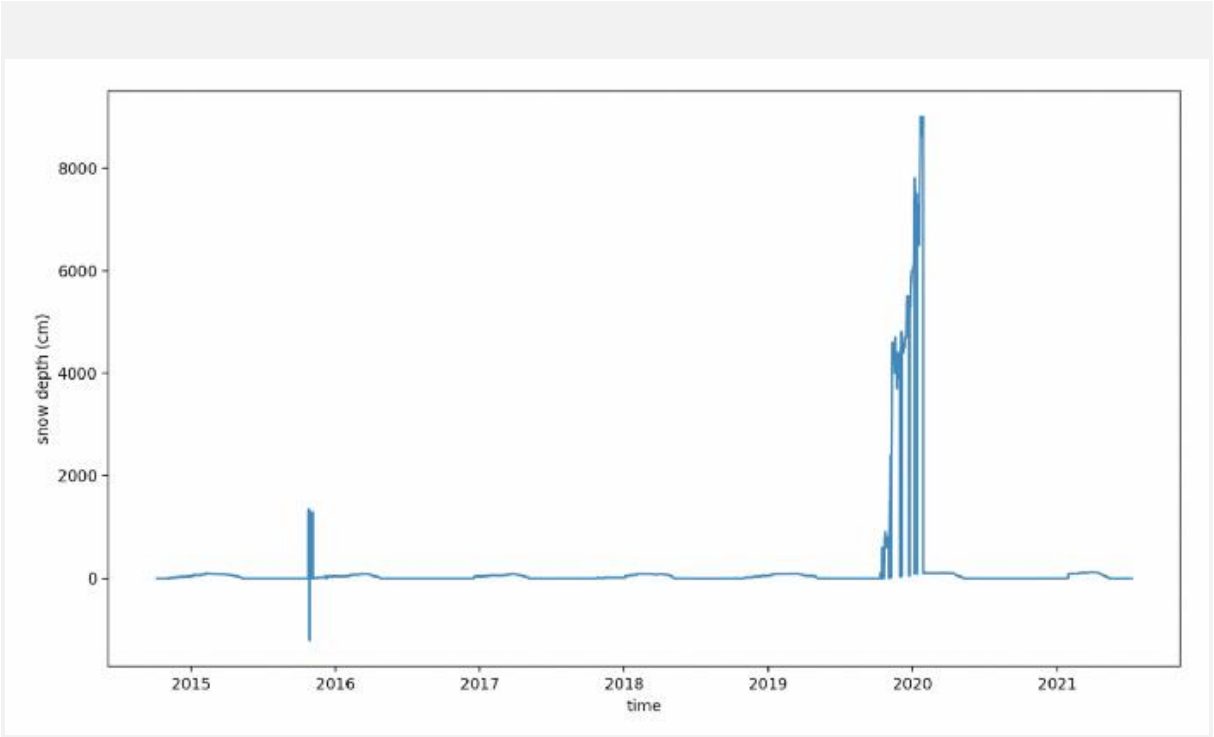
Cleaning the Data

This was by far the most arduous process in the entire project, not necessarily because it was the most technical, but because it was the least interesting.

The first thing I did was get rid of all NaN values from the dataset; even the best datasets will have a couple, and getting rid of them meant working with only numerical data going forward.

To do this, two options presented themselves: I could either delete the rows containing NaN values, or just replace all NaN values with the mean of the previous two values. I chose the latter solution based on the assumption that the weather wasn't likely to vary significantly from one hour to the next. Since there weren't too many continuous NaN values, this wasn't a problem.

Having done this, we can finally look at our data for the very first time:



Uncleaned snow depth over time data

Not very pretty.

The first thing that stands out is the extreme variation, in both the positive and negative directions. By scrolling through the raw data in Excel, it became obvious that this stemmed from an error with the order of magnitude of the data: instead of 8,000 cm of snow, there was really only 80.

The first step to fixing this was to take the absolute value of all the data; getting negative values for snow depth seemed unlikely. The next step was trying to find the right power of 10 to multiply/divide any outliers by so that they fell in line with the rest of the data.

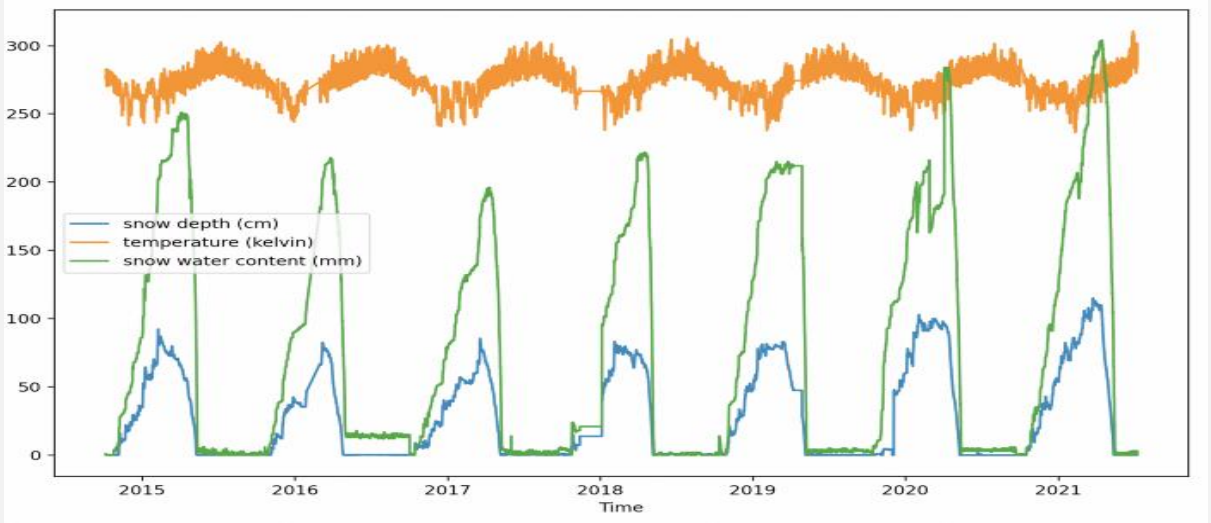
Finding the outliers themselves wasn't the main issue: because I was dealing with an order of magnitude problem, I assumed (probably correctly) that if the depth of the snow increases or decreases by a factor of 10 within an hour, something's gone wrong.

The slightly harder part was figuring out what the order of magnitude is supposed to be. This is a relatively easy task for a person looking through the file in Excel, but a somewhat harder one to program efficiently. This is because the snow data I used is precise to the millimetre, which means that it ranges from 0.1 cm all the way up to about 120 cm: four orders of magnitude.

The solution I ended up implementing is as follows: if a correction is in order, Python checks if the previous value was corrected. If it was, it corrects the current value by the same amount as it corrected the previous one. Otherwise, it assumes that the order of magnitude of the previous value is the same as the true order of magnitude of the current value. This solution breaks if consecutive values are wrong by different orders of magnitude, but this approach was the simplest one I tested which still worked consistently.

The final correction I had to make was to remove any outliers (defined as values more than 3 standard deviations away from the mean) from the data that didn't seem to be caused by any issues in the order of magnitude of the measurement; these were just random values that appeared in the data for no apparent reason. These were dealt with in the same way as NaN values: by replacing the outlier with the mean of the two previous values.

Having done all this to each of the three datasets, the cleaned data looks like this:



Cleaned snow depth, temperature, and snow water content data from Aiken Lake station

(**Note:** the temperature here is in Kelvin in order to keep the values from all the datasets positive. This made cleaning them easier.)

Using the handy `pandas.DataFrame.merge()` function, I combined and downloaded these as one csv file.

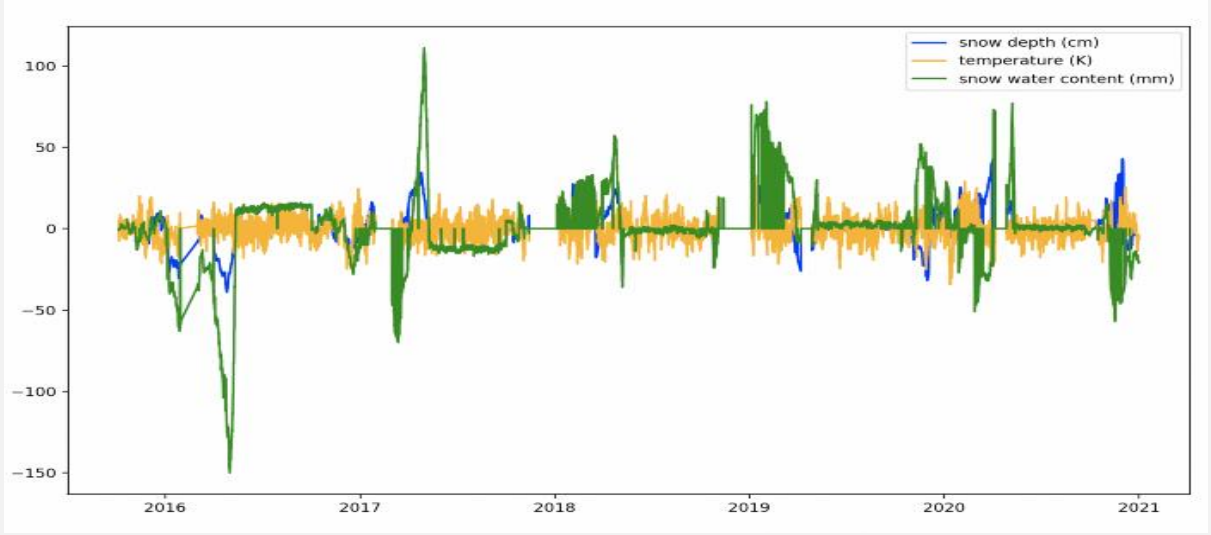
Preparing the Data

Before running the models, however, there is one final matter I had to attend to. Almost all classical statistical models, including the ARMA and VAR models that I ended up using, require a stationary time series. Effectively, this means that the time series can't have any time-dependant patterns or trends (a more detailed explanation of the concept of seasonality can be found here).

Looking at the data, it's pretty obvious that a seasonal pattern is at play here; this is what we would expect from meteorological data like this. While it is possible to run seasonal ARMA models, they're pretty power intensive, and the seasonality must be corrected for the VAR model to work properly anyway. This is most easily done by subtracting each value in the dataset from the value it had a year prior (eg. snow depth on the 1st of January 2021 at 00:00:00 is now equal to itself minus the snow depth on the 1st of January 2020 at 00:00:00). This is a process known as seasonal differencing.

Unfortunately, this means that the first year of our data is unusable, but there is still enough remaining for the models to train on.

The stationary data looks like this:



Snow depth, temperature, and snow water content data with seasonality removed

To make absolutely sure the data was stationary, I ran a unit root test on each dataset. I used the augmented Dickey-Fuller test provided by the statsmodels library (statsmodels.tsa.stattools.adfuller()), although other options are available and equivalent.

For each of the datasets, the Dickey-Fuller test rejected the hypothesis that they have a unit root, implying that the data is stationary. More information on how unit root tests work and how they can be implemented can be found here.

I was now ready to run the models!

Modelling the Data

I used three approaches to try to forecast snow depth: a classical univariate, classical multivariate, and neural network approach. For each of these, different models was used.

Classical Univariate: ARMA

An ARMA (Autoregressive Moving Average) model uses past values of the dataset to forecast future values of the same dataset by combining different models into one. Consequently, I only used snow depth data for this model, since that's what I wanted to predict.

Fully understanding what each component of the ARMA model does is essential in picking the right hyperparameters. Autoregressive (AR) models are models which use previous values to predict future values. For our purposes, the important thing we have to keep track of is how many previous values are used, also called the "order" of the model. As an example, an order 3 autoregressive model would use the three previous values of the dataset to predict the next one. (For more details, here's a website I found useful.)

In a similar way, a moving average (MA) model uses past errors in the model to predict future values. Once again, the "order" of the model refers to how many steps you take back in time in order to predict the next value. (For more details, the same website as above has a pretty good explanation of MA models as well.)

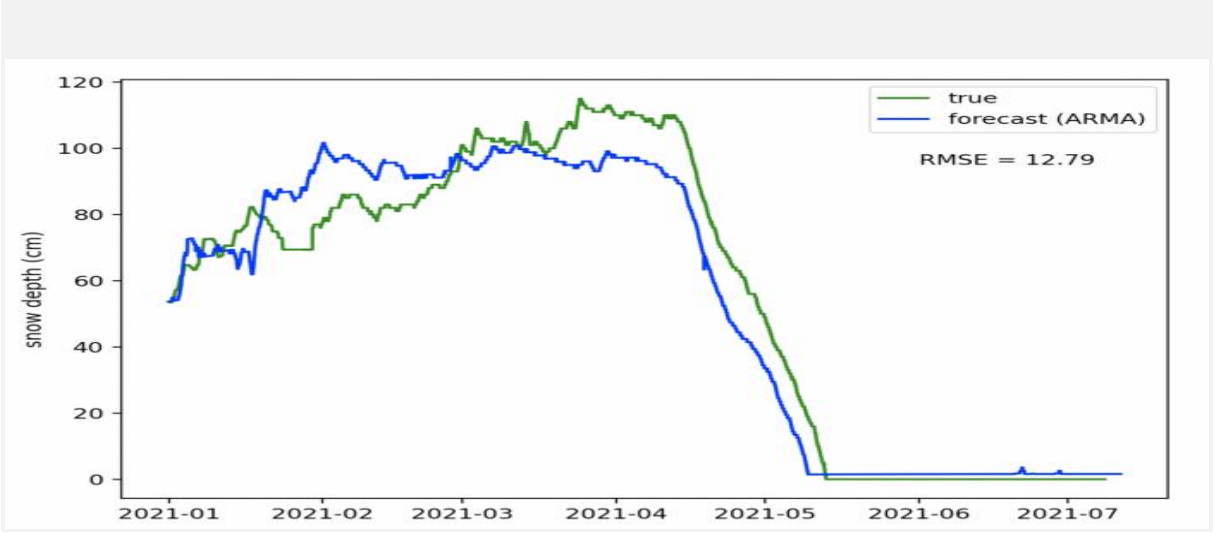
An ARMA model is a combination of an AR and MA model, and requires two hyperparameters to work: the order of the AR model (p) and the order of the MA model (q), written as ARMA(p, q). Although there are more technical and systematic ways to pick these two hyperparameters, the easiest is to just try out a bunch of values and see which ones work best. For this, however, I had to start coding.

As with most statistical models, I was able to find a library that did most of the heavy lifting for me: the `statsmodels.tsa.arima.model.ARIMA()` class, which takes as parameters a time series and an order in the form of a tuplet (p, d, q).

While this (an ARIMA model) is not the same as an ARMA model, we can set d equal to zero, and keep p and q as the orders of the AR and MA models respectively to create an ARMA model. If the data I used wasn't stationary, I would have had to consider d more carefully, but the seasonal differencing was enough in this case to ensure stationarity.

I then split the dataset into train and test data, choosing the arbitrary cut-off point of January 1st, 2021, and applied the model to my train data. Testing a couple different values of p and q, and using RMSE to measure the quality of each resulting model, I arrived on an ARMA(1, 2) model as my solution.

Once the model was fitted, all I had to do was undo the seasonal differencing by adding the value of the previous year to each data point. This gives the following results:



Results for ARMA model predicting snow depth

Pretty good!

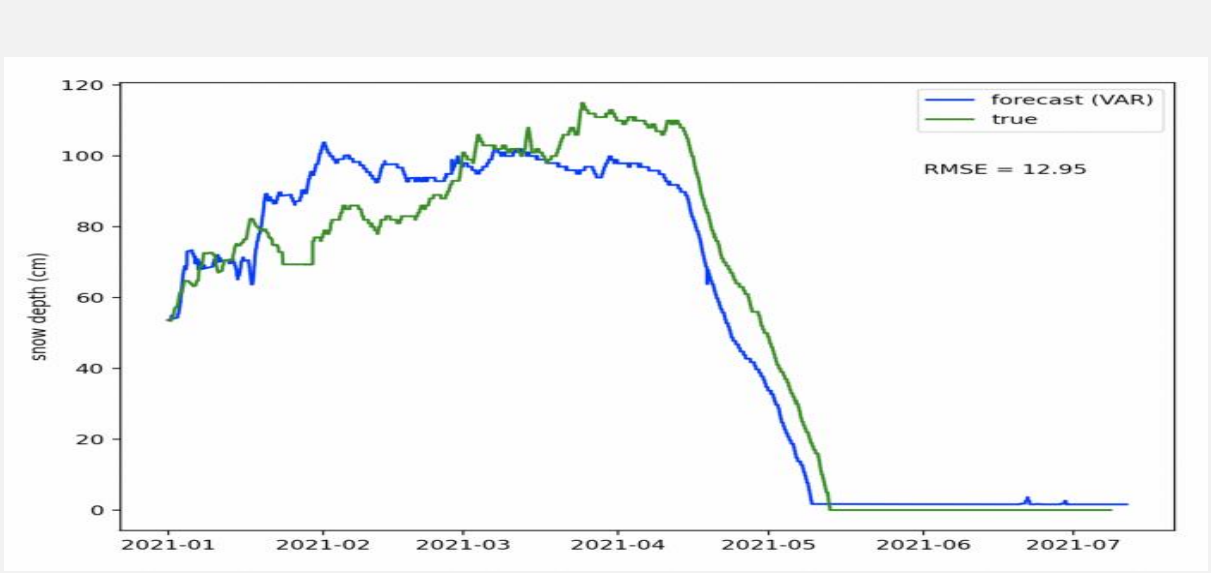
Classical Multivariate: VAR

A vector autoregression model is like an autoregression model but using several time series to predict each other. In this case, I used past values of snow depth, temperature, and snow water content to make predictions about future values of snow depth. Like an AR model, it has an "order", referring to how many previous values of each time series it will consider.

The actual programming is not too different from the ARMA: I once again divided the datasets into train and test, the only difference being all three time series were used instead of only snow depth. These help predict snow depth, assuming that they are all correlated with each other (which, looking at their plots above, they seem to be).

Like with the ARMA model, I used the `statsmodels.tsa.api.VAR()` class to make predictions. Even more conveniently than the ARMA model, the statsmodels VAR class allows you to input a maximum order for the model, and simply checks every order for the model until that value, picking the right one using an information criterion that must be specified (I used the AIC, but others are available). These effectively judge the quality of the model by trying to balance the quality of the predictions and the quantity of parameters (to prevent overfitting).

The final step was to correct the seasonal differencing, and I got the following predictions:



Results for VAR model predicting snow depth

(**Note**: since snow depth was what I was primarily interested it, that's the only thing I plotted but the VAR model makes predictions for each one of the time series I inputted.)

The most striking thing about this model when compared to the ARMA model is its extreme degree of similarity. Because they have (slightly) different RMSEs, it's clear that different models were run, but the extreme similarity does indicate that something may have gone wrong.

My guess is that the other time series added to the model (temperature, snow water content) were not sufficiently correlated with snow depth, so the model treated them as insignificant and basically just ran an autoregression model on the snow depth data.

In any case, however, a RMSE of about 13 cm is also pretty good in this case too.

ARMA model predicting snow depth for the next month

It's hard to know whether scaling up the problem for the LSTM (six or seven months, rather than just one) will result in the same quality of prediction, but doing so requires more time and computing power than I currently have on my hands. However, I think that, at least on this scale, this implementation of an LSTM can definitely be considered successful.

Conclusion

Before the end of the article, I just wanted to mention some generally useful tips for anyone who's interested in starting.

- If you ever find yourself thinking "why hasn't someone already written a method for this?" someone probably already has.

When I first started, I spent about three hours creating and testing a function that takes a string and turns into a `datetime.datetime()` object, before discovering the pandas `to_datetime()` method. Don't make the same mistake I did.

- Understand the models you're using

There are many, many websites and tutorials explaining the models described above with examples that you can very easily copy-paste into your IDE. This is fine, as long as you don't then spend a few hours guessing different hyperparameters without even knowing what order of magnitude they're supposed to be.

- Don't be afraid to use other people's work

This is probably the most important piece of advice I'd give to someone just starting: someone more experienced than you has already done a lot of the hard part for you. There are dozens of different libraries that do most of the technical computational part of the process behind the

scenes. As long as you learn to use them properly, and have a vague understanding of what you need to do to make them work, you should implement them in whatever way you find useful.

The process of finding out more about data science and modelling through practical examples like these has been incredibly interesting and intellectually stimulating, and I would recommend to anyone interested in this field to start by getting some real data and working with it in a way similar to what I did.

Finally, I want to thank everyone at Wegaw for this opportunity, but especially Daria Ludtke for organising the internship, and Thomas James for his patience in taking me through some of the more complicated parts of the process.

Practical Manual

Time Series Techniques for Forecasting in Agriculture

December 1- 10, 2021

Division of Agriculture Economics, ICAR-IARI, New Delhi

Course Convenor

Alka Singh
Professor and Head
Division of Agricultural Economics
ICAR-Indian Agricultural Research Institute
Pusa Campus, Delhi – 110012
Email: asingh.eco@gmail.com

Co-Convenors

Girish K Jha
Principal Scientist
Division of Agricultural Economics
ICAR-Indian Agricultural Research Institute
New Delhi 110 012
E-mail: girish.stat@gmail.com

Nithyashree M L
Scientist
Division of Agricultural Economics
ICAR-Indian Agricultural Research Institute
New Delhi 110 012
E-mail: nithya.econ@gmail.com

Asha Devi S S
Scientist
Division of Agricultural Economics
ICAR-Indian Agricultural Research Institute
New Delhi 110 012
E-mail:ash.nibha@gmail.com

Published By
NAHEP- Centre for Advanced Agriculture Science and Technology
ICAR- Indian Agriculture Research Institute, New Delhi
Website: www.nahep-caast.iari.res.in