

Effective use of Big Data Analytics in Crop planning to increase Agriculture Production in India

Team Members:

Anju Prasannan (AM.EN.D*CSE20420)

Joice T (AM.EN.D*CSE20409)

Overview



- Background Study
- Problem Statement
- Proposed Solution
- Technology Stack
- Methodology
 - Solution Paradigm
 - Workflow
 - Dataset
 - Data Preprocessing
- Result
 - Demand Calculation using Spark
 - Demand Calculation using Map Reduce
 - K-means Clustering
 - Data Visualizations
 - Spark Vs Map Reduce
- Conclusion

Background Study



- Big Data Analytics is one of the affirmative platforms to implement large scale Data Analytics
- Big Data Analytics is the process to find unidentified correlations, hidden patterns, market trends, customer preferences, and other essential data from extensive distributed datasets.
- Big Data Analytics provides various advantages—it can be used for better decision making, preventing fraudulent activities, among other things.
- Big Data with the help of Machine Learning algorithms can categorize the input data, recognize patterns and translate the data into insights

Problem Statement



Quality of inputs is vital for improving crop quality and yield, so availability and accessibility of right inputs to farmers is a key for their empowerment and this emphasizes the usage of Big Data which enables the farmers to improve the quality of their products.

Proposed Solution



- Govt. of India created an open data ecosystem for the motive of sharing crop dataset as per National Data Sharing and Accessibility Policy (NDSAP) initiated Open Government Data (OGD) Platform.
- Traditional data analysis methodologies may not be sufficient to predict the crop patterns in the dataset. If the entire process is done by a single node, it usually gets exhausted and consumes time to analyze crop price and yield information.
- This approach proposes popular Map-Reduce concept and R programming concept utilized in clustered file system extensively with Hadoop Distributed File System (HDFS).
- The purpose behind the **Map-Reduce** paradigm is highly scalable which executes massively parallel and distributed computations over a huge number of computing nodes. **R** is a free open source software environment, user interface design for statistical computing and data visualization. **Spark** is a lightning faster tool used for data analytics, and it works better in small dataset

Technology Stack



- Java: jdk1.8.0_281
- Maven: 3.6.3
- Eclipse: 2020-12 (4.18.0)
- Hadoop: 3.2.1
- R: 4.0.4
- R-Studio: 1.4.1106
- Spark through Databricks
- Git



Methodology

Solution Paradigm

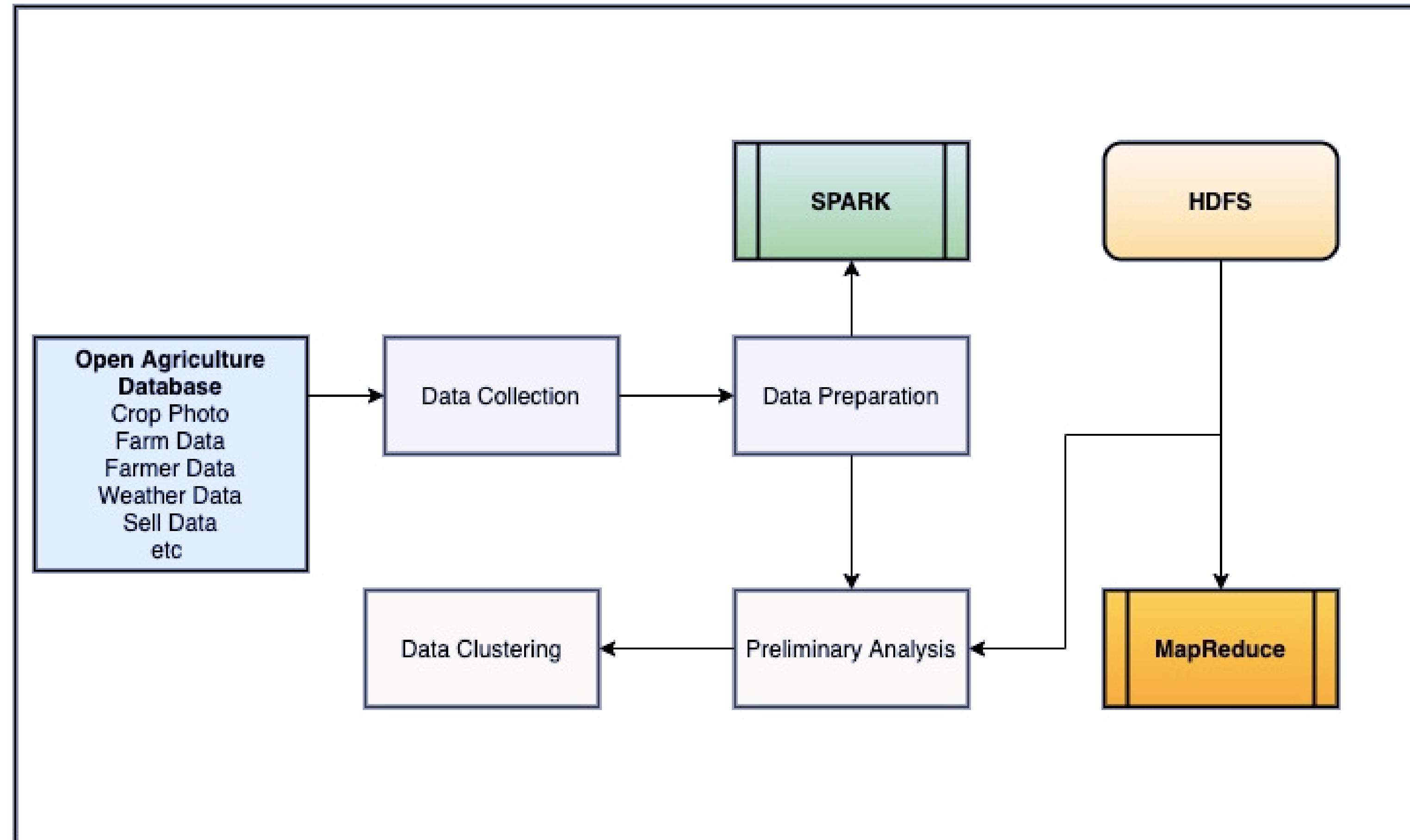


The following Data Analysis methodologies are used:

- Demand Calculation using ***Map Reduce*** Paradigm
- Demand Calculation using ***Spark*** through Databricks
- K-means Clustering using ***R***
- Data Visualizations using ***R***

Methodology

Workflow



Methodology

Dataset



- **Reliable crop dataset collected from an open data ecosystem of Open Government Data (OGD) Platform India published by National Data Sharing and Accessibility Policy.**
- **Dataset 1:**
area_production_and_productivity_of_principal_crops_2019.csv
- **Major Fields are :**
[Crop,Area (Hectare),Production (Tonnes), Productivity (in Kg / Hectare), Max.Production]
- **Dataset 2: Apy.csv (is dataset 246091 obs. of 7 variables)**
- **Major Fields are :**
[State_Name, District_Name, Crop_Year, Season, Crop, Area]

Methodology

Data Preprocessing



- Upon analysis of the data, we focused on removing the row wise summary information of state wise crop production.
- We have added a new field 'demand' to the dataset by deriving it from available fields like production.
- For these preprocessing tasks we used Microsoft Excel as a tool.



Result

Demand Calculation using Spark

```
1 from pyspark.sql.types import *
2 from pyspark.sql.functions import *
3 df=spark.read.csv("/FileStore/tables/areal-6.csv",header="true")
4 display(df)
```

▶ (1) Spark Jobs
▶ df: pyspark.sql.dataframe.DataFrame = [S.No: string, Crop: string ... 7 more fields]

	S.No	Crop	Area (Hectare)	Production (Tonnes)	Productivity (in Kg / Hectare)	Max.Production	Demand
1	1	Paddy in cereals	1828919	6638450	3630	6638975970	6638972340
2	2	Cholam Jowar in cereals	385646	430661	1117	430766582	430765465
3	3	Cumbu Bajra in cereals	63029	143548	2277	143517033	143514756
4	4	Ragi in cereals	86513	321332	3714	321309282	321305568
5	5	Maize in cereals	324518	2591632	7986	2591600748	2591592762
6	6	Small Millets in cereals	25278	31226	1235	31218330	31217095
7	7	Bengal gram in pulses	5205	4820	926	4819830	4818904
8	8	Red gram in pulses	49225	53788	1093	53802925	53801832
9	9	Green gram in pulses	180587	78259	433	78194171	78193738
10	10	Black gram in pulses	426332	301662	707	301416724	301416017
...

Showing all 21 rows.

Command took 22.95 seconds -- by joicet@am.amrita.edu at 6/5/2021, 1:35:05 pm on My Cluster

```
1 df2=df.select("Crop","Demand","Group").groupBy("Group").agg(sum("Demand"))
2 df2.show()
```

▶ (2) Spark Jobs
▶ df2: pyspark.sql.dataframe.DataFrame = [Group: string, sum(Demand): double]

Group	sum(Demand)
oil seeds	5.208673368E9
Cereals	1.0157367986E10
Other crops	7.58807045E8
pulses	5.56132525E8

Command took 3.76 seconds -- by joicet@am.amrita.edu at 6/5/2021, 1:46:59 pm on My Cluster

Result

Demand Calculation using MapReduce

eclipse-workspace - CropPlanningBDA/src/main/java/com/anjus/mapreduce/CropDemandDriver.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- CropPlanningBDA
 - src/main/java
 - com.anjus.mapreduce
 - CropDemandDriver.java
 - CropDemandMapper.java
 - CropDemandReducer.java
 - com.anjus.utils
 - Constants.java
 - src/main/resources
 - input
 - apy.csv
 - output
 - _SUCCESS
 - part-r-00000
 - log4j.properties
 - JRE System Library [JavaSE-15]
 - Maven Dependencies
 - bin
 - com
 - input
 - output
 - RScrips
 - Clustering.R
 - log4j.properties
 - logs
 - src
 - target
 - dependency-reduced-pom.xml
 - pom.xml
 - HelloWorld
 - MapReduceExample

log4j.properties

CropDemandDriver.java

```
1 package com.anjus.mapreduce;
```

Problems Javadoc Declaration Console Progress Terminal

<terminated> CropDemandDriver (1) [Java Application] C:\Program Files\eclipse\eclipse-java-2020-12-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.

2021-05-06 22:43:44-8081 INFO org.apache.hadoop.mapreduce.Job: map 100% reduce 100%

2021-05-06 22:43:44-8082 INFO org.apache.hadoop.mapreduce.Job: Job job_local170606648_0001 completed successfully

2021-05-06 22:43:44-8094 INFO org.apache.hadoop.mapreduce.Job: Counters: 30

File System Counters

- FILE: Number of bytes read=30638814
- FILE: Number of bytes written=720924
- FILE: Number of read operations=0
- FILE: Number of large read operations=0
- FILE: Number of write operations=0

Map-Reduce Framework

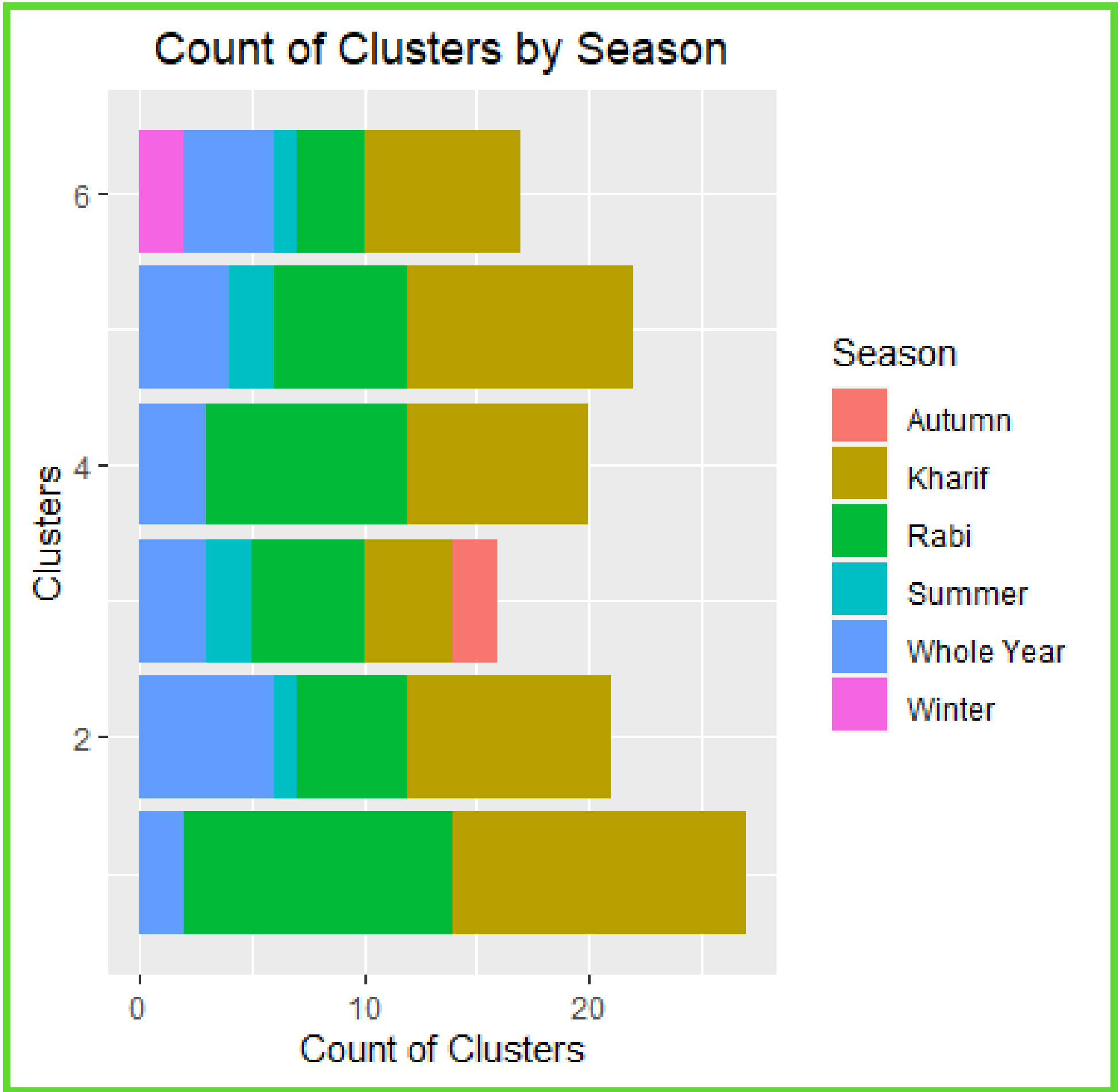
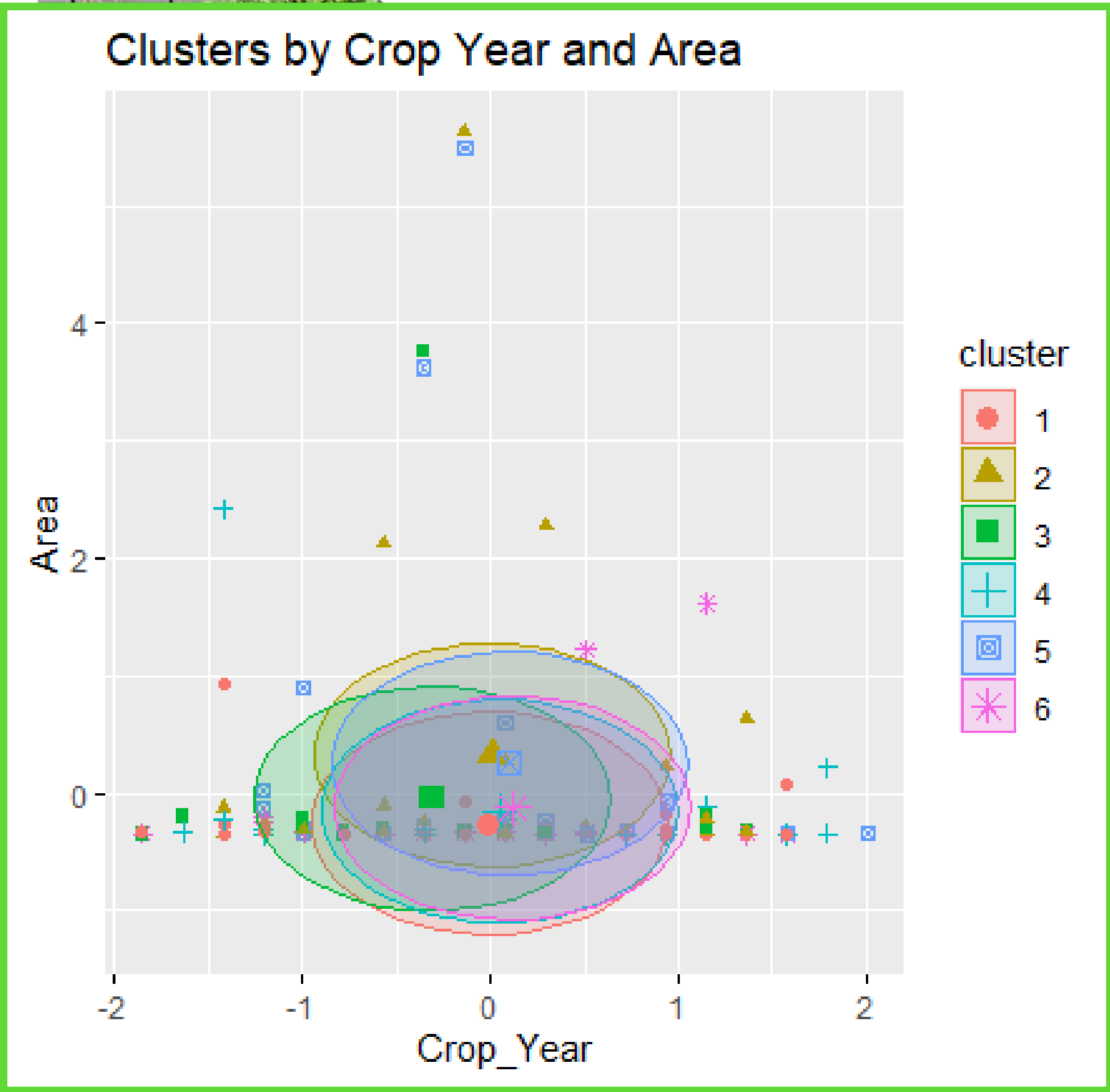
- Map input records=246091
- Map output records=242361
- Map output bytes=4093877
- Map output materialized bytes=2505
- Input split bytes=152
- Combine input records=242361
- Combine output records=123
- Reduce input groups=123
- Reduce shuffle bytes=2505
- Reduce input records=123
- Reduce output records=123
- Spilled Records=246
- Shuffled Maps =1
- Failed Shuffles=0
- Merged Map outputs=1
- GC time elapsed (ms)=73
- Total committed heap usage (bytes)=524288000

Shuffle Errors

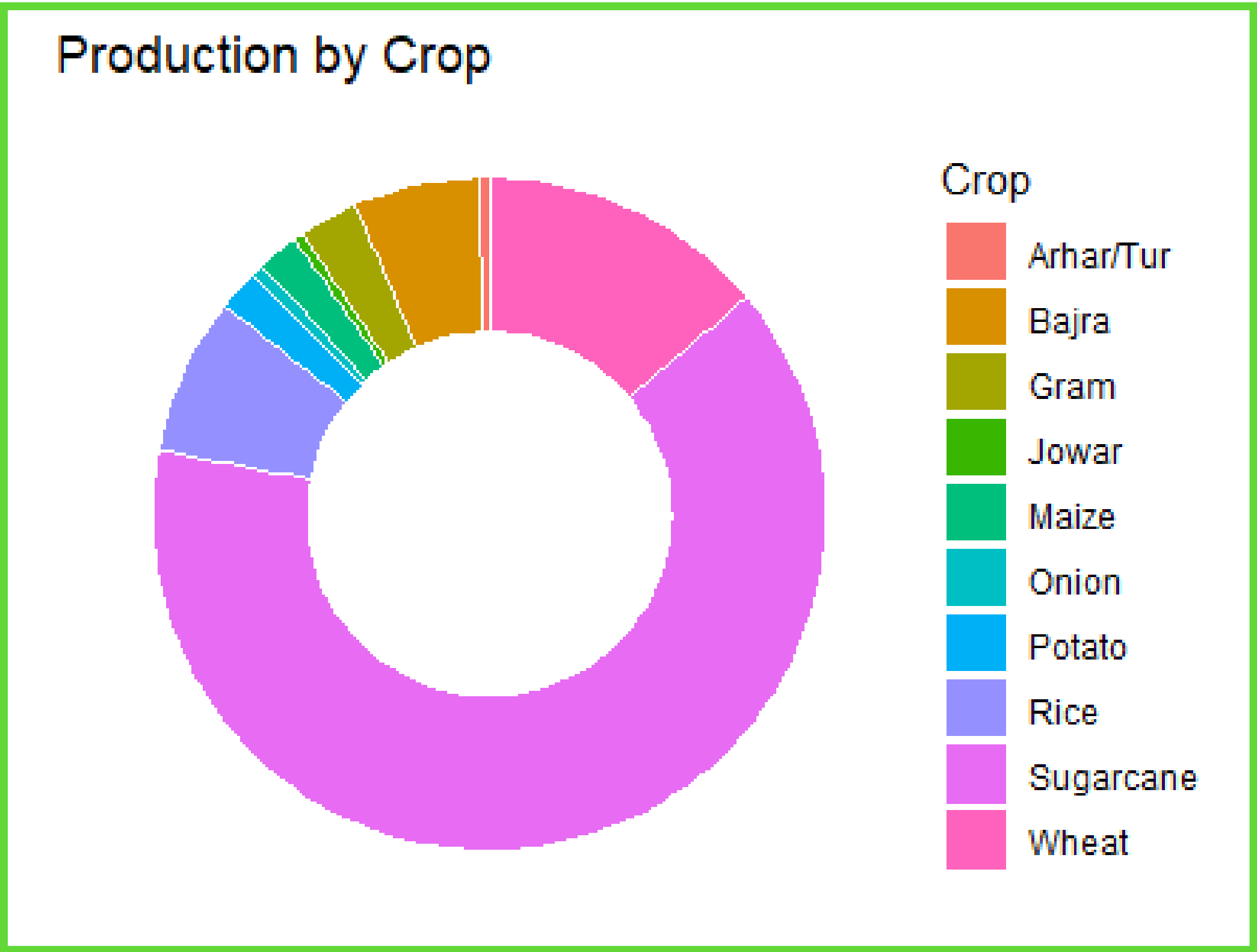
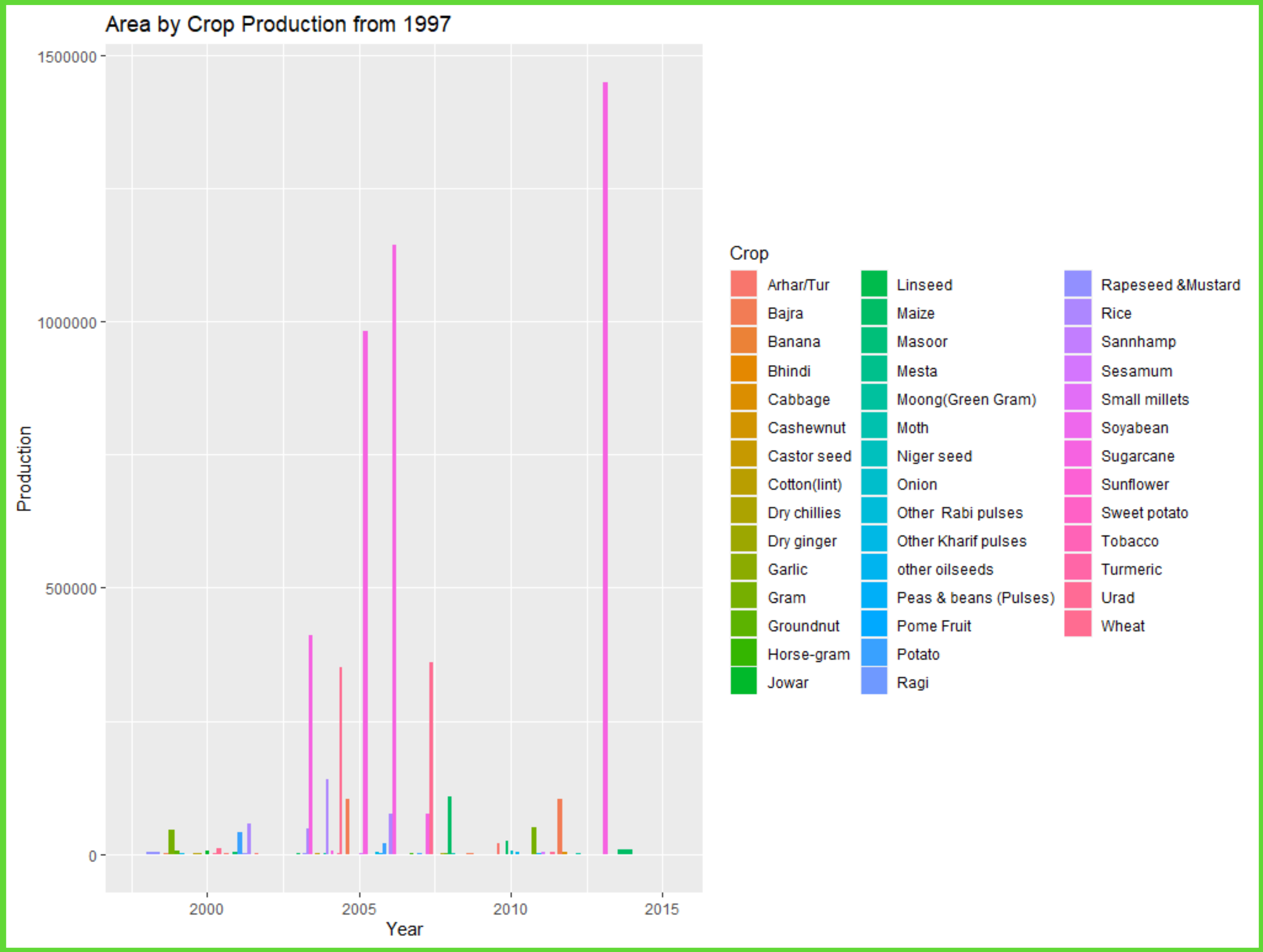
- BAD_ID=0
- CONNECTION=0
- IO_ERROR=0

Result

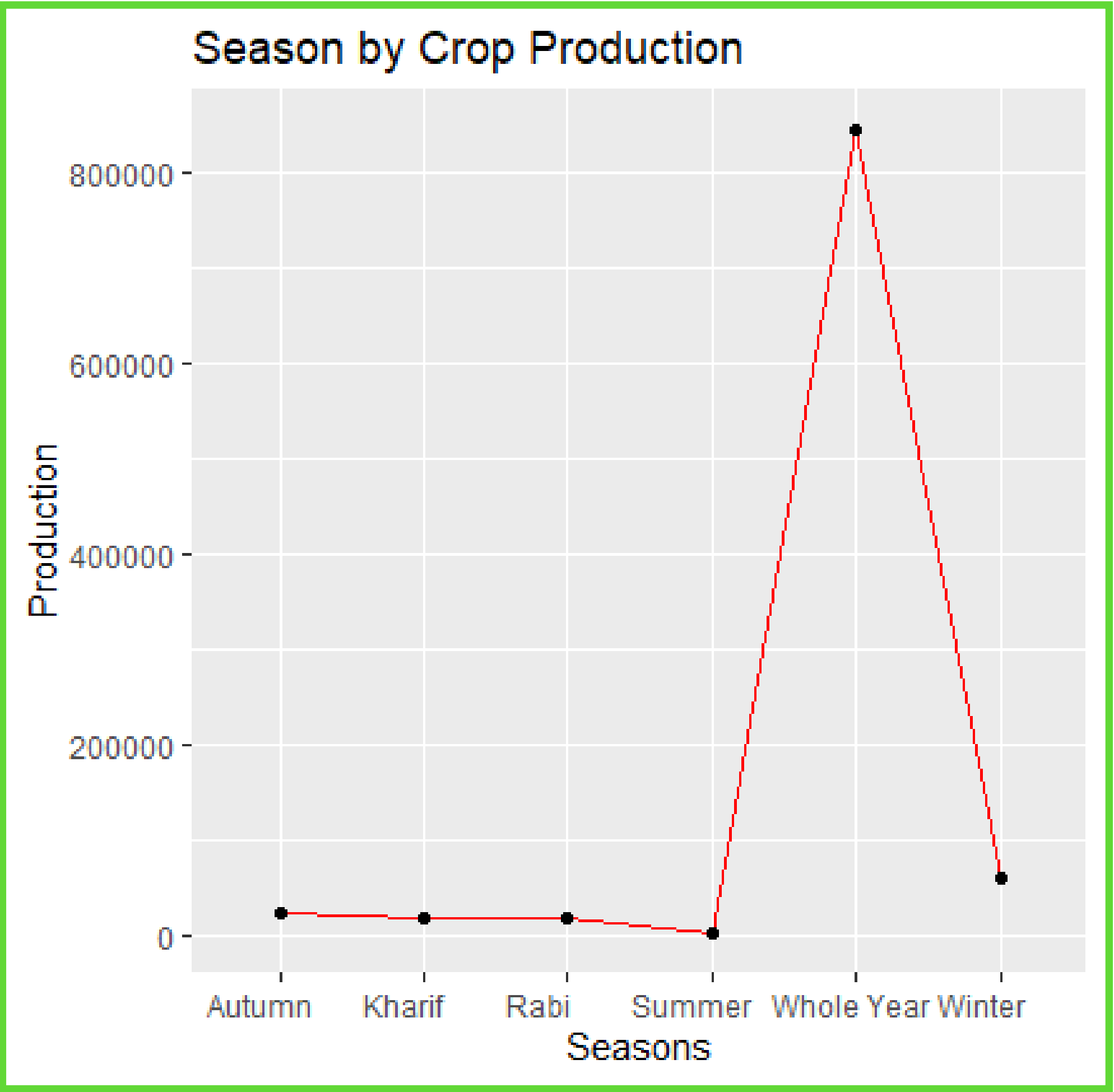
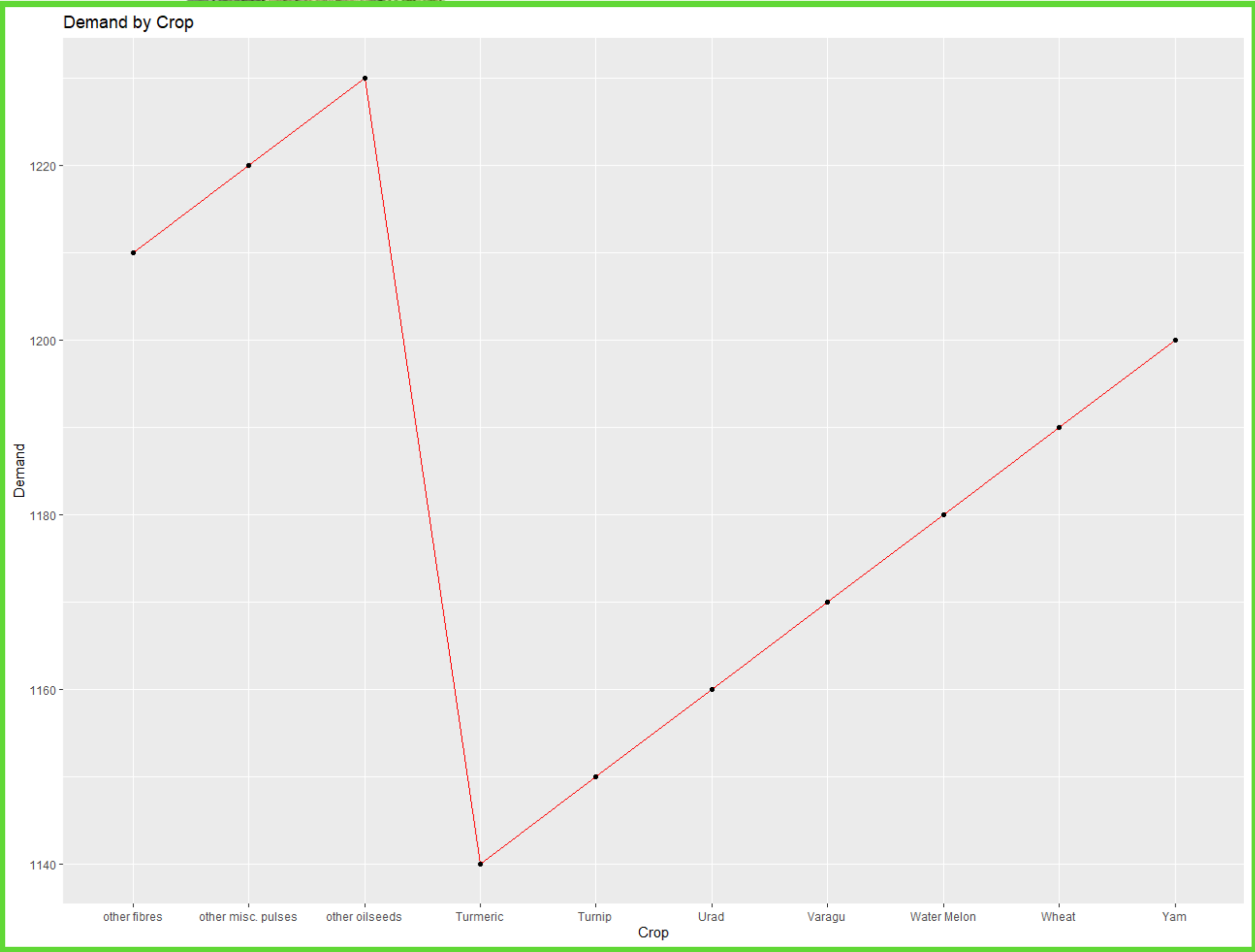
K Means Clustering



Data Visualizations



Data Visualizations



Spark Vs Map Reduce



<i>Factors</i>	<i>Spark</i>	<i>Map Reduce</i>
Speed	100x faster than Map Reduce	Faster than traditional processing
Written In	Scala	Java
Data Processing	Batch/real-time/iterative/interactive/graph	Batch processing
Ease of use	Compact and easy	Complex and lengthy
Caching	In-memory caching of data	Caching of data is not supported
Cost	More cost since it requires little high end commodity hardware with more RAM	Works with lesser RAM, commodity hardware is sufficient and hence less cost
Scheduling	Map Reduce requires YARN or Mesos for execution.	Spark can run in standalone mode using default scheduler. It can also run on YARN or Mesos.
Security	Less support for authentication using HDFS ACLs, Kerberos and shared secrets.	Uses all Hadoop security benefits and integrates with Hadoop Security projects like Knox, Gateway and Sentry.
Latency	Low	High
Interactive mode	Supports spark shell for Scala/ Python/ R	No
Machine Learning/ Graph Processing	Dedicated modules like Spark MLlib and GraphX for ML and Graph processing.	No inbuilt support. But Mahout can be used for ML.

Conclusion



Big Data Analytics is one of the best systems for crop planning to increase agriculture productivity.

Effective use of Big Data Analytics on crop planning may be very significant to boom agricultural manufacturing.

The advancements in Information and Communication Technology is a boon to common man.



Thank You...!