Lecture 05: Data Mining and Statistics in SQL and R

KD44103 Big Data Analytics
Faculty of Computing & Informatics,
Universiti Malaysia Sabah

Learning outcomes

- Explain Implementation Process of Data Mining
- Managing data with R and Oracle Data Mining
- Provides hands-on, Oracle R Enterprise Works.
- Explain the basic concepts, typical R Approach and Challenges

Data mining

- is the process of sorting through data to identify patterns and relationships using statistical algorithms that may help us understand which factors affect an outcome of something, or they may be used to predict future outcomes.
- Data mining might be used to attempt to answer questions such as ‘How many items are likely to be sold next month?’

Data mining

- Data mining can do a range of tasks for which many different algorithms have been developed to fulfil different specific purposes:
- Predictive tasks
- Descriptive tasks

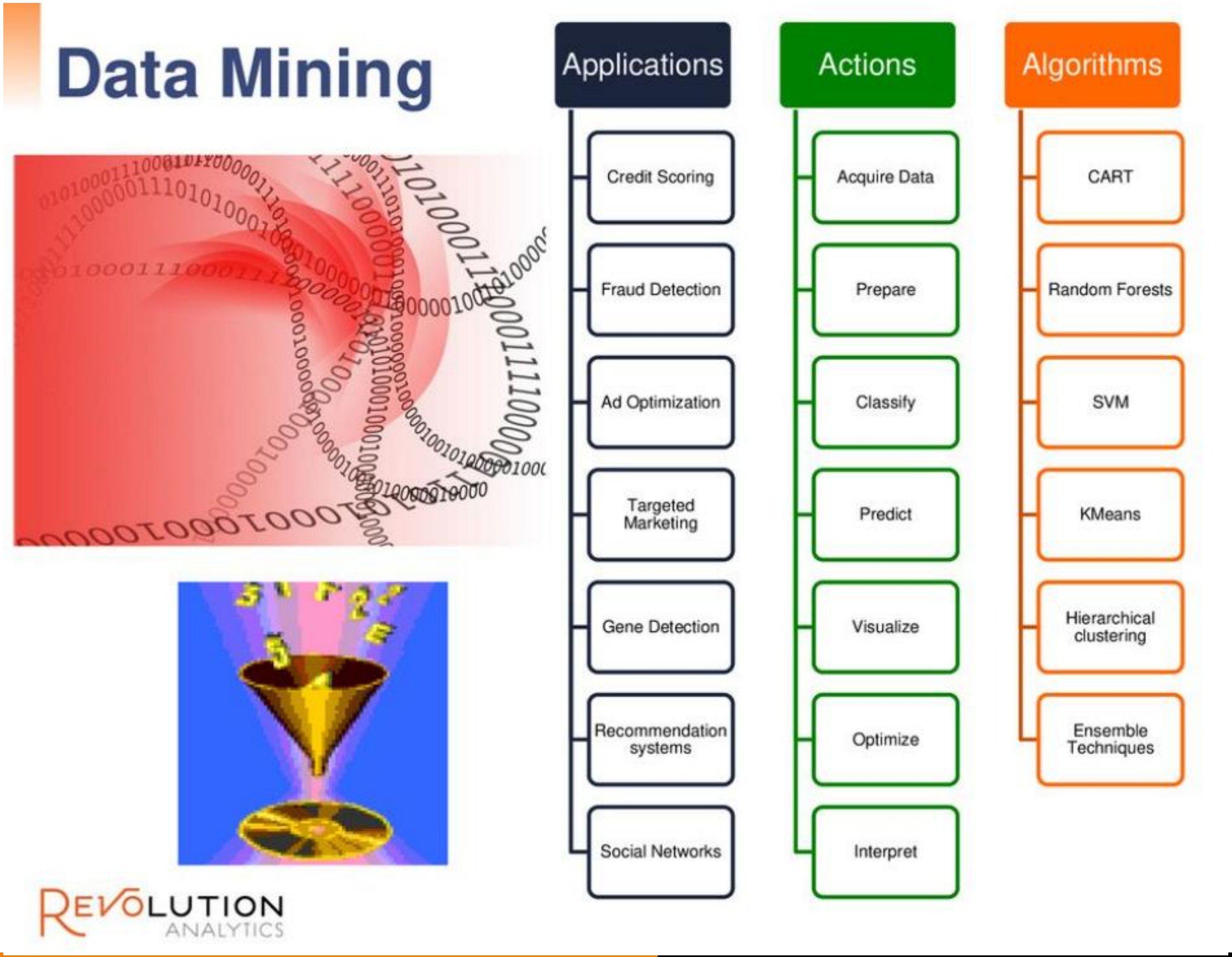
Predictive tasks

- involve constructing one or more models of the data, to predict the behaviour of new instances of the data.
- Example: estimate the sales in the next quarter based on historical sales data. Widely used techniques:
- Classification
- Regression analysis

Descriptive tasks

- concerned with describing data in a concise and summary manner and attempting to identify meaningful properties of that data.
- Example: identify different market segments by grouping customers based on their characteristics and spending patterns. Widely used techniques:
- Clustering
- Association rule discovery
- Visualisation

Data Mining with R



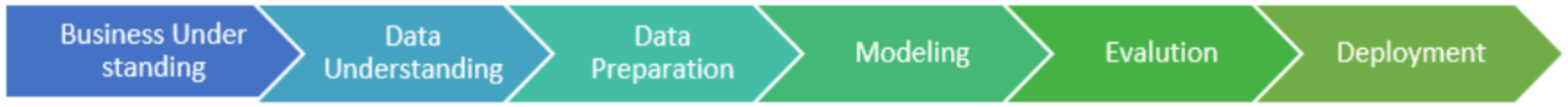
Data Mining

Applications	Actions	Algorithms
Credit Scoring	Acquire Data	CART
Fraud Detection	Prepare	Random Forests
Ad Optimization	Classify	SVM
Targeted Marketing	Predict	KMeans
Gene Detection	Visualize	Hierarchical clustering
Recommendation systems	Optimize	Ensemble Techniques
Social Networks	Interpret	

REVOLUTION ANALYTICS

Source: Introduction to R for Data Mining . STRATA 2012. Joseph B. Rickert , Revolution Analytics. February 28, 2012.

Implementation Process of Data Mining



Types of Data

Data mining can be performed on following types of data

- Relational databases
- Data warhouses
- Advanced DB and information repositories
- Object-oriented and object-relational databases
- Transactional and Spatial databases
- Heterogeneous and legacy databases
- Multimedia and streaming database
- Text databases
- Text mining and Web mining

Data preparation

- n this phase, data is made production ready.
- The data preparation process consumes about 90% of the time of the project.
- The data from different sources should be selected, cleaned, transformed, formatted, anonymized, and constructed (if required).
- Data cleaning is a process to “clean” the data by smoothing noisy data and filling in missing values.
- For example, for a customer demographics profile, age data is missing. The data is incomplete and should be filled. In some cases, there could be data outliers. For instance, age has a value 300. Data could be inconsistent. For instance, name of the customer is different in different tables.
- Data transformation operations change the data to make it useful in data mining. Following transformation can be applied

Data transformation:

- Data transformation operations would contribute toward the success of the mining process.
- **Smoothing:** It helps to remove noise from the data.
- **Aggregation:** Summary or aggregation operations are applied to the data. I.e., the weekly sales data is aggregated to calculate the monthly and yearly total.
- **Generalization:** In this step, Low-level data is replaced by higher-level concepts with the help of concept hierarchies. For example, the city is replaced by the county.
- **Normalization:** Normalization performed when the attribute data are scaled up o scaled down. Example: Data should fall in the range -2.0 to 2.0 post-normalization.
- **Attribute construction**: these attributes are constructed and included the given set of attributes helpful for data mining.
- The result of this process is a final data set that can be used in modeling.

Modelling

- In this phase, mathematical models are used to determine data patterns.
- Based on the business objectives, suitable modeling techniques should be selected for the prepared dataset.
- Create a scenario to test check the quality and validity of the model.
- Run the model on the prepared dataset.
- Results should be assessed by all stakeholders to make sure that model can meet data mining objectives.

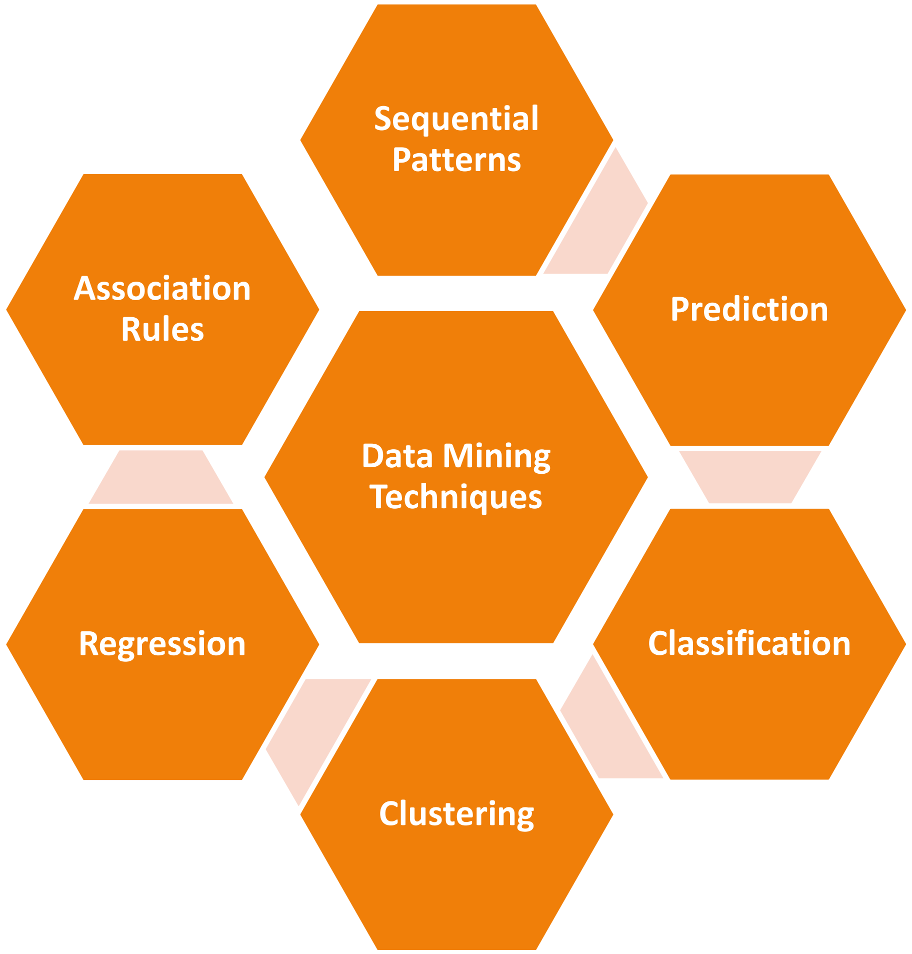
Evaluation

- In this phase, patterns identified are evaluated against the business objectives.
- Results generated by the data mining model should be evaluated against the business objectives.
- Gaining business understanding is an iterative process. In fact, while understanding, new business requirements may be raised because of data mining.
- A go or no-go decision is taken to move the model in the deployment phase.

Deployment

- In the deployment phase, you ship your data mining discoveries to everyday business operations.
- The knowledge or information discovered during data mining process should be made easy to understand for non-technical stakeholders.
- A detailed deployment plan, for shipping, maintenance, and monitoring of data mining discoveries is created.
- A final project report is created with lessons learned and key experiences during the project. This helps to improve the organization's business policy.

Data Mining Techniques



Challenges of Implementation of Data mining

- Skilled Experts are needed to formulate the data mining queries.
- Overfitting: Due to small size training database, a model may not fit future states.
- Data mining needs large databases which sometimes are difficult to manage
- Business practices may need to be modified to determine to use the information uncovered.
- If the data set is not diverse, data mining results may not be accurate.
- Integration information needed from heterogeneous databases and global information systems could be complex

Data Mining Tools

Following are 2 popular Data Mining Tools widely used in Industry

- **R-language:**
- **Oracle Data Mining:**

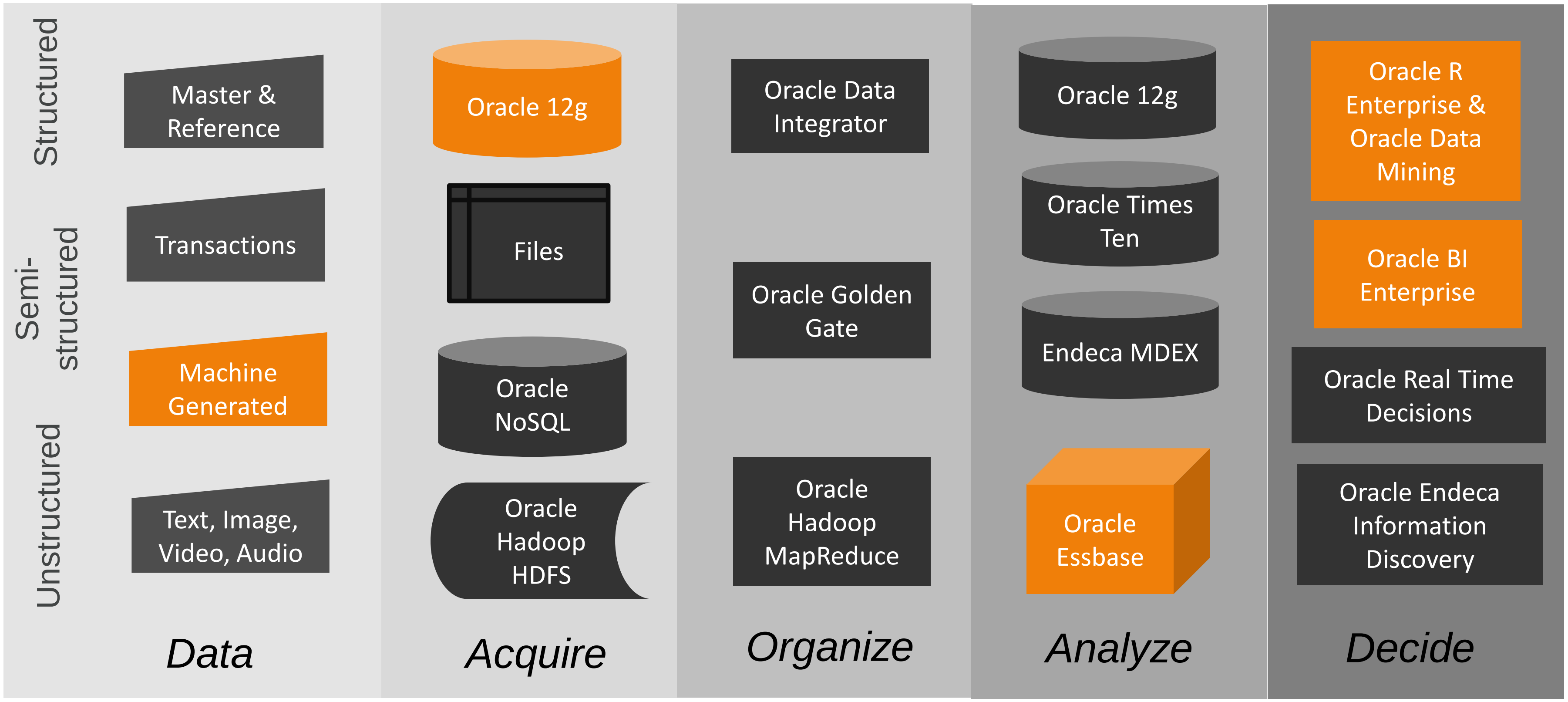
R-language

- Is an open source tool for statistical computing and graphics. R has a wide variety of statistical, classical statistical tests, time-series analysis, classification and graphical techniques. It offers effective data handing and storage facility.

Oracle Data Mining

- popularly knowns as ODM is a module of the Oracle Advanced Analytics Database. This Data mining tool allows data analysts to generate detailed insights and makes predictions. It helps predict customer behavior, develops customer profiles, identifies cross-selling opportunities.

Oracle Technology mapped to Analytics Landscape

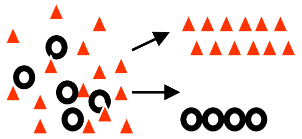
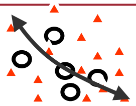
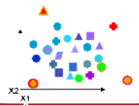
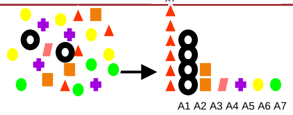
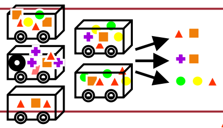
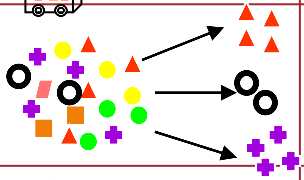
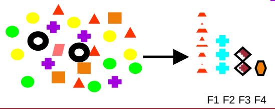


Oracle Advanced Analytics Option—Agenda

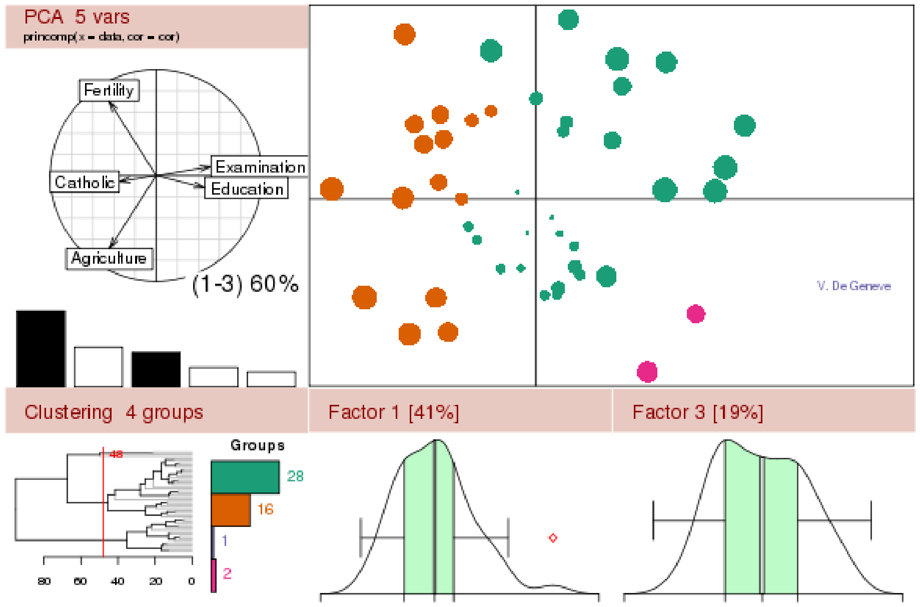
- Oracle Data Mining
  - SQL & PL/SQL focused in-database data mining and predictive analytics
- Oracle R Enterprise
  - Integrates Open Source R with the Oracle Database

- Extending the Database into a Comprehensive Advanced Analytics Platform

Oracle Data Mining Algorithms

Problem	Algorithm	Applicability
Classification 	Logistic Regression (GLM) Decision Trees Naïve Bayes Support Vector Machine	Classical statistical technique Popular / Rules / transparency Embedded app Wide / narrow data / text
Regression 	Multiple Regression (GLM) Support Vector Machine	Classical statistical technique Wide / narrow data / text
Anomaly Detection 	One Class SVM	Lack examples of target field
Attribute Importance 	Minimum Description Length (MDL)	Attribute reduction Identify useful data Reduce data noise
Association Rules 	Apriori	Market basket analysis Link analysis
Clustering 	Hierarchical K-Means Hierarchical O-Cluster	Product grouping Text mining Gene and protein analysis
Feature Extraction 	Nonnegative Matrix Factorization	Text analysis Feature reduction

R Statistical Programming Language



Open source language and environment

Used for statistical computing and graphics

Strength in easily producing publication-quality plots

Highly extensible with open source community R packages

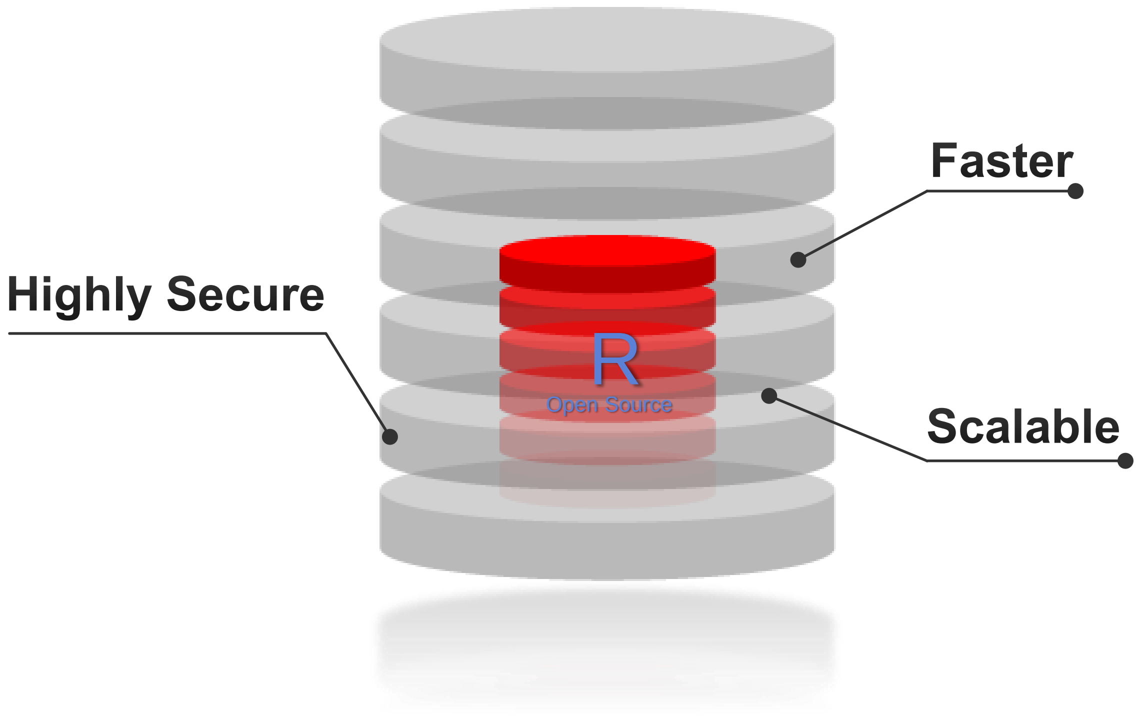
Typical R Approach

Statistical and advanced analyses are run and stored on the user's laptop

What Are's Challenges?

1. R is memory constrained
   - R processing is single threaded - does not exploit available compute infrastructure
   - R lacks industrial strength for enterprise use cases
2. R has lacked mindshare in Enterprise market
   - R is still met with caution by the long established SAS and IBM/SPSS statistical community
     - However, major university (e.g. Yale) Statistics courses now taught in R
     - The FDA has recently shown indications for approval of new drugs for which the submission's data analysis was performed using R

Oracle R Enterprise Approach



Data and statistical analysis are stored and run in-database

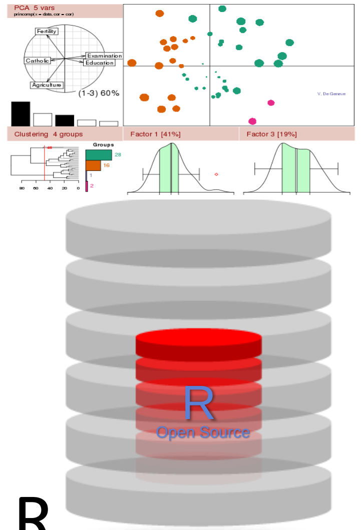
Same R user experience & same R clients

Embed in operational systems

Complements Oracle Data Mining

What is Enterprise?

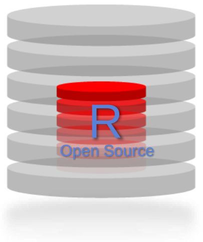
ORACLE R



- Oracle R Enterprise brings R's statistical functionality closer to the Oracle Database

1. Eliminate R's memory constraint by enabling R to work directly/transparently on database objects
   – Allows R to run on very large data sets, tables, views
2. Architected for Enterprise production infrastructure
   – Automatically exploits database parallelism without requiring parallel R programming
   – Build and immediately deploy R scripts

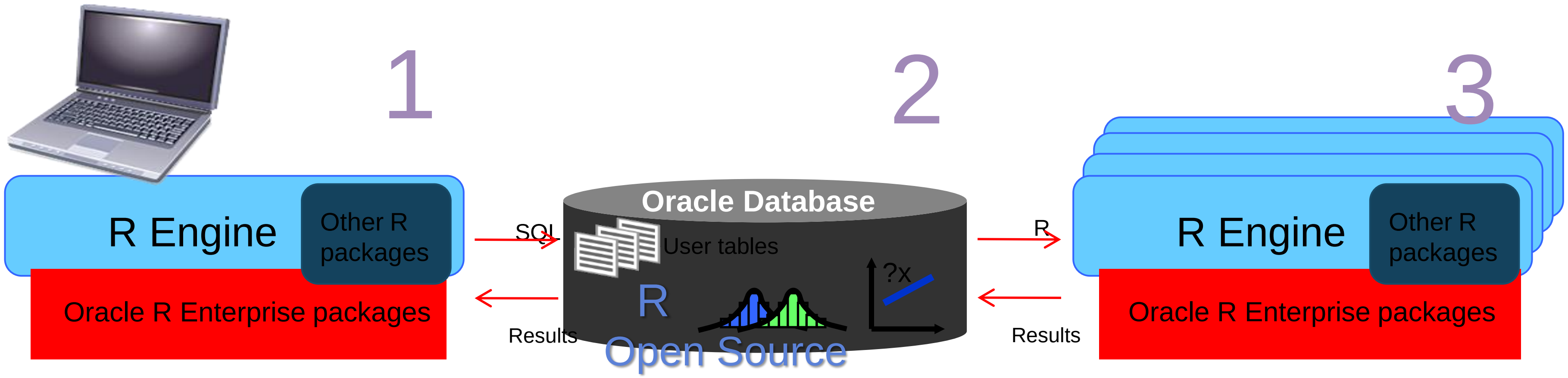
How Oracle R Enterprise Works



- *ORE Computation Engines*
- Oracle R Enterprise eliminates data movement and duplication, maintains security and minimizes latency time from raw data to new information.
  - The database is always involved in serving up data to the R code.
  - Oracle R Enterprise runs in the Oracle Database
- Three ORE Computation Engines
  - Oracle R Enterprise provides three different interfaces between the open-source R engine and the Oracle database:

1. Oracle R Enterprise (ORE) Transparency Layer
2. Oracle Statistics Engine
3. Embedded R

Oracle R Enterprise Compute Engines



User R Engine on desktop

- R-SQL Transparency Framework intercepts R functions for scalable in-database execution
- Submit entire R scripts for execution by Oracle Database

Database Compute Engine

- Access tables, views, and external tables, as well as data through DB LINKS
- Leverage database SQL parallelism
- Leverage new and existing in-database statistical and data mining capabilitie

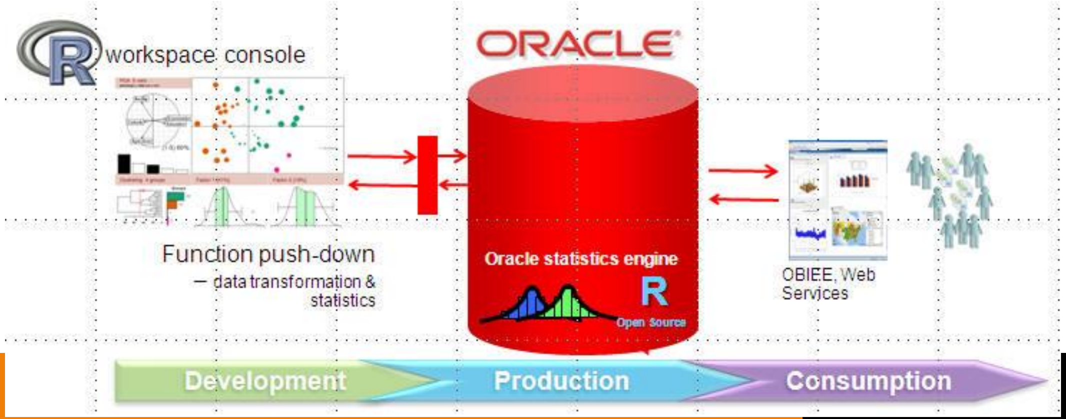
R Engine(s) spawned by Oracle DB

- Database can spawn multiple R engines for database-managed parallelism
- Efficient data transfer to spawned R engines
- Emulate map-reduce style algorithms and applications
- Enables "lights-out"

How Oracle R Enterprise Works

- *ORE Computation Engines*

1. Oracle R Enterprise (ORE) Transparency Layer
   - Traps all R commands and scripts prior to execution and looks for opportunities to function ship them to the database for native execution
   - ORE transparency layer converts R commands/scripts into SQL equivalents and thereby leverages the database as a compute engine.

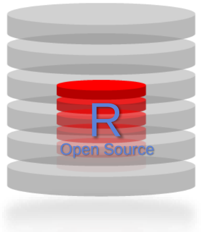


How Oracle R Enterprise Works

- *ORE Computation Engines*

2. In-Database Statistics Engine

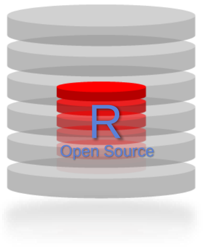
- – Significantly extends the Oracle Database's library of statistical functions and advanced analytical computations
- – Provides support for the complete R language and statistical functions found in Base R and selected R packages based on customer usage
  - Open source packages - written entirely in R language with only the functions for which we have implemented SQL counterparts - can be translated to execute in database.
- – Without anything visibly different to the R users, their R commands and scripts are oftentimes accelerated by a factor of 10-100x



All Base R functions
R Multiple Regression
…. Driven by customers

ORE Functions
- ORE SUMMARY
- ORE FREQUENCY
- ORE CORR
- ORE UNIVARITE
- ORE CROSSTAB
- ORE RANK
- ORE SORT
- …

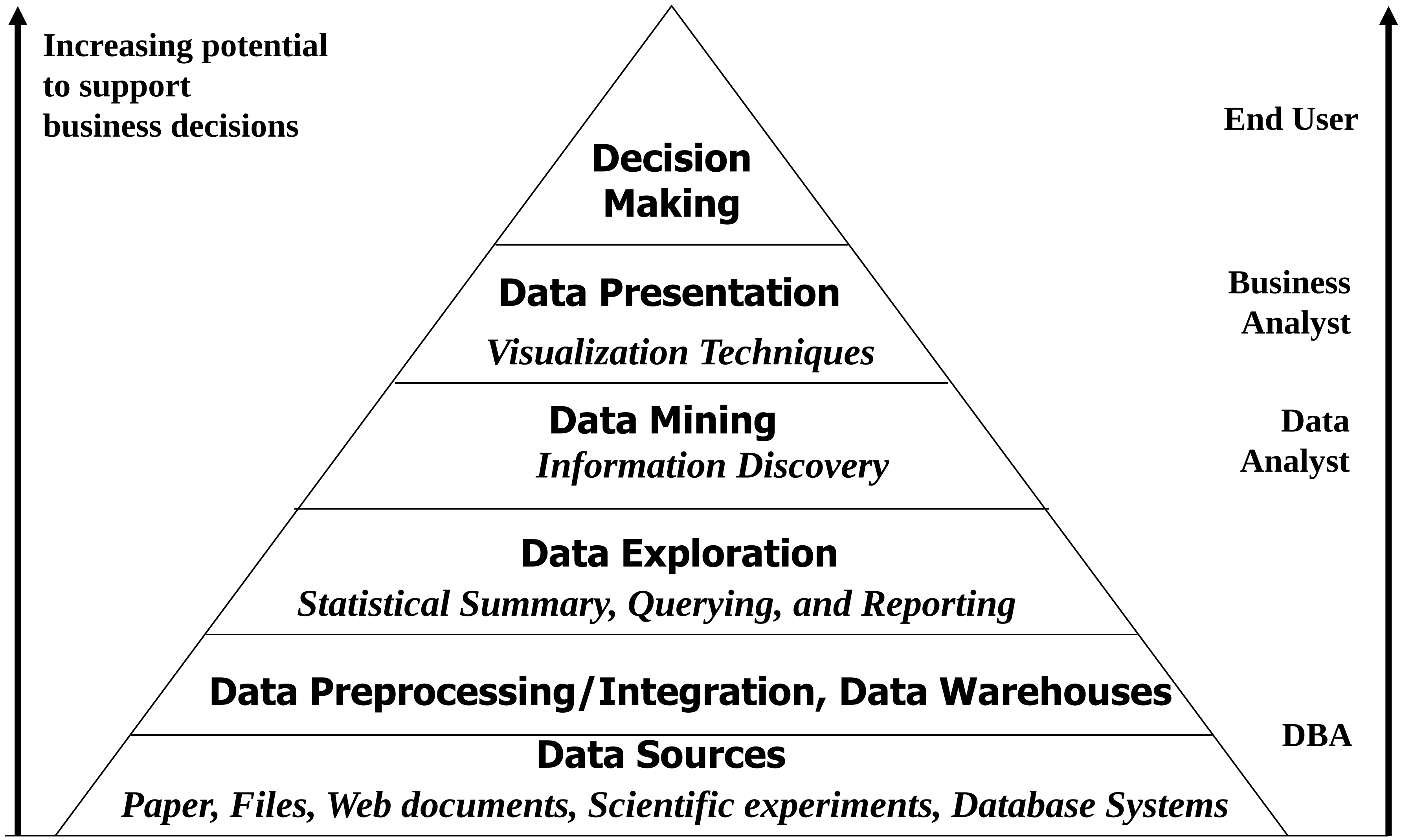
How Oracle R Enterprise Works



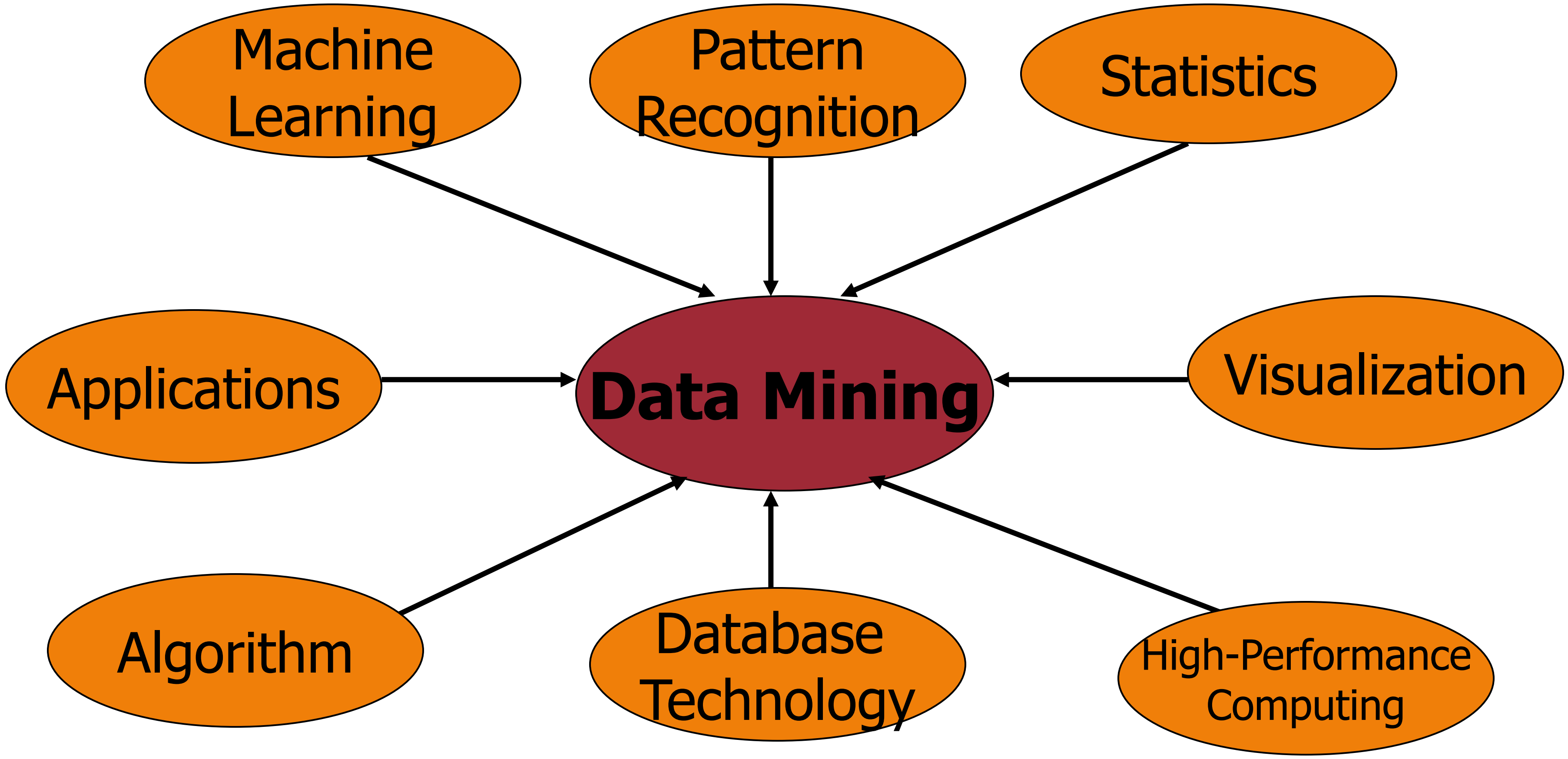
- *ORE Computation Engines*

3. Embedded R Engine
   - For R functions not able to be mapped to native in-database functions, Oracle R Enterprise makes "extproc" remote procedure calls to multiple R engines running on multiple database servers/nodes
   - This Oracle R Enterprise embedded layer uses the database as a data provider providing data level parallelism to R code

Data Mining in Business Intelligence

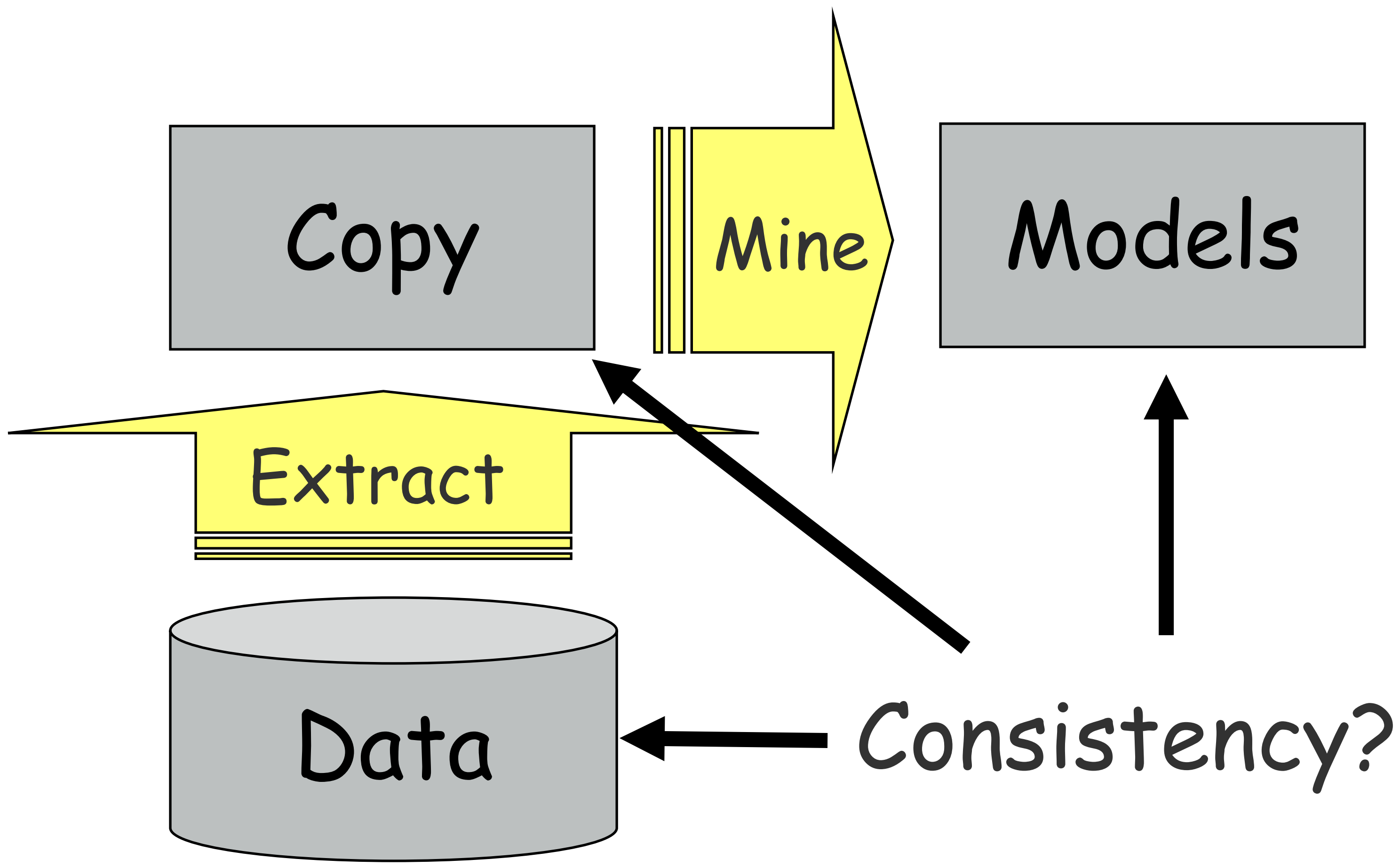

Increasing potential to support business decisions
Decision Making
Data Presentation
Visualization Techniques
Data Mining
Information Discovery
Data Exploration
Statistical Summary, Querying, and Reporting
Data Preprocessing/Integration, Data Warehouses
Data Sources
Paper, Files, Web documents, Scientific experiments, Database Systems
End User
Business Analyst
Data Analyst
DBA

Data Mining: Confluence of Multiple Disciplines



DBMS Support for DM

Why Integrate DM into a DBMS?



Integration Objectives

- Avoid isolation of querying from mining
  - Difficult to do "ad-hoc" mining
- Provide simple programming approach to creating and using DM models

Analysts (users)

- Make it possible to add new models
- Make it possible to add new, scalable algorithms

DM Vendors

SQL/MM: Data Mining

- A collection of classes that provide a standard interface for invoking DM algorithms from SQL systems.
- Four data models are supported:
  - Frequent itemsets, association rules
  - Clusters
  - Regression trees
  - Classification trees

Data Import and Export using R

- Can import any data into R and export R objects to other formats as well. This include:

1- Save and Load R Data

2- Import from and Export to .CSV File

3- Import Data from SAS

4- Import/Export via ODBC

 - Read from Databases

 - Output to and Input from EXCEL Files

5- Read and Write EXCEL files with package xlsx

1- Save and Load R Data

- After saved data file in R, then Rdata files can be reloaded into R with load().

```
> a <- 1:10
> b <- letters[1:5]
> save(a, b, file="./data/mydatafile.Rdata")
> rm(a, b)
> load("./data/mydatafile.Rdata")
➢ print(a)
        [1] 1 2 3 4 5 6 7 8 9 10
> print(b)
        [1] "a" "b" "c" "d" "e"
```

2-Import from and Export to .CSV Files

- Data frame is a data format that we mostly deal with in R. A data frame is similar to a table in databases, with each row being an observation (or record) and each column beding a variable (or feature).
- At first, we create three vectors, an integer vector, a numeric (real) vector and a character vector, use function data.frame() to build them into dataframe df1 and save it into a .CSV file with write.csv().

2-Import from and Export to .CSV Files

- Function sample(5) produces a random sample of five numbers out of 1 to 5.
- Column names in the data frame are then set with function names().
- After that, we reload the data frame from the file to a new data frame df2 with read.csv().
- Note that the very first column printed below is the row names, created automatically by R.

2-Import from and Export to .CSV Files

```
> var1 <- sample(5)
> var2 <- var1 / 10
> var3 <- c("R", "and", "Data Mining", "Examples", "Case
Studies")
> df1 <- data.frame(var1, var2, var3)
> names(df1) <- c("Var.Int", "Var.Num", "Var.Char") >
write.csv(df1, "./data/mydatafile3.csv", row.names = FALSE)
> df2 <- read.csv("./data/mydatafile3.csv")
> print(df2)
```

3- Import Data from SAS

- The function read.ssd() for importing SAS datasets (.sas7bdat files) into R. Three main steps:

- SAS must be available on your computer, and read.ssd() will call SAS to read SAS datasets and import them into R.
- The file name of a SAS dataset has to be no longer than eight characters. Otherwise, the importing would fail. There is no such a limit when importing from a .CSV file.
- During importing, variable names longer than eight characters are truncated to eight characters, which often makes it difficult to know the meanings of variables. One way to get around this issue is to import variable names separately from a .CSV file, which keeps full names of variables.

3- Import Data from SAS

- Using the simple two steps can generate empty CSV file, as follow:

1. Create an empty SAS table dumVariables from dumData as follows.

```
data work.dumVariables;
        set work.dumData(obs=0);
 run;
```

2. Export table dumVariables as a .CSV file.

3- Import Data from SAS

- To importing data from a SAS dataset. Assume that there is a SAS data file dumData.sas7bdat and a .CSV file dumVariables.csv in folder “Current working directory/data”.

```
> library(foreign) # for importing SAS data
> # the path of SAS on your computer
> sashome <- "C:/Program Files/SAS/SASFoundation/9.2"
> filepath <- "./data"
> # filename should be no more than 8 characters, without extension
> fileName <- "mySasDataFile"
> # read data from a SAS dataset
> a <- read.ssd(file.path(filepath), fileName, sascmd=file.path(sashome, "sas.exe"))
> print(a)
```

3- Import Data from SAS

- The variable names above are truncated. The full names can be imported from a .CSV file with the following code.

```
> # read variable names from a .CSV file
> variableFileName <-
"sasVariableNames.csv"
> myNames <- read.csv(file.path(filepath,
variableFileName))
> names(a) <- names(myNames)
> print(a)
```

4- Import/Export via ODBC

- RODBC package provide a connection to ODBC database.
- First: Read from databases, there are Three functions can used as follow:
  - Function odbcConnect() sets up a connection to database,
  - sqlQuery() sends an SQL query to the database, and
  - odbcClose() closes the connection.

```
> library(RODBC)
> connection <-
odbcConnect(dsn="servername",uid="userid",pwd="******")
> query <- "SELECT * FROM lib.table WHERE ..."
> # or read query from file > # query <-
readChar("data/myQuery.sql", nchars=99999)
> myData <- sqlQuery(connection, query, errors=TRUE)
> odbcClose(connection)
```

4- Import/Export via ODBC

- Second: Output to and Input from EXCEL Files

- where a sheet name needs to be provided in function sqlFetch().

```
> library(RODBC)
> filename <- "data/myExcelFile.xls"
> xlsFile <- odbcConnectExcel(filename, readOnly = FALSE)
> sqlSave(xlsFile, a, rownames = FALSE)
> b <- sqlFetch(xlsFile, "sheetname")
> odbcClose(xlsFile)
```

Note: There might be a limit of 65,536 rows to write to an excel file.

5- Read and Write EXCEL files with package xlsx

- The example below demonstrates creation of an EXCEL file iris.xlsx with three sheets. Function library() loads an R package (or library), and table() returns the frequencies of values in a vector.
- We can see that there are three species, with each having 50 observations. Observations of species “setosa” are extracted first with function subset() and then saved into sheet “setosa” in the EXCEl file with function write.xlsx(). Row names are excluded using row.names=F. Then data of the other two species are saved into the same file, but in different sheets. When writing the second and third sheets, we need to use append=T to add new sheets to the existing file, instead of overwriting it. Finally, we read from sheet “setosa” with function read.xlsx() and show the first six observations with function head().

5- Read and Write EXCEL files with package xlsx

```
> library(xlsx)
> table(iris$Species)
```

```
> setosa <- subset(iris, Species == "setosa")
> write.xlsx(setosa, file="./data/iris.xlsx", sheetName="setosa",
row.names=F)
> versicolor <- subset(iris, Species == "versicolor")
> write.xlsx(versicolor, file="./data/iris.xlsx", sheetName="versicolor", +
row.names=F, append=T)
> virginica <- subset(iris, Species == "virginica")
> write.xlsx(virginica, file="./data/iris.xlsx", sheetName="virginica", +
row.names=F, append=T)
> a <- read.xlsx("./data/iris.xlsx", sheetName="setosa")
> head(a)
```

The End