
ORACLE®

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle's Machine Learning

Workshop / Hands-on Lab with Oracle Database 19c

Olivier Perard
Olivier.perard@oracle.com

Iberia Technology Sales Consulting
January, 2020

Oracle Advanced Analytics

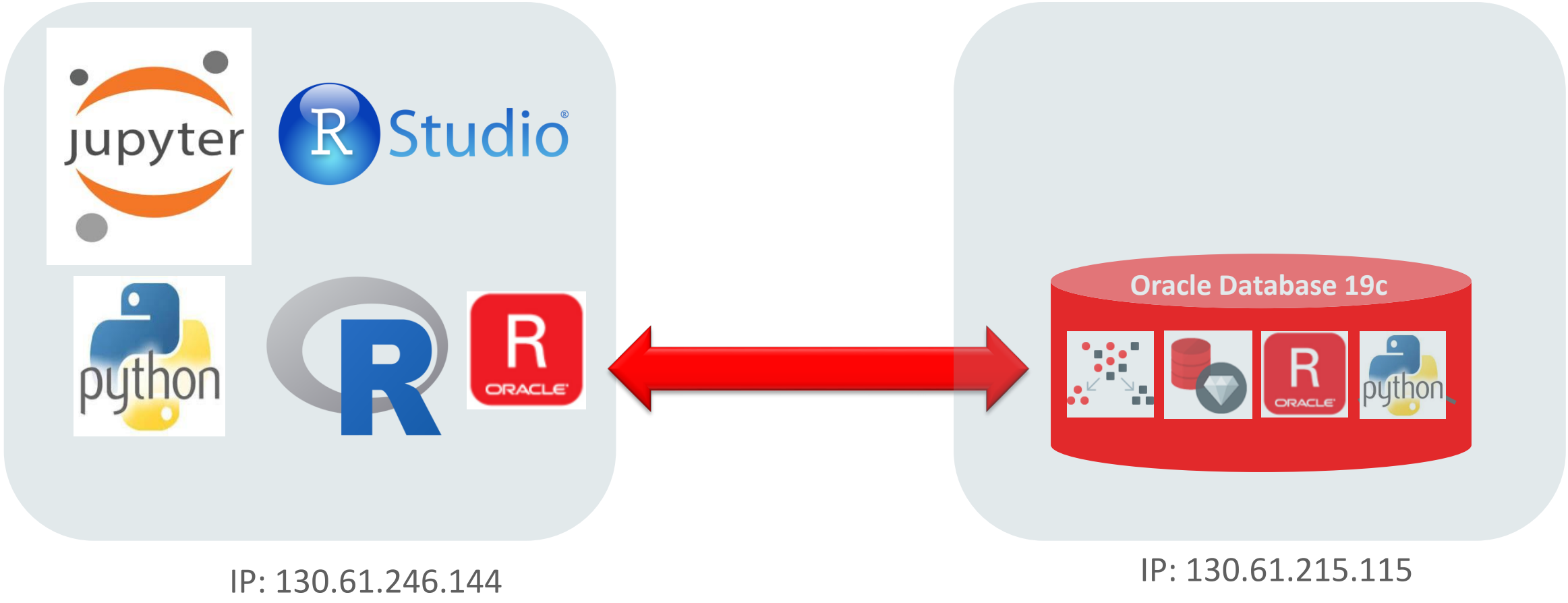
- Documentation:
- https://docs.oracle.com/en/database/oracle/r-enterprise/1.5.1/oread/index.html

- Oracle Open Source:
- https://github.com/oracle

Environment for Hands-on-Lab with Oracle Database 19c

- IP: 130.61.246.144
- RSTUDIO: 8080
- USERS: workshop01 / ….etc…. / workshop20
- PWD: Welcome2020#
- URL: http://130.61.246.144:8080/
- GitHub:https://github.com/operard/workshops/tree/master/oaa

Workshop Environment



IP: 130.61.246.144

IP: 130.61.215.115

Oracle Advanced Analytics

OAA/Oracle R Enterprise (R integration)

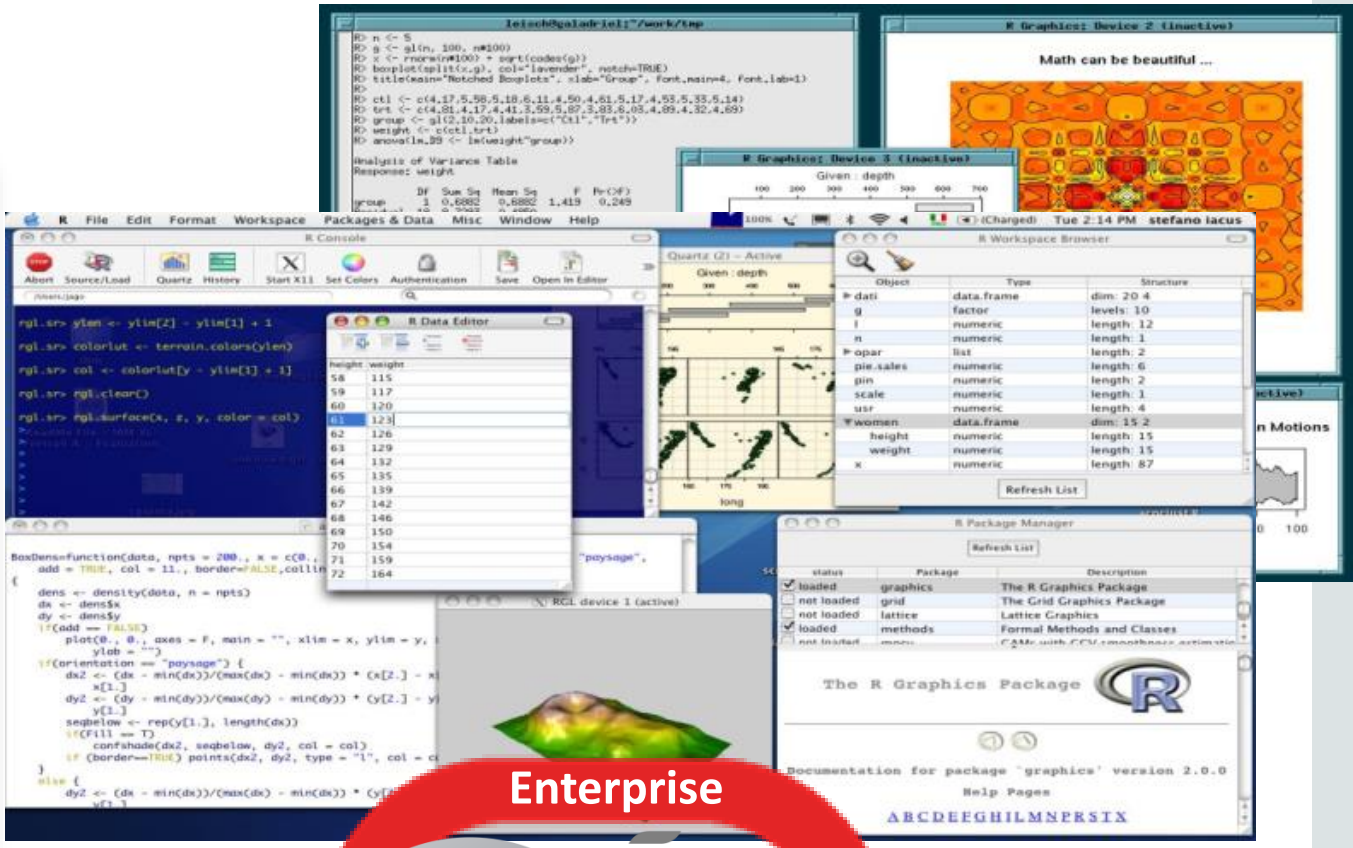
ORACLE

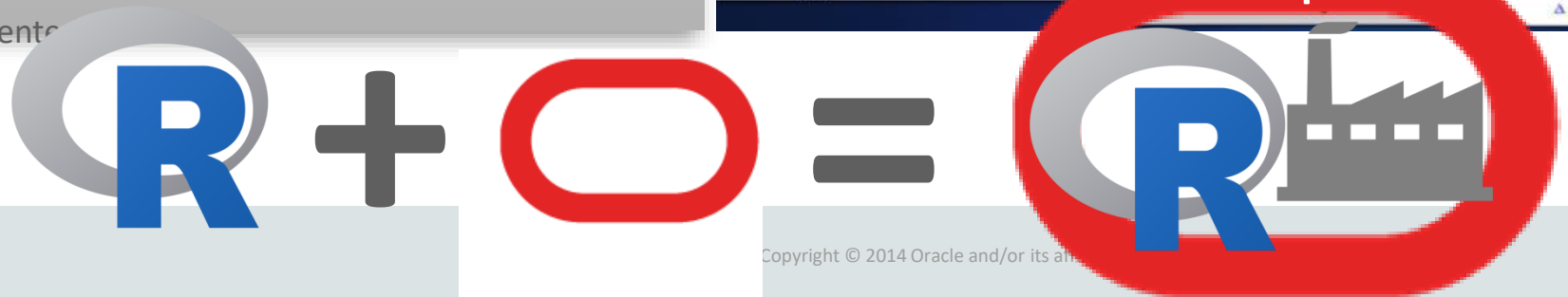
R—Widely Popular

R is a statistics language similar to Base SAS or SPSS statistics

R environment

- Strengths
  - Powerful & Extensible
  - Graphical & Extensive statistics
  - Free—open source (CRAN + 9000 components)
  - Standard for Data Scientist
- Challenges
  - Memory constrained
  - Single threaded
  - Outer loop—slows down process
  - Not Enterprise Orient

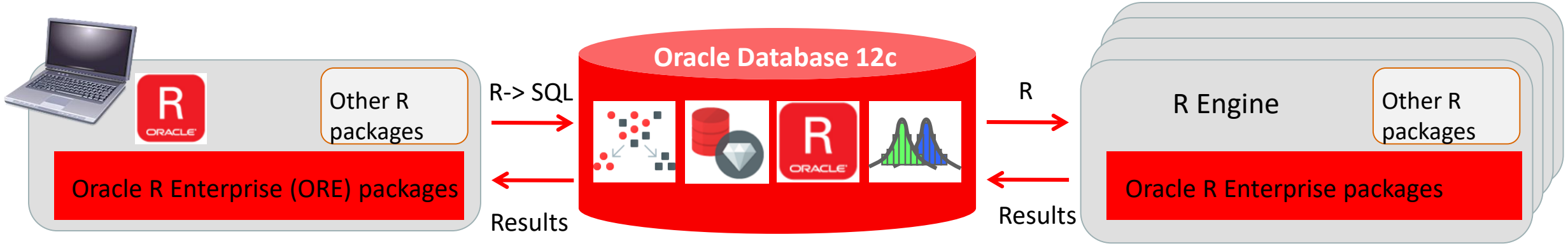




Oracle Advanced Analytics

How Oracle R Enterprise Compute Engines Work

Advanced Analytics



1 R-> SQL Transparency "Push-Down"

- R language for interaction with the database
- R-SQL Transparency Framework overloads R functions for scalable in-database execution
- Function overload for data selection, manipulation and transforms
- Interactive display of graphical results and flow control as in standard R
- Submit user-defined R functions for execution at database server under control of Oracle Database

2 In-Database Adv Analytical SQL Functions

- 15+ Powerful data mining algorithms (regression, clustering, AR, DT, etc._
- Run Oracle Data Mining SQL data mining functioning (ORE.odmSVM, ORE.odmDT, etc.)
- Speak "R" but executes as proprietary in-database SQL functions—machine learning algorithms and statistical functions
- Leverage database strengths: SQL parallelism, scale to large datasets, security
- Access big data in Database and Hadoop via SQL, R, and Big Data SQL

3 Embedded R Package Callouts

- R Engine(s) spawned by Oracle DB for database-managed parallelism
- ore.groupApply high performance scoring
- Efficient data transfer to spawned R engines
- Emulate map-reduce style algorithms and applications
- Enables production deployment and automated execution of R scripts

ORACLE

08. Advanced Analytics

8.0 Oracle BI EE 12c Advanced Analytics

8.00 Advanced Analytics:
Overview, Binning, Trendline, Forecast, Outlier, Cluster, Regression

8.01 Visualizing using R:
Interactive Boxplot, Interactive 3D Scatter, Bubble Chart Grid, Variable Width Bar, Random Dots

8.02 Functional Examples (Evaluate Script):
Text Sentiment, Text Term Frequency, Timeseries Decomposition, Market Basket Analysis, Collaborative Filtering, Delay Prediction (Precomputed Model)

8.1 Descriptive Analytics

8.10 Binning and Tiling:
Comparative Dist, Distr. II, Distribution, Large Data, Ntiling, Percentiles, Width Bucket

8.11 Comparative Analysis:
Age Pyramid, Benchmark, Index To Avg, Indexing, Lift, Tiering, TopN, TopN History

8.12 Descriptive Stats:
8020, Control Chart, Correlation, Data Density, Deviants, Scatter, See Also, StdDev, Variability

8.13 History and Trend:
History, Seasonality, Trend Lines, Trending

8.2 Oracle Database Analytics

8.20 Text Analytics:
Cost Analysis per Token, Cost Per Token Frequency, Text Aggregation, Text Classes, Text Classification, Text Filtering, Words Distribution

8.21 Temporal and Time:
Months Between, Temporal Query (12c Session), Temporal Query (12c), TimeZone with DST, Timezone Conversion

8.22 Analytic Clauses:
Frequent Itemset, Model Projection, Pattern Detection (12c), Projection Interactive

8.23 Other DB Analytics:
CLOB Datatype, Column Statistics, DB Web Services, Text Aggregation, JSON Parsing, Approximate Count Distinct

8.3 Oracle Data Mining

8.30 ODM Classification:
Classification Tree, Geo LTV Prediction, LTV Details, LTV Prediction, LTV Probabilities, LTV What If Scoring, Dynamic Classification (12c)

8.31 ODM Regression:
Regression, Regression Variance, Variance Heatmap, Dyn Predictive Regression (12c)

8.32 ODM Clustering, Association and Attribute Importance:
Clustering, Market Basket Analysis, Attribute Importance

8.33 ODM Mining On-the-fly:
Anomaly Influencers (12c), Dyn Anomaly Detection (12c), Dyn Predictive Regression (12c), Dynamic Classification (12c), Dyn Prediction Delay Grp (12c In-mem), Dyn Anomaly Analysis (12c In-mem)

8.34 ODM Data Miner Workflows:
List of Examples, Overview

8.4 Oracle R Enterprise

8.40 ORE Integration:
R Integration, R End-User Interaction, R Workbench, R Results Object in RPD, BIP Sourcing from R, Quality Control Chart (BIP)

8.41 ORE Time Series:
T. Series Decomposition, T. Series Forecasting, T. Series Moving Average, T. Series Auto ARIMA, T. Series Holt, T. Series SES, T. Series ACF PACF

8.42 ORE Datamining:
Multivariate Adaptive Regression Splines, Support Vector Machines, Association Rules, Variable Importance, Clustering with k-Means++, In-Database Associations, ORE GroupApply, ORE IndexApply

8.43 ORE Visualizations:
Quality Control Chart, Boxplot, Cond. Histogram, Corr. Matrix Circles, Corr. Matrix Ellipses, Heatmap, Multipanel Geo Lattice, Volcano, sinc Perspective

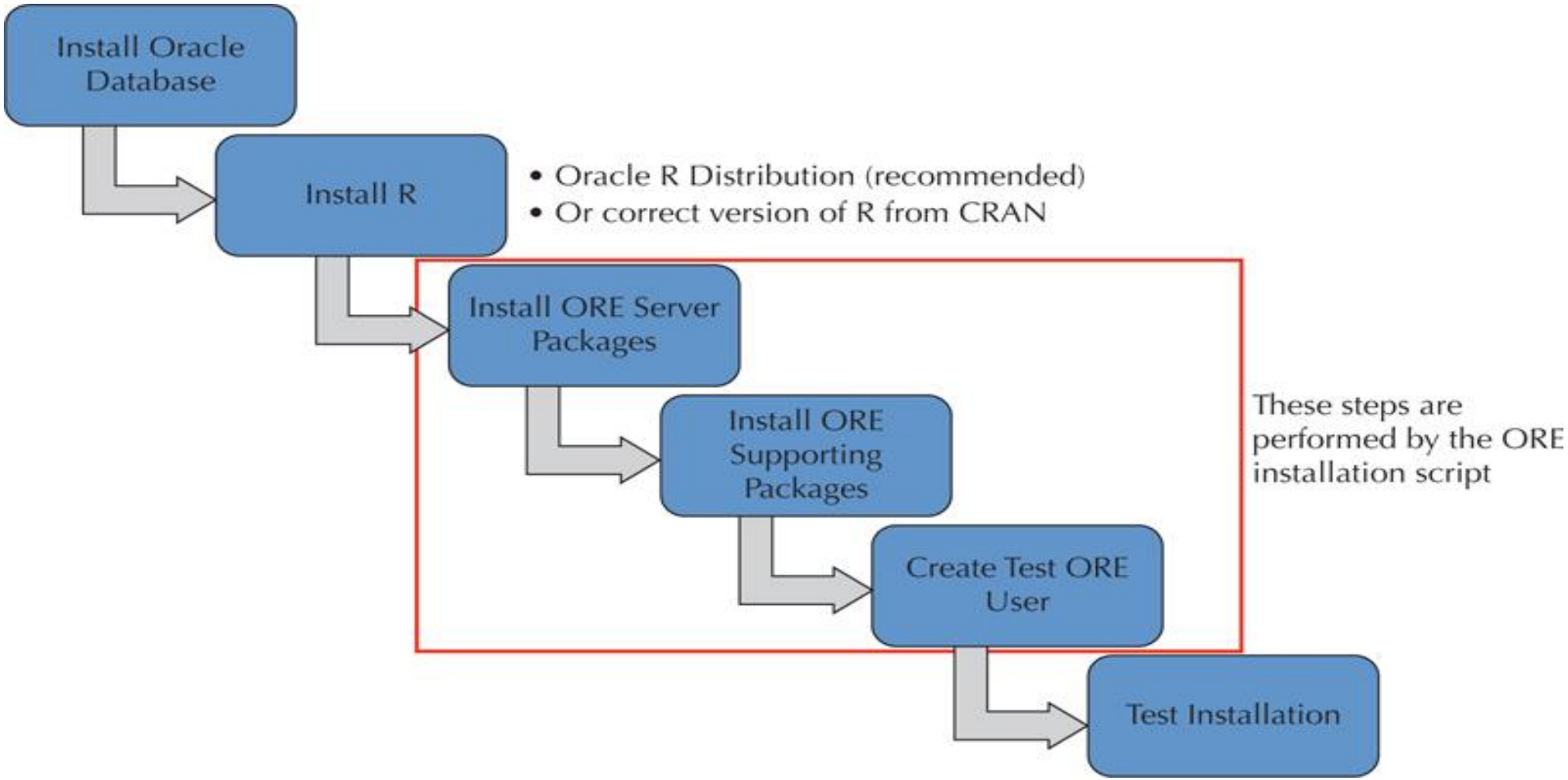
ORE Packages

ORE Package	Description
ORE	The top-level package for Oracle R Enterprise
OREbase	Corresponds to R's base package
OREcommon	Contains common low-level functionality for Oracle R Enterprise
OREdm	Exposes Oracle Data Mining algorithms
OREeda	Exploratory data analysis package
OREembed	Supports the embedded execution of R in the database
OREgraphics	Corresponds to the open source R graphics package
OREmodels	Contains functions for advanced analytical modeling
OREpredict	Enables scoring data in Oracle DB using R models
OREserver	Contains functions for an Oracle R Enterprise Server
OREstats	Corresponds to R's stat package
ORExml	Supports XML translation between R and Oracle database

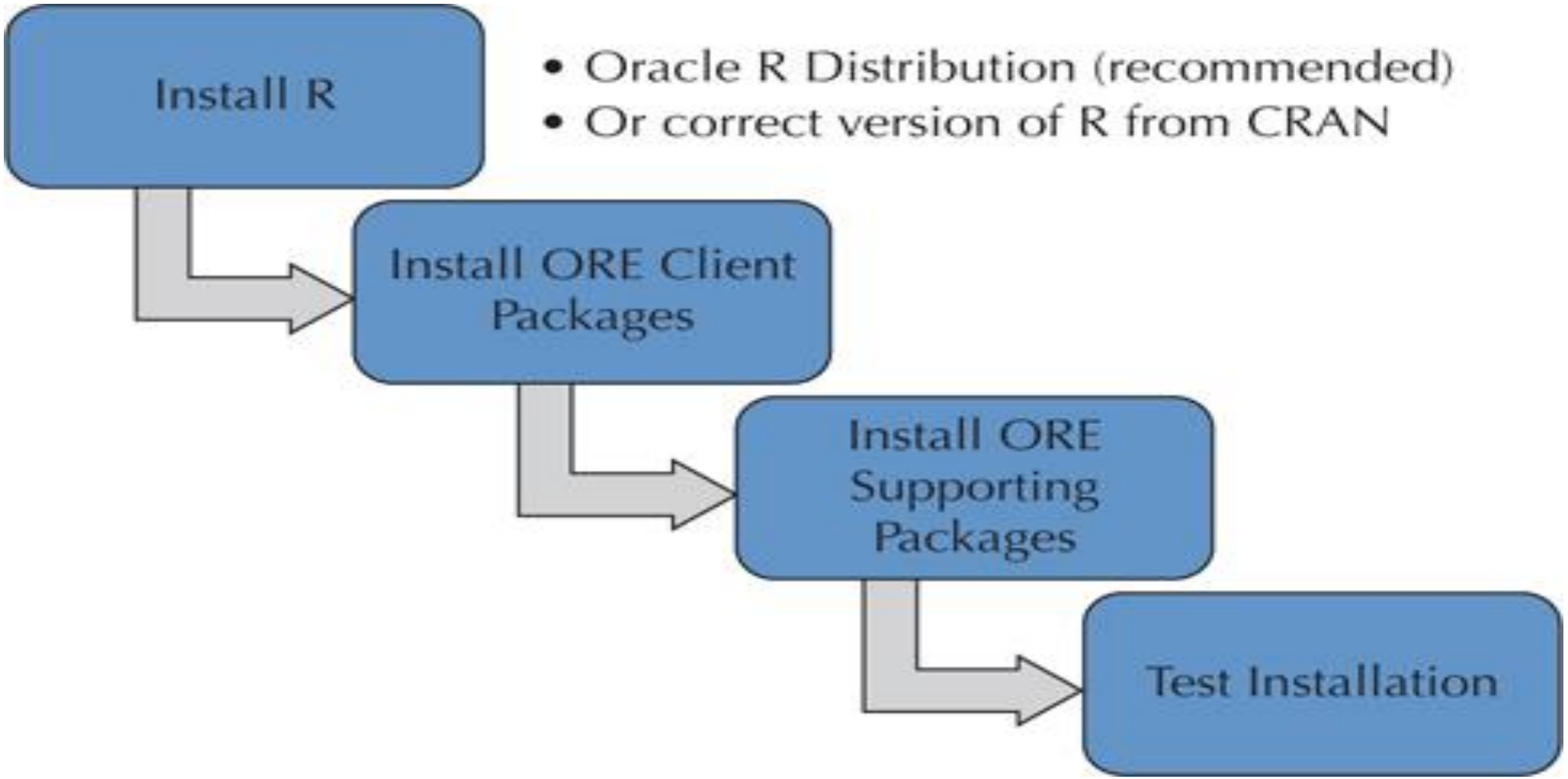
More Packages

Package Name	Package Description
arules	Allows for frequent item sets and association rules. Provides support for representing, manipulating, and analyzing the transactional data and the patterns of the results.
Cairo	Supports graphic rendering on Oracle Enterprise Server.
DBI	The database interface definition for communicating between R and the Oracle Database.
png	Supports the reading and writing of PNG images for Oracle R Enterprise objects.
randomForest	Supports the ORE implementation of randomForests.
ROracle	The Oracle Database interface for R-based Oracle Call Interface (OCI).
statmod	Provides a variety of statistical modeling functions

Oracle R Enterprise installation steps on the Database Server



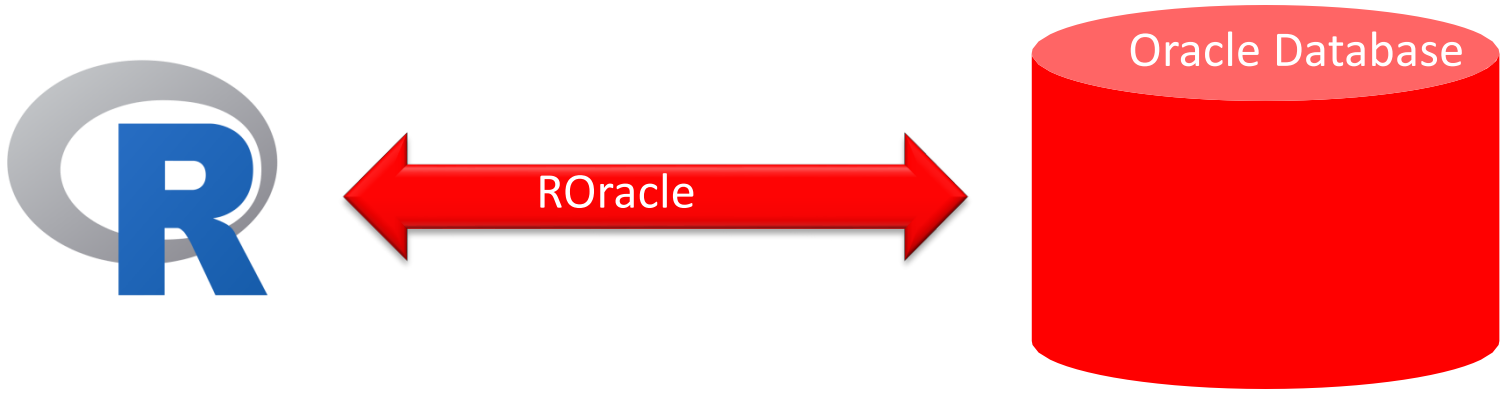
Client installation of Oracle R Enterprise



ROracle Package

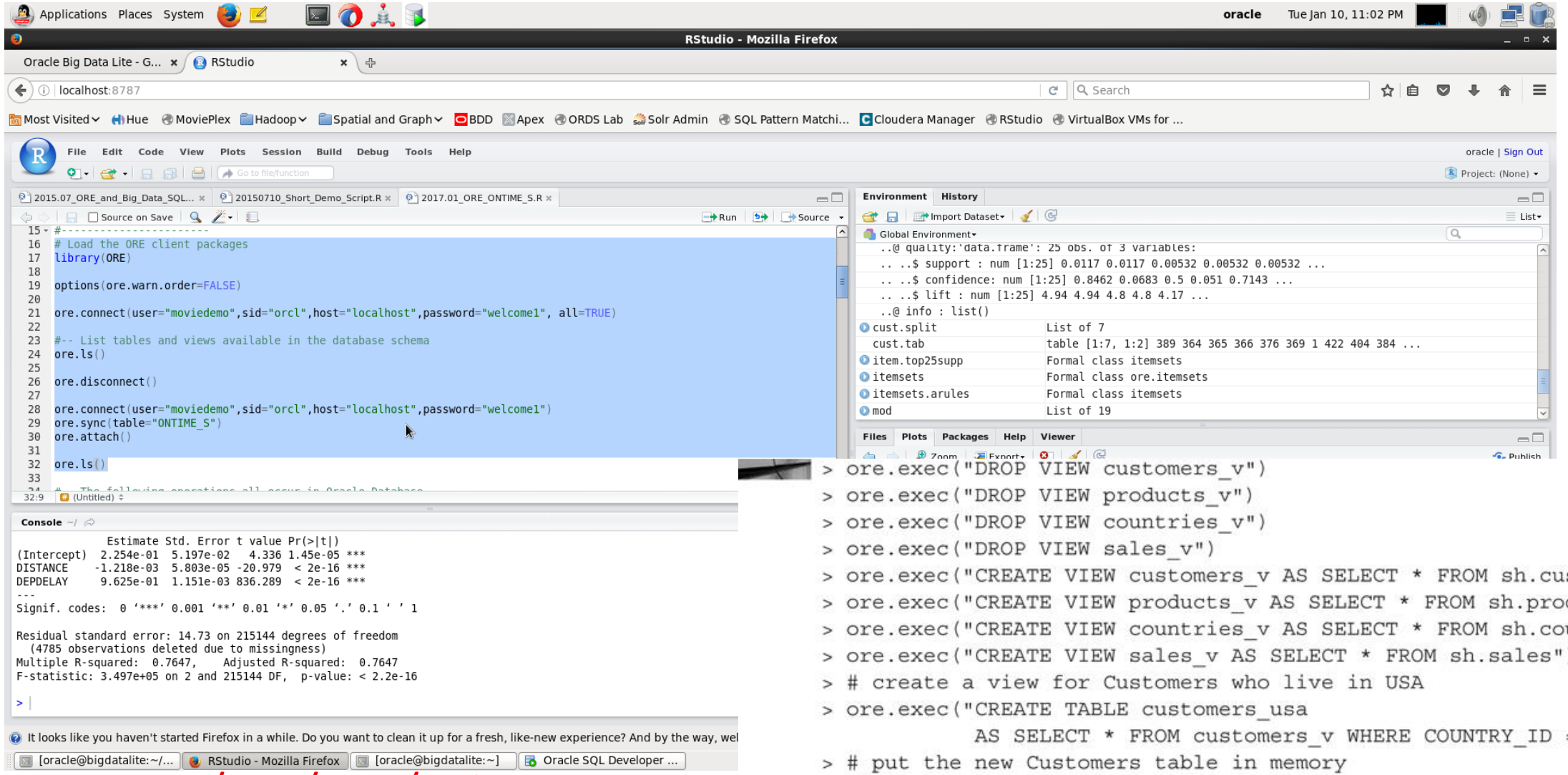
ROracle

DEMO : /tmp/OAA/script_roracle.R



ORE: Transparency Layer

Connecting to Oracle Database




```
> ore.exec("DROP VIEW customers_v")
> ore.exec("DROP VIEW products_v")
> ore.exec("DROP VIEW countries_v")
> ore.exec("DROP VIEW sales_v")
> ore.exec("CREATE VIEW customers_v AS SELECT * FROM sh.customers")
> ore.exec("CREATE VIEW products_v AS SELECT * FROM sh.products")
> ore.exec("CREATE VIEW countries_v AS SELECT * FROM sh.countries")
> ore.exec("CREATE VIEW sales_v AS SELECT * FROM sh.sales")
> # create a view for Customers who live in USA
> ore.exec("CREATE TABLE customers_usa
            AS SELECT * FROM customers_v WHERE COUNTRY_ID = 52790")
> # put the new Customers table in memory
> ore.exec("ALTER TABLE customers_usa inmemory")
```

DEMO : /tmp/OAA/script_ore_0001.R

Embedded R Execution: R Interface

ORACLE

Embedded R Script Execution – R Interface

R Interface function	Purpose
ore.doEval()	Invoke stand-alone R script
ore.tableApply()	Invoke R script with ore.frame as input
ore.rowApply()	Invoke R script on one row at a time, or multiple rows in chunks from ore.frame
ore.groupApply()	Invoke R script on data partitioned by grouping column of an ore.frame
ore.indexApply()	Invoke R script N times
ore.scriptCreate()	Create an R script in the database
ore.scriptDrop()	Drop an R script in the database

ORE's embedded R execution provides specialized operations that support both:

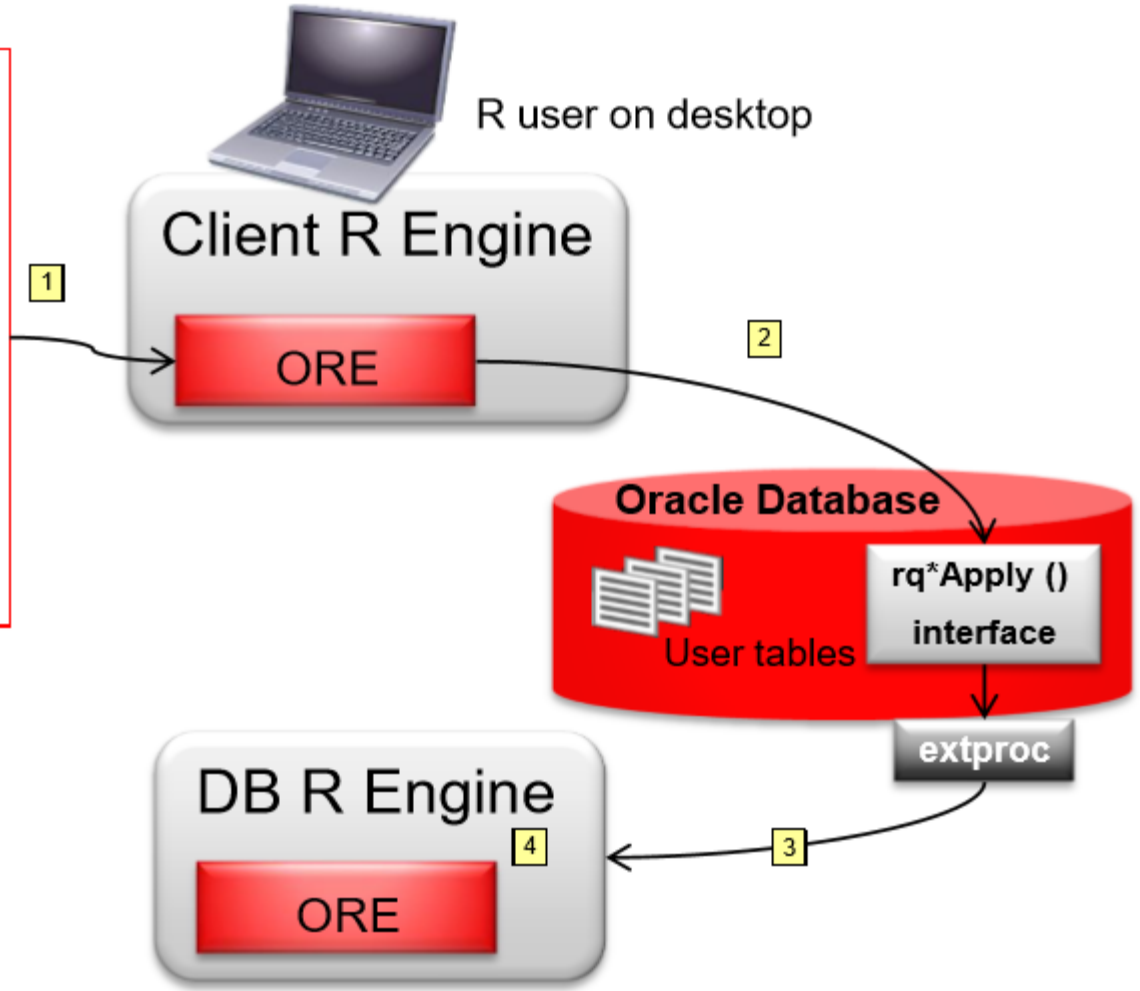
- **data-parallel** (ore.groupApply, ore.rowApply)
- **task-parallel** (ore.indexApply) execution where users can specify the degree of parallelism, i.e., the number of parallel R engines desired.

Ore.doEval – invoking a simple R Script

```
res <-
   ore.doEval(function (num = 10, scale = 100) {
           ID <- seq(num)
           data.frame(ID = ID, RES = ID / scale)
           })
class(res)
res
local_res <- ore.pull(res)
class(local_res)
local_res
```

Goal: scales the first n integers by value provided

Result: a serialized R data.frame



DEMO : 20150710_Short_Demo_Script.R

Ore.doEval – using R Script

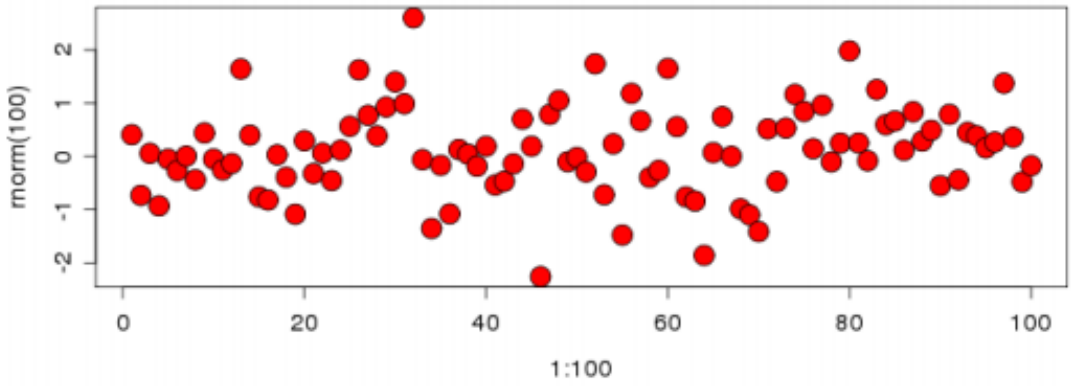
```
ore.scriptDrop("SimpleScript1")
ore.scriptCreate("SimpleScript1",
          function (num = 10, scale = 100) {
            ID <- seq(num)
            data.frame(ID = ID, RES = ID / scale)
            })
res <- ore.doEval(FUN.NAME="SimpleScript1",
                  num = 20, scale = 1000)
```

```
R> ore.scriptDrop("SimpleScript1")
R> ore.scriptCreate("SimpleScript1", function (num=10, scale=100) {
+    ID <- seq(num)
+    data.frame(ID=ID, RES=ID/scale)
+    })
R>
R> ore.doEval(FUN.NAME="SimpleScript1", num=20, scale=1000)
   ID   RES
1   1 0.001
2   2 0.002
3   3 0.003
4   4 0.004
5   5 0.005
6   6 0.006
7   7 0.007
8   8 0.008
9   9 0.009
10 10 0.010
11 11 0.011
12 12 0.012
13 13 0.013
14 14 0.014
15 15 0.015
16 16 0.016
17 17 0.017
18 18 0.018
19 19 0.019
20 20 0.020
```

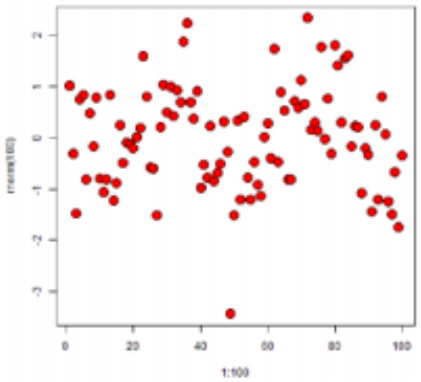
Production Deployment: Same R Function, multiple use

```
begin
  sys.rqScriptDrop('RandomRedDots');
  sys.rqScriptCreate('RandomRedDots',
 'function(){
    id <- 1:10
    plot(1:100,rnorm(100),pch=21,bg="red",cex =2)
    data.frame(id=id, val=id / 100)
  }');
end;
/
```

```
select     value
from       table(rqEval( NULL,'XML', 'RandomRedDots'));

select     ID, IMAGE
from       table(rqEval( NULL,'PNG', 'RandomRedDots'));

select     *
from       table(rqEval( NULL,
    'select 1 id, 1 val from dual','RandomRedDots'));
```



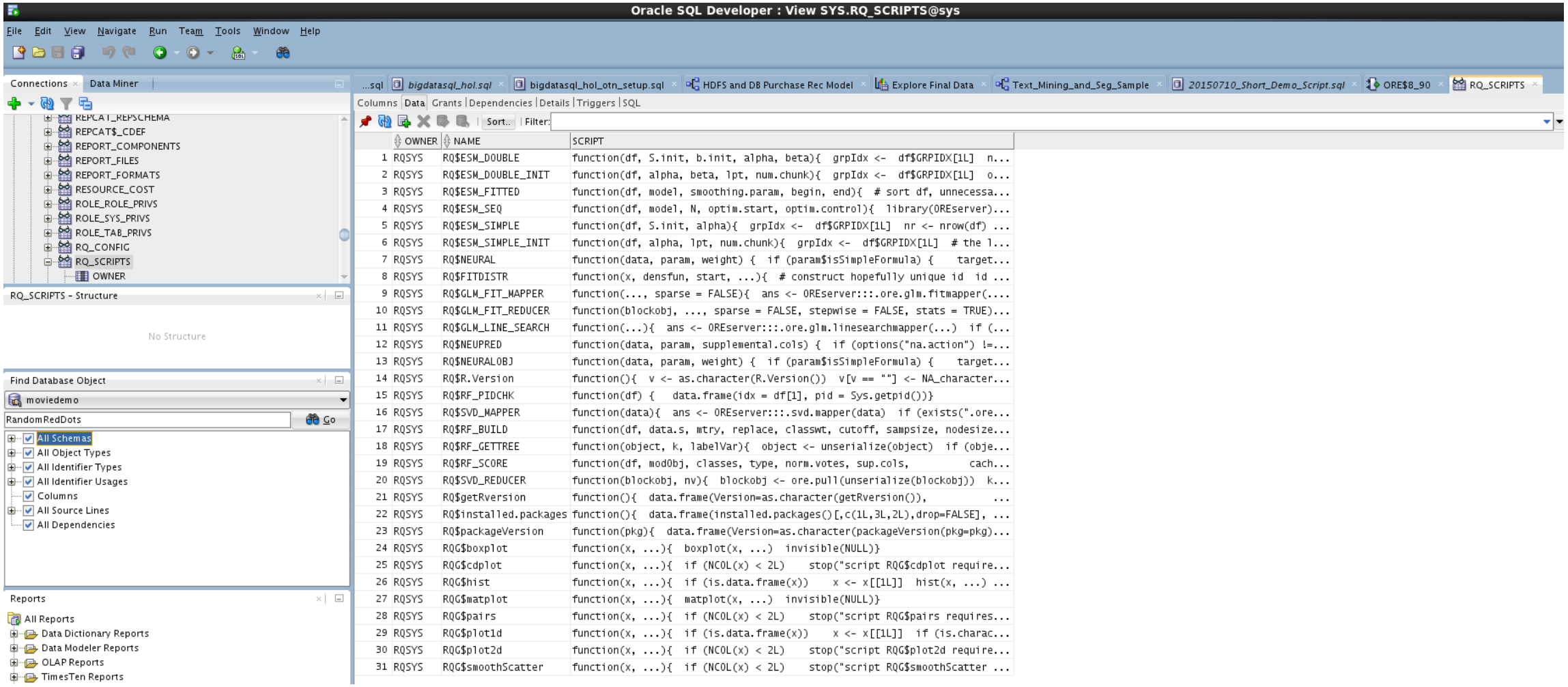

```
> ore.doEval(FUN.NAME="RandomRedDots")
   id  val
1   1 0.01
2   2 0.02
3   3 0.03
4   4 0.04
5   5 0.05
6   6 0.06
7   7 0.07
8   8 0.08
9   9 0.09
10 10 0.10
```



DEMO : 20150710_Short_Demo_Script.sql

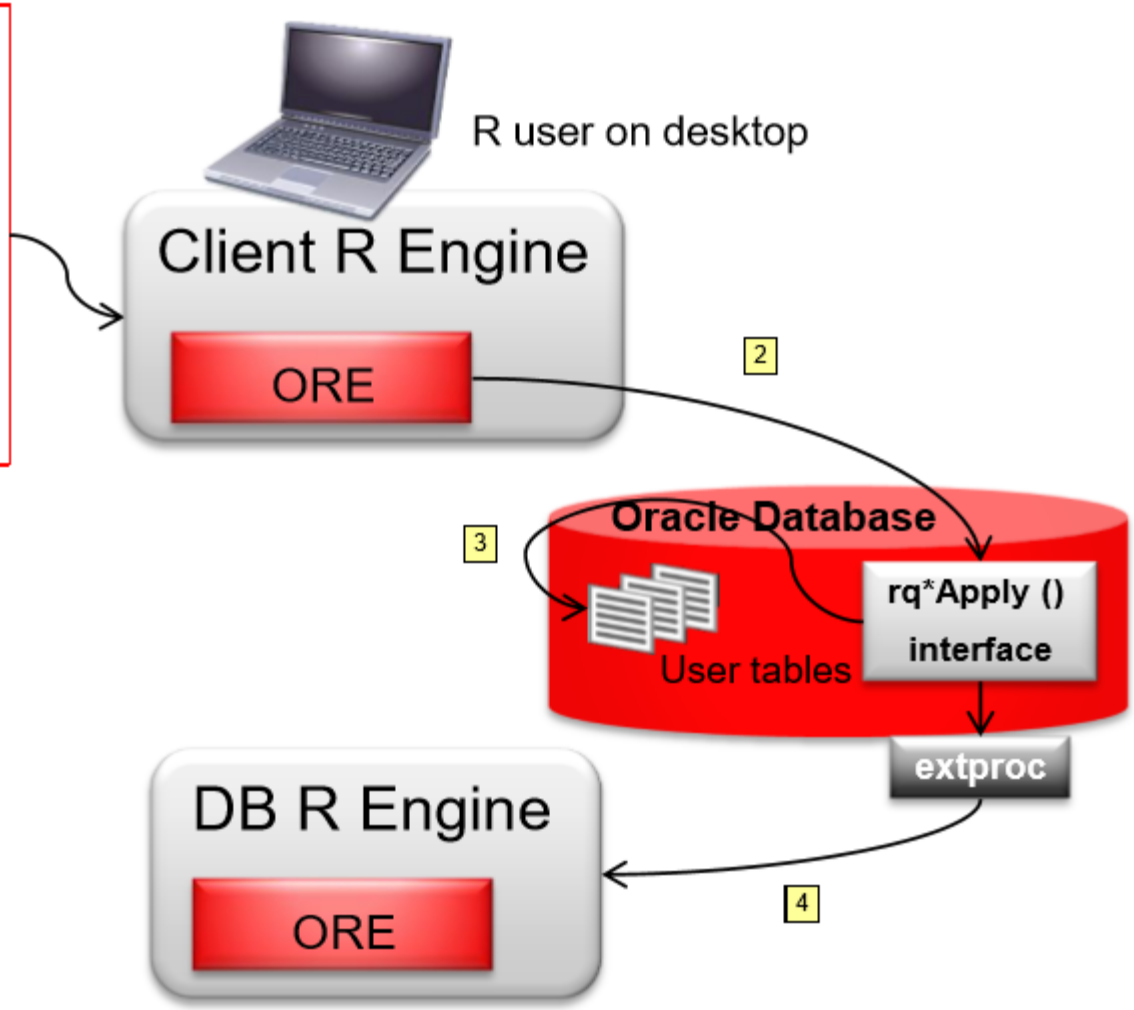
ORACLE

SQL API	Equivalent ORE function	Description
`rqEval`	ore.doEval	Executes a user-defined R script that is passed to it and returns any result generated.
`rqTableEval`	ore.tableApply	Executes a function or script on all rows on a supplied data set.
`"rqGroupEval"`	`ore.groupApply`	Executes a function or script on partitions of the supplied data set. The partitions are defined on one or more attributes of the data set. Parallel execution is supported for each partition. Note: There is no specific function called "`rqGroupEval`." Instead you have to define this function, in a specific way, for your data.
`rqRowEval`	ore.rowApply	Executes a function or script on a defined set of rows (chunks) from the supplied data set. Parallel execution is supported for each set of rows (chunks).



Ore.tableApply – with parameter passing

```
modCoef <- ore.tableApply(
  ONTIME_S[,c("ARRDELAY","DISTANCE","DEPDELAY")],
  function(dat, family) {
    mod <- glm(ARRDELAY ~ DISTANCE + DEPDELAY,
               data=dat, family=family)
    coef(mod)
    }, family=gaussian());
modCoef
```



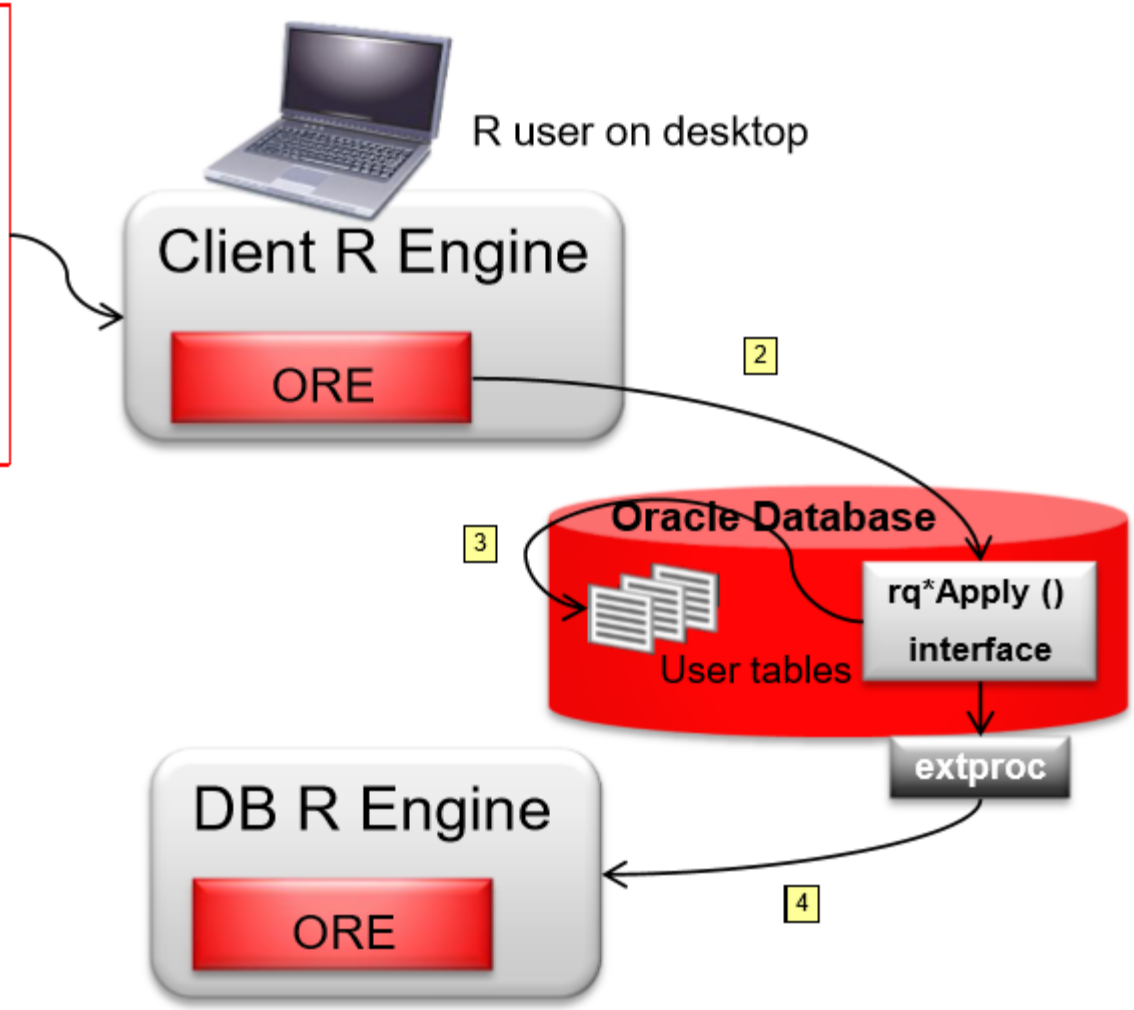
Goal: Build model on data from input cursor with parameter family = gaussian().

Data set loaded into R memory at DB R Engine and passed to function as first argument, *x*

Result coefficient(mod) returned as R object

Ore.tableApply – with parameter passing

```
modCoef <- ore.tableApply(
  ONTIME_S[,c("ARRDELAY","DISTANCE","DEPDELAY")],
  function(dat, family) {
    mod <- glm(ARRDELAY ~ DISTANCE + DEPDELAY,
               data=dat, family=family)
    coef(mod)
    }, family=gaussian());
modCoef
```



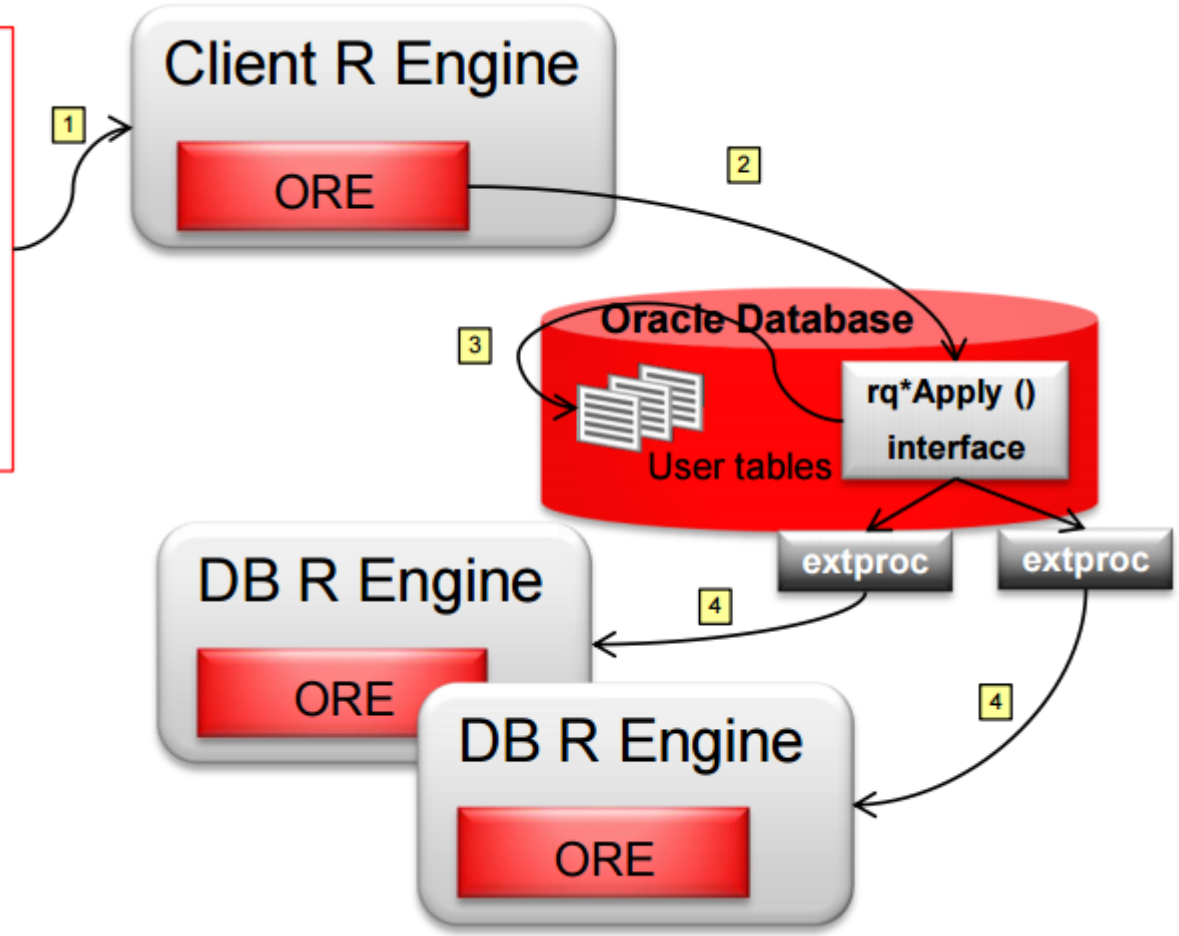
Goal: Build model on data from input cursor with parameter family = gaussian().

Data set loaded into R memory at DB R Engine and passed to function as first argument, *x*

Result coefficient(mod) returned as R object

Ore.groupApply : Partitioned Data Flow

```
modList <- ore.groupApply(
    X=ONTIME_S,
    INDEX=ONTIME_S$DEST,
    function(dat) {
        lm(ARRDELAY ~ DISTANCE + DEPDELAY, dat)
    });
modList_local <- ore.pull(modList)
summary(modList_local$BOS) ## return model for BOS
```



R: Transparency through function overloading

Invoke in-database aggregation function

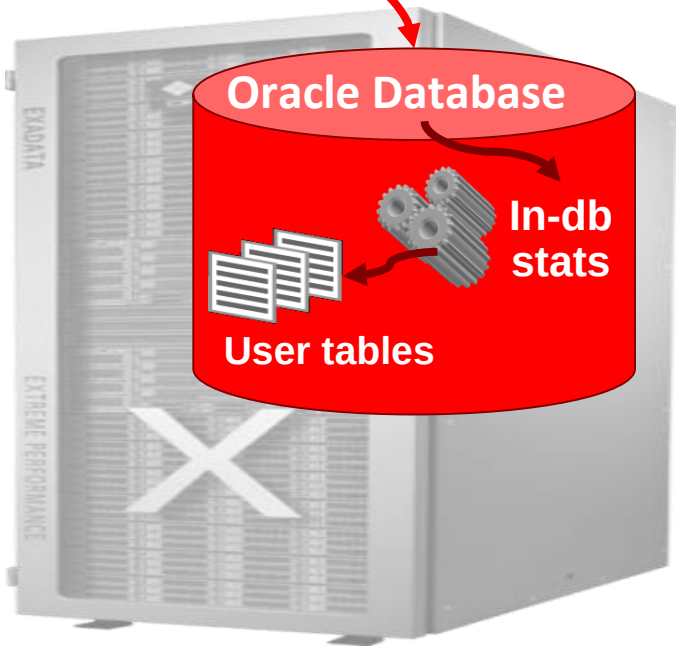
R Language

```
aggdata <- aggregate(ONTIME_S$DEST,
                     by = list(ONTIME_S$DEST),
                     FUN = length)

class(aggdata)

head(aggdata)
```

```
> aggdata <- aggregate(ONTIME_S$DEST,
+                      by = list(ONTIME_S$DEST),
+                      FUN = length)
> class(aggdata)
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
> head(aggdata)
   Group.1    x
1 ABE      237
2 ABI       34
3 ABQ     1357
4 ABY       10
5 ACK        3
6 ACT       33
```

Oracle SQL

```
select DEST, count(*)
from ONTIME_S
group by DEST
```



R: Transparency through function overloading

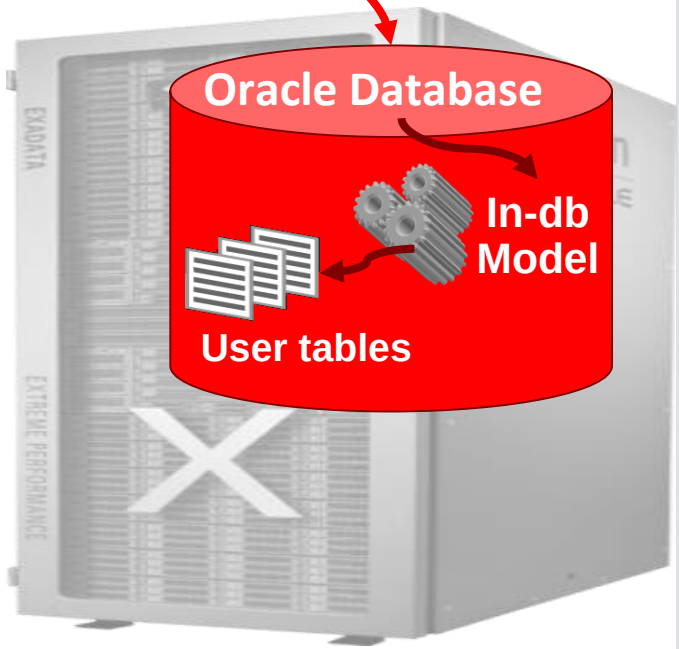
Invoke in-database Logistic Regression Model

R Language

```
form <-DAYS_ACCESS_SURGERY_LT_2~AGE_GROUP+GENDER+
HEART_FAILURE_P_EDEMA+CO_OBSTR_P_D+
ISCHEMIC_HEART_D_A+ISCHEMIC_HEART_D_C+HYPERTENSION+D
IABETES_COMPLIC

mod <- ore.odmGLM(formula=form, data=HIPFINAL,
                  type="logistic", ridge=TRUE,
                  auto.data.prep=FALSE)
```

Oracle PL/SQL

```
BEGIN
DBMS_DATA_MINING.CREATE_MODEL(
 model_name => 'GLM_MOD',
 mining_function =>
dbms_data_mining.classification
...
```




```
>mod
Response: DAYS_ACCESS_SURGERY_LT_2 == "1"
Call: ore.odmGLM(formula = form, data=HIPFINAL, type="logistic", ridge = TRUE,
auto.data.prep = FALSE)
Coefficients:
      (Intercept)          AGE_GROUP             GENDER  HEART_FAILURE_P_EDEMA
         -1.95550           -0.08139            0.20156                1.12284
     CO_OBSTR_P_D ISCHEMIC_HEART_D_A ISCHEMIC_HEART_D_C          HYPERTENSION
          0.58242            1.27826            0.47849                0.25116
 DIABETES_COMPLIC
          0.22427
Degrees of Freedom: 8958 Total (i.e. Null); 8950
Residual Null Deviance: 6917
Residual Deviance: 6700 AIC: 6718
```

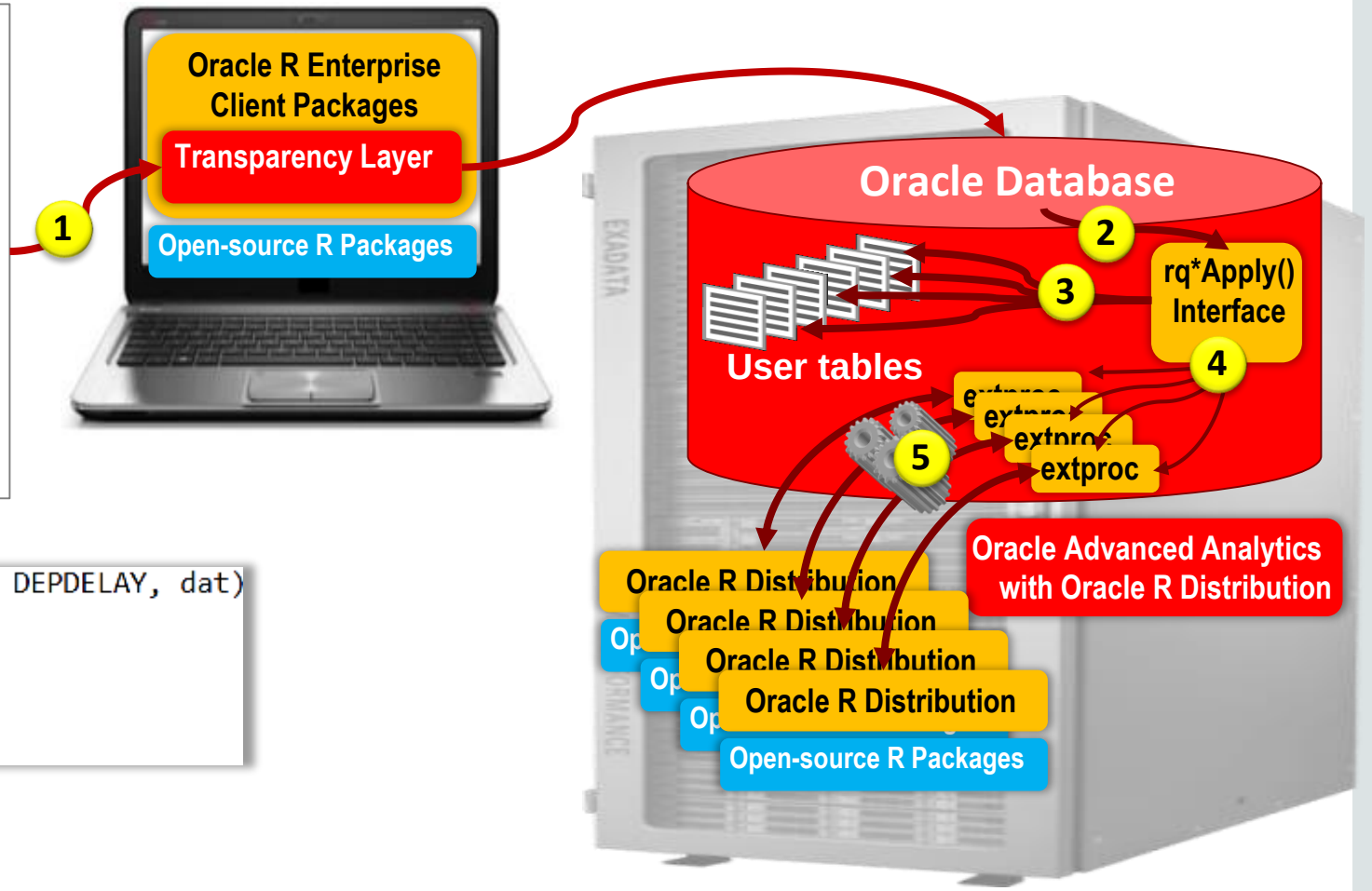
R: Transparency through function overloading

Invoke in-database aggregation function

R Language

```
modList <- ore.groupApply(
    X = ONTIME_S,
    INDEX = ONTIME_S$DEST,
    function(dat) {
      library(biglm)
      biglm(ARRDELAY ~ DISTANCE + DEPDELAY, dat)
    });
modList_local <- ore.pull(modList)
summary(modList_local$BOS) ## return model BOS
```

```
Large data regression model: biglm(ARRDELAY ~ DISTANCE + DEPDELAY, dat)
Sample size =  3928
               Coef    (95%     CI)     SE      p
(Intercept)  0.0638 -0.7418  0.8693 0.4028 0.8742
DISTANCE    -0.0014 -0.0021 -0.0006 0.0004 0.0002
DEPDELAY     1.0552  1.0373  1.0731 0.0090 0.0000
```



ORE: Build and Execute a Model

ORACLE

Build and test models in parallel with ore.indexApply

```
produceModels <- function(models.list, trainData, model.datastore, overwrite=FALSE, parallel = FALSE) {
  # local function that builds model with trainData
  local.build.model <- function (idx, test.models, dat, model.datastore) {
    model.name <- names(test.models)[idx]
    assign(model.name, do.call(test.models[[idx]], list(dat)) )
    ore.save(list = model.name, name = model.datastore, append=TRUE)
    model.name
  }
  # check overwrite
  if (overwrite && nrow(ore.datastore(name=model.datastore)) > 0L)
    ore.delete(name=model.datastore)

  # build models
  trainData <- ore.pull(trainData)
  models.success <- ore.pull(ore.indexApply(length(models.list), local.build.model,
                                            test.models=models.list, dat=trainData,
                                            model.datastore=model.datastore, parallel=parallel,
                                            ore.connect=TRUE))
  as.character(models.success)
}
```

Select best model and save in database ‘datastore’ object (1)

```
selectBestModel <- function(testData, evaluate.func,
                            model.datastore, modelnames.list=character(0),
                            production.datastore=character(0), parallel=FALSE) {
  # get names of models to select from
  modelNames <- ore.datastoreSummary(name = model.datastore)$object.name
  modelNames <- intersect(modelNames, modelnames.list)

  # local function that scores model with test data
  local.model.score <- function(idx, model.names, datastore.name, dat, evaluate) {
    modName <- model.names[idx]
    ore.load(list=modName, name=datastore.name)
    mod <- get(modName)
    predicted <- predict(mod, dat)
    do.call(evaluate, list(modName, dat, predicted))
  }
```

Select best model and save in database ‘datastore’ object (2)

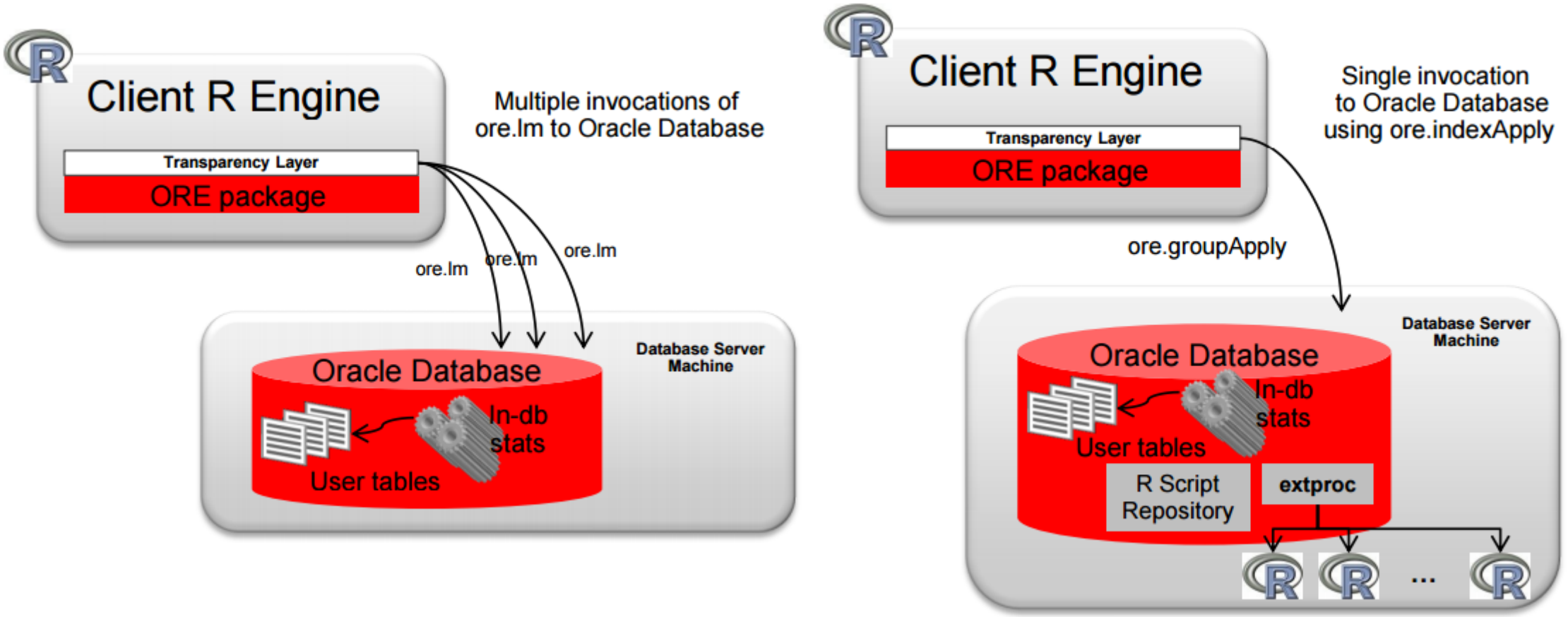
```
  # score these models
  testData <- ore.pull(testData)
  scores <- ore.pull(ore.indexApply(length(modelNames), local.model.score,
                                    model.names=modelNames,
                                    datastore.name=model.datastore, dat=testData,
                                    evaluate=evaluate.func, parallel=parallel,
                                    ore.connect=TRUE))
  # get best model based upon scores
  bestmodel.idx <- order(as.numeric(scores))[1]
  bestmodel.score <- scores[[bestmodel.idx]]
  bestmodel.name <- modelNames[bestmodel.idx]
  ore.load(list=bestmodel.name, name=model.datastore)
  if (length(production.datastore) > 0L)
    ore.save(list=bestmodel.name, name=production.datastore, append=TRUE)
  names(bestmodel.score) <- bestmodel.name
  bestmodel.score
}
```

Generate the Best Model

```
generateBestModel <- function(data, datastore.name, models.list,
                              evaluate.func, parallel=FALSE) {
  data <- sampleData(data)
  trainData <- data$train
  testData  <- data$test
  produceModels(models.list, trainData, model.datastore="ds.tempModelset",
                overwrite=TRUE, parallel=parallel)
  bestModelName <- names(selectBestModel(testData, evaluate.func,
                         model.datastore="ds.tempModelset",
                         production.datastore=datastore.name, parallel=parallel))
  bestModelName
}
```

Bagging Execution Model

2 options: Client-Controlled or Database-Controlled



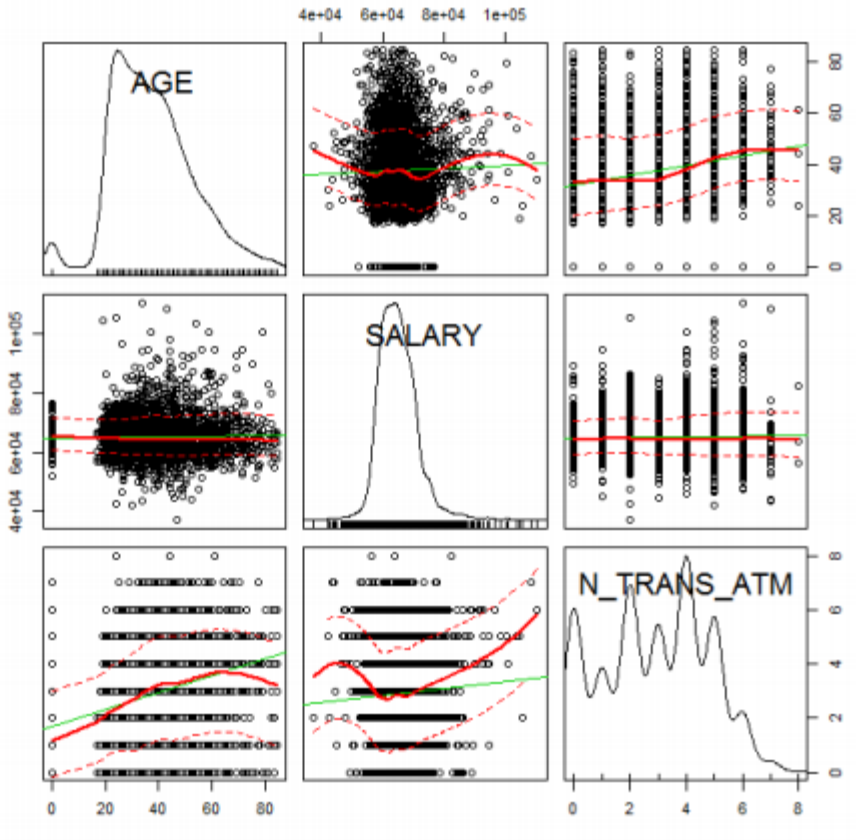
ORE EDA: Data Exploration

ORACLE

Data Exploration

```
library(car)
LTV <- CUSTOMER_LTV
row.names(LTV) <- LTV$CUST_ID
summary(LTV[,c("CUST_ID","AGE","SALARY",
               "MARITAL_STATUS","N_TRANS_ATM","LTV")])
ltv <- ore.pull(LTV)
ltv.sample <- ltv[sample(1:nrow(ltv),4000),]
scatterplotMatrix(~AGE+SALARY+N_TRANS_ATM,
                  data=ltv.sample)
```

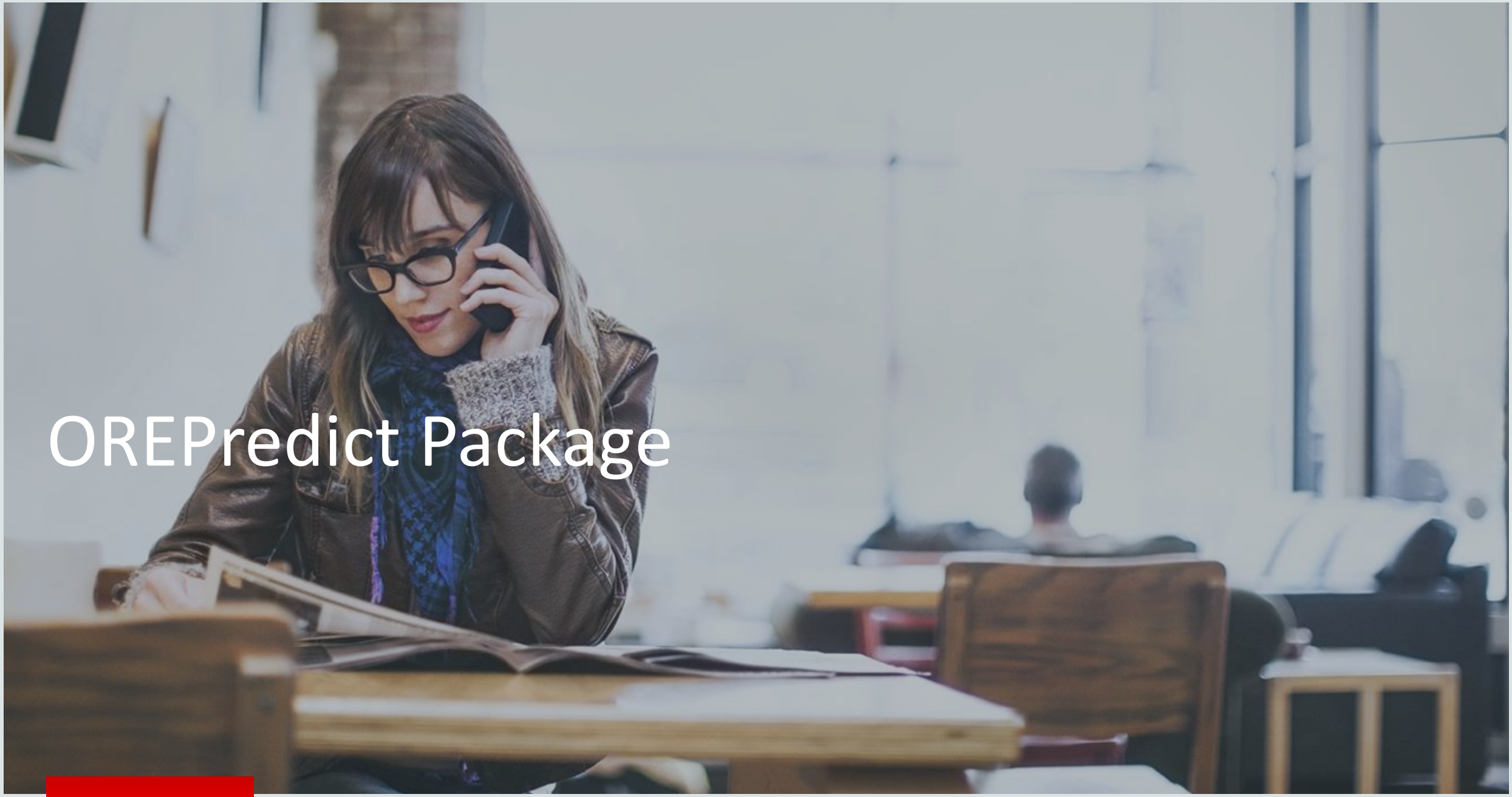
```
> summary(LTV[,c("CUST_ID","AGE","SALARY","MARITAL_STATUS",
+                "N_TRANS_ATM","LTV")])
    CUST_ID          AGE              SALARY        MARITAL_STATUS  N_TRANS_ATM        LTV
 CU1    :    1   Min.   : 0.0   Min.   : 37107   SINGLE  :5359   Min.   :0.00   Min.   :    0
 CU10   :    1   1st Qu.:27.0   1st Qu.: 60730   MARRIED :5284   1st Qu.:1.00   1st Qu.:18657
 CU100  :    1   Median :36.0   Median : 64452   DIVORCED:3901   Median :3.00   Median :23054
 CU1000 :    1   Mean   :37.4   Mean   : 65132   WIDOWED : 669   Mean   :2.88   Mean   :22267
 CU10000:    1   3rd Qu.:46.0   3rd Qu.: 68496   OTHER   : 129   3rd Qu.:4.00   3rd Qu.:26104
 CU10001:    1   Max.   :84.0   Max.   :111605                   Max.   :8.00   Max.   :47502
 (Other):15336
```



Exploratory Data Analysis Functions

ORE Function	Description
`ore.corr`	Used to perform correlation analysis across numeric columns.
`ore.crosstab`	Used to build cross-tabulations; supports multiple columns. Also allows optional aggregations, weighting, and ordering options.
`ore.esm`	Builds an exponential smoothing model on data in an ordered `ore.vector` function.
`ore.freq`	Using the output of the `ore.crosstab` function, it determines whether two-way cross-tabulation or N-way cross-tabulation tables should be used for the results.
`ore.rank`	Allows you to investigate the distribution of values along numeric columns.
`ore.sort`	Allows you to sort the data in a variety of ways.
`ore.summary`	Provides a series of descriptive analytics based on the data in an ORE data frame.
`ore.univariate`	Provides distribution analysis of numeric columns in an ORE data frame. Gives all the statistics from an `ore.summary` function plus signed rank test and extreme values.

OREPredict Package



Supported Algorithms

Class	Package	Description
glm	stats	Generalized Linear Model
negbin	MASS	Negative binomial Generalized Linear Model
hclust	stats	Hierarchical Clustering
kmeans	stats	K-Means Clustering
lm	stats	Linear Model
multinom	nnet	Multinomial Log-Linear Model
nnet	nnet	Neural Network
rpart	rpart	Recursive Partitioning and Regression Tree

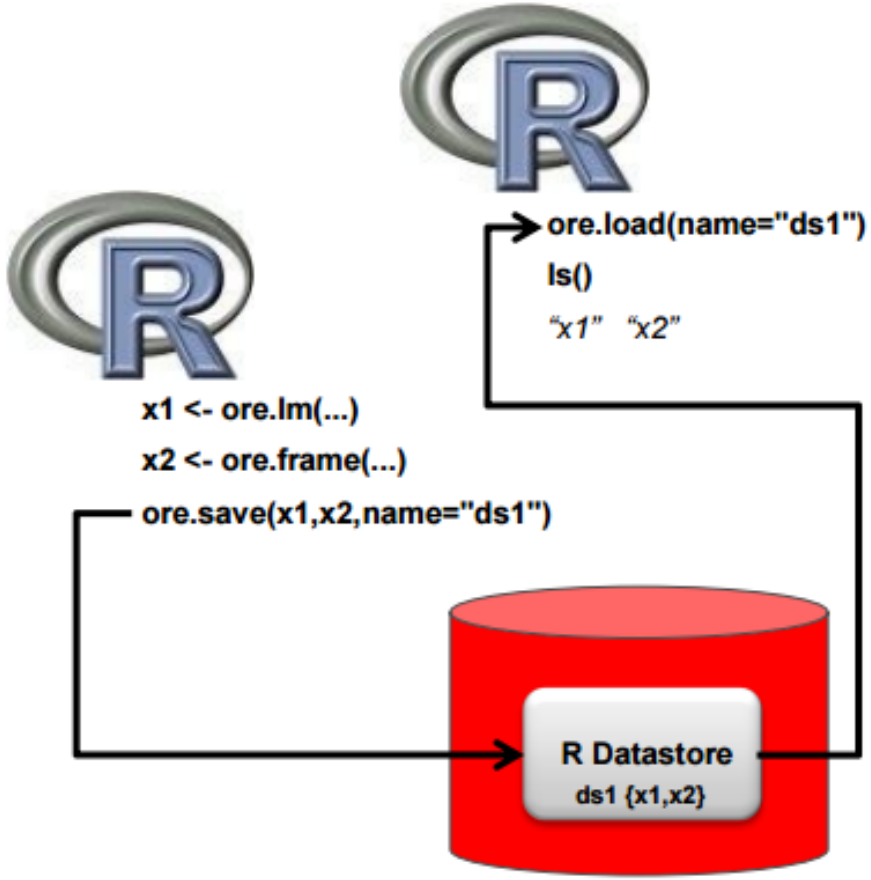
ORE: Data Manipulation

R Objects Persistence

- ore.save()
- ore.load()

- Provide database storage to save/restore R and ORE objects across R sessions

- Use cases include
  - Enable passing of predictive model for embedded R execution, instead of recreating them inside the R functions
  - Passing arguments to R functions with embedded R execution
  - Preserve ORE objects across R sessions



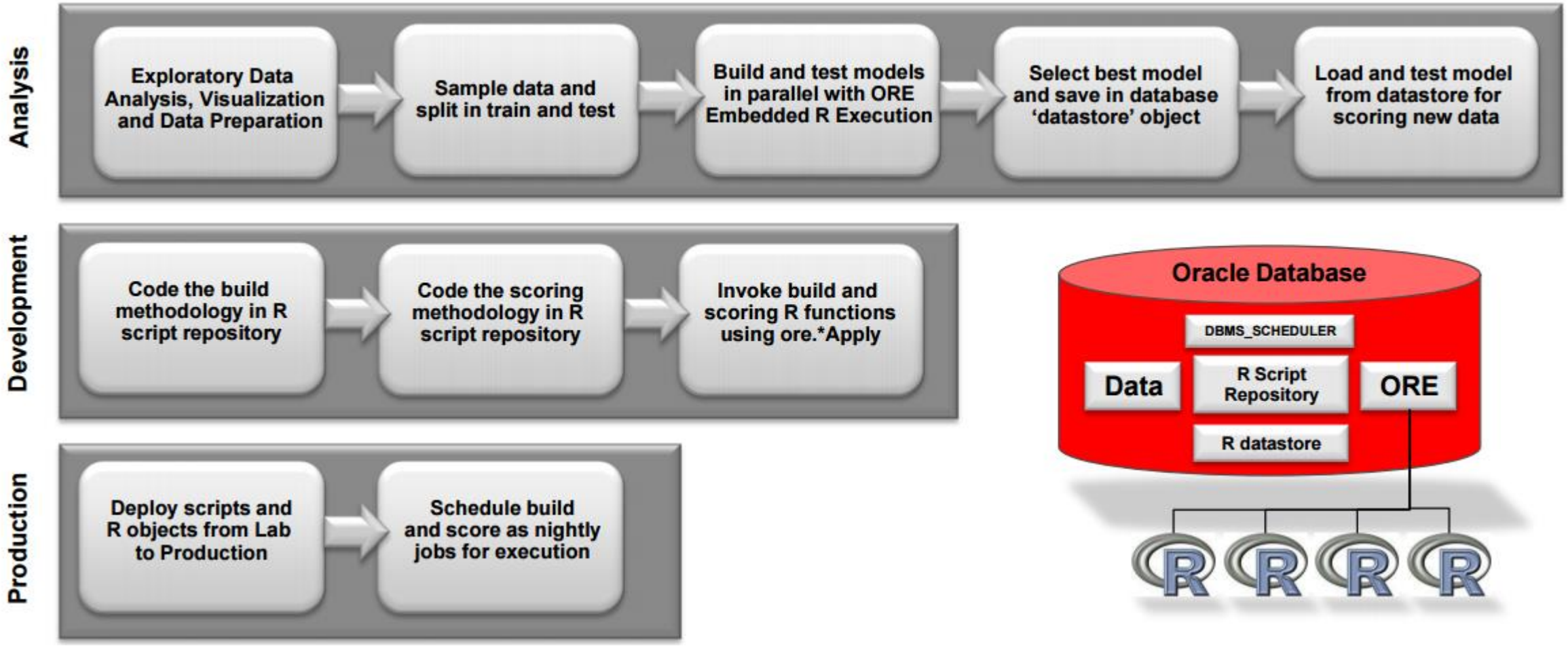
Functions for using the In-database ORE DataStore

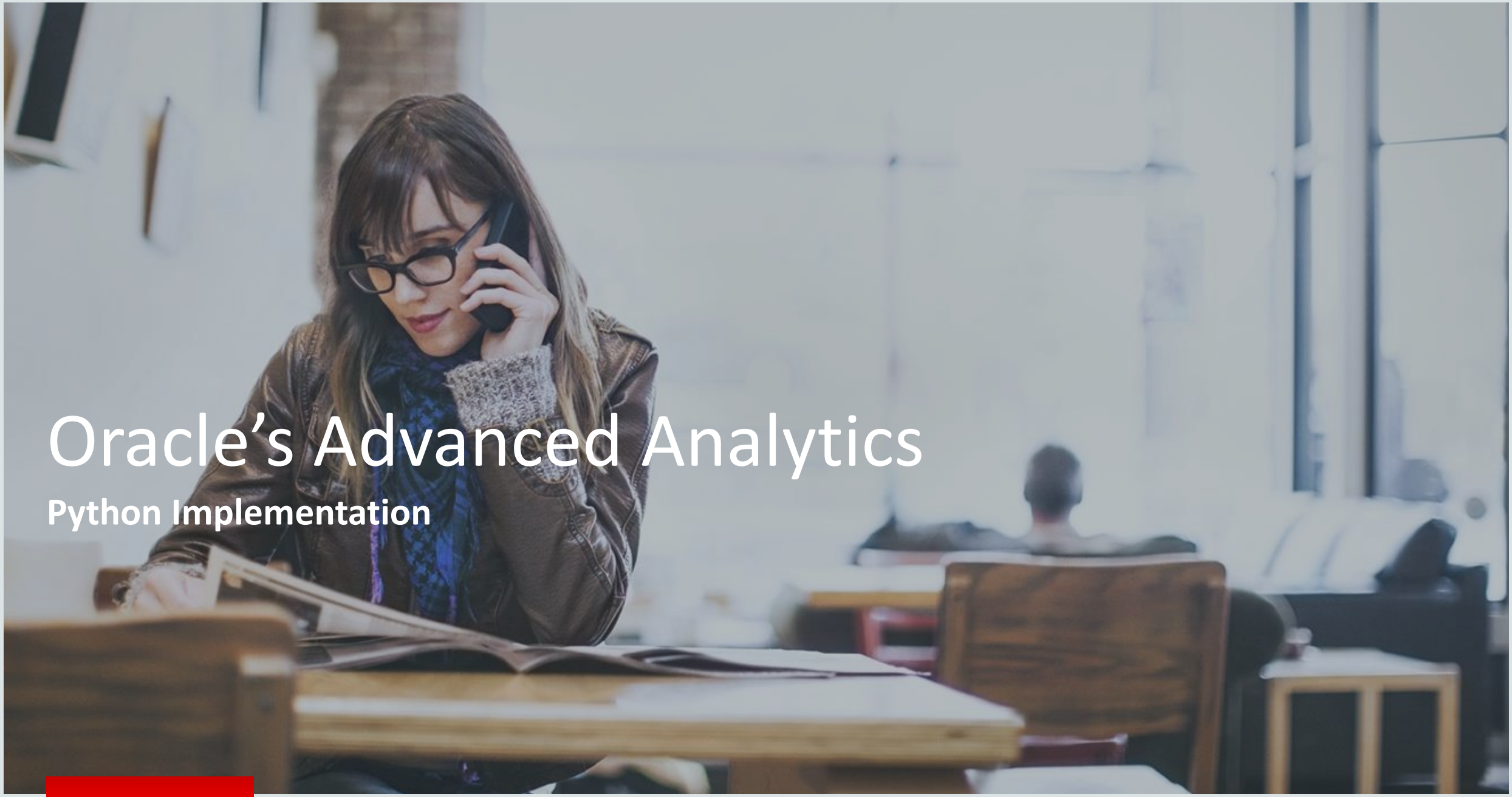
Function	Description
`ore.datastore`	Lists information about the ORE datastore that exists in your schema. This defaults to the current schema. When a name is provided, this function returns the details of that ORE datastore. The details displayed for each ORE datastore include the name, the number of objects, the size of the ORE datastore, the date created, and any description that has been assigned to the ORE datastore.
`ore.datastoreSummary`	Provides detailed information about the objects in a specified ORE datastore.
`ore.delete`	Deletes the ORE datastore from the database. This will also remove all the ORE objects contained in the ORE datastore.
`ore.grant`	Grants access to the ORE datastore to other users in the database if the ORE datastore had been created as grantable.
`ore.load`	Loads the objects from the ORE datastore back into the R environment. These objects are available for use immediately. Either all the objects can be retrieved or only the objects listed.
`ore.lazyLoad`	Loads the objects from the ORE datastore when they are first used by your R code. Objects are only restored one at a time.
`ore.revoke`	Revokes access that a schema has to an ORE datastore.
`ore.save`	Creates an ORE datastore and stores the listed objects in the ORE datastore for future use.

```
select object_name, object_type,
last_ddl_time
from user_objects
where object_name like 'ORE$%';
```

Oracle R Enterprise as Framework for Advanced Analytics

Workflow Example





Oracle's Advanced Analytics

Python Implementation

ORACLE

Oracle's Advanced Analytics

Multiple interfaces across platforms — SQL, R, Python, GUI, Dashboards, Apps

Users

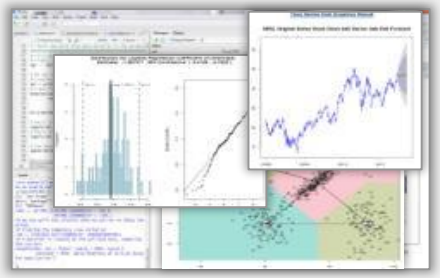
R & Python programmers

Data / Business Analysts

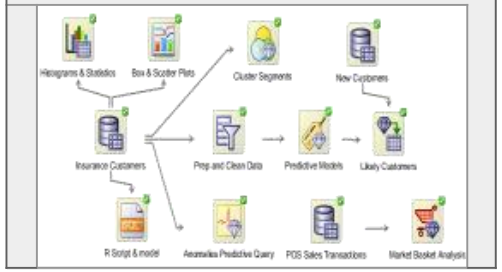
Business Analysts/Mgrs

Domain End Users

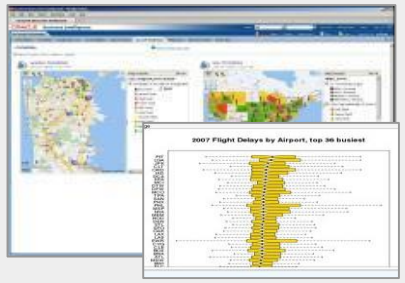
R & Python Clients



SQL Developer/ Oracle Data Miner



OBIEE



Applications

Platform

hadoop

ORAAH

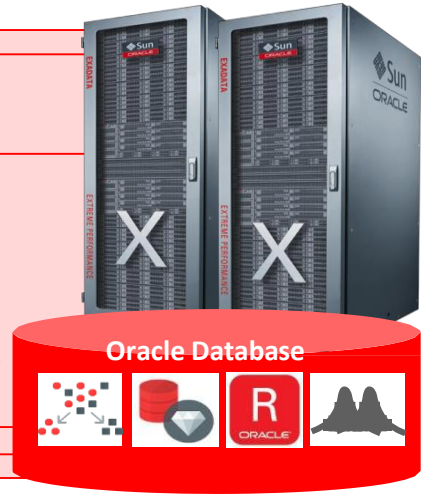
Parallel, distributed algorithms

Oracle Database Enterprise Edition

Oracle Advanced Analytics - Database Option

SQL, R & Python Integration

for Scalable, Distributed, Parallel in-Database ML Execution



Oracle Cloud

Python IDEs

Oracle Advanced Analytics differentiators

Work directly with data in Database and Hadoop

- Eliminate need to request extracts from IT/DBA – immediate access to database and Hadoop data
- Process data where they reside – minimize or eliminate data movement

Scalability and Performance

- Use parallel, distributed algorithms that scale to big data on Oracle Database
- Leverage powerful engineered systems to build models on billions of rows of data or millions of models in parallel

Ease of deployment

- Using Oracle Database, place **R, Python and SQL scripts** immediately in production (no need to recode)
- Use production quality infrastructure without custom plumbing or extra complexity

Process support

- Maintain and ensure data security, backup, and recovery using existing processes
- Store, access, manage, and track analytics objects (models, scripts, workflows, data) in Oracle Database

Oracle's Python Technologies

Supporting Oracle Database

- cx_Oracle:
  - https://github.com/oracle/python-cx_Oracle

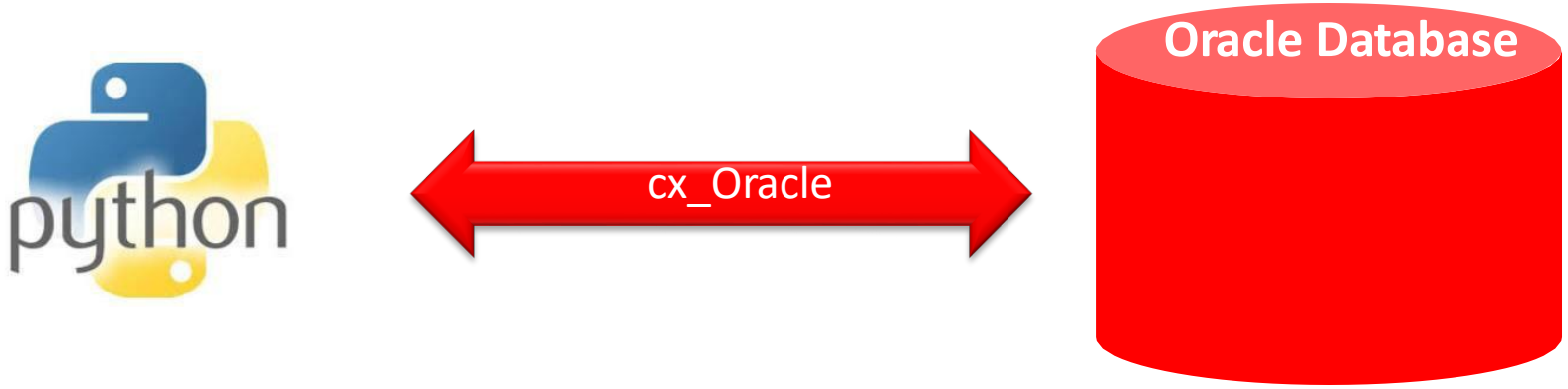
- Oracle Machine Learning for Python

  Component of the Oracle Advanced Analytics Option to Oracle Database

cx_Oracle Package

cx_Oracle

- Python package enabling scalable and performant connectivity to Oracle Database
  - Open source, publicly available on CRAN
  - Oracle is maintainer
- Oracle Database Interface for Python confirming to Python DB API 2.0 specification
  - Optimized driver based on OCI
  - Execute SQL statements from Python interface
  - Enables transactional behavior for insert, update, and delete



cx_Oracle - Requirements

- Easily installed from PyPI
- Support for Python 2 and 3
- Support for Oracle Client 11.2, 12.1, 12.2, 18
  - Oracle's standard cross-version interoperability, allows easy upgrades and connectivity to different Oracle Database versions
- Connect to Oracle Database 9.2, 10, 11, 12, 18
  - (Depending on the Oracle Client version used)
- SQL and PL/SQL Execution
  - Underlying Oracle Client libraries have optimizations: compressed fetch, pre-fetching, client and server result set caching, and statement caching with auto-tuning

cx_Oracle Example

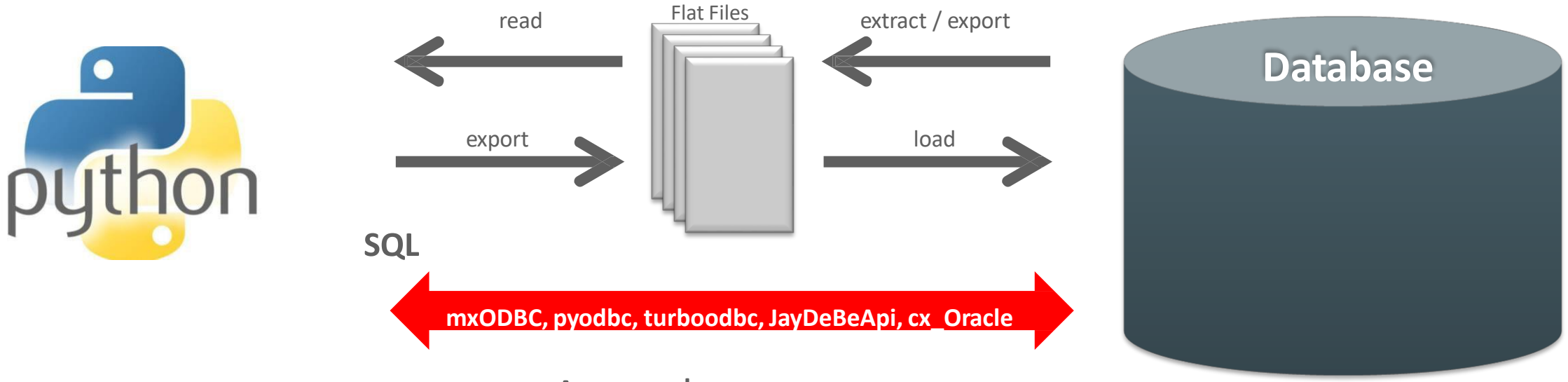
```
import cx_Oracle
con = cx_Oracle.connect('pythonhol/welcome@127.0.0.1/orcl')
print con.version
con.close()



con = cx_Oracle.connect('pythonhol', 'welcome',  '127.0.0.1:/orcl:pooled',
                        cclass = "HOL", purity = cx_Oracle.ATTR_PURITY_SELF)
print con.version
con.close()
```

cx_Oracle Example

```
import cx_Oracle
con = cx_Oracle.connect('pythonhol/welcome@127.0.0.1/orcl')
cur = con.cursor() # opens cursor for statements to use
cur.execute('select * from departmentsorder  by department_id')
for result in cur:                  # prints out all data
    print result
#or
row = cur.fetchone()                # return a single row as tuple and advance row
print row
row = cur.fetchone()
print row
#or
res = cur.fetchmany(numRows=3)  # returns list of tuples
print res
cur.close()
con.close()
```

Oracle Machine Learning for Python

Traditional Python and Database Interaction



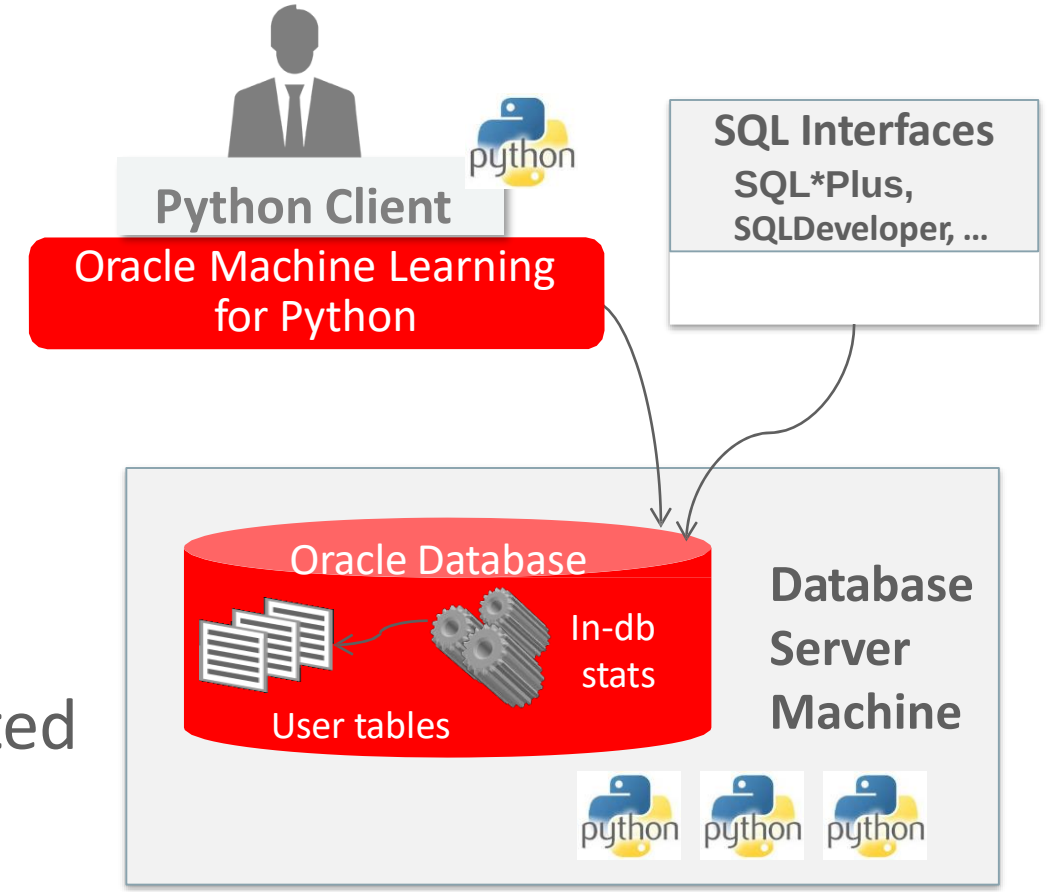
Python script cron job

- Access latency
- Paradigm shift: Python → SQL → Python
- Memory limitation – data size
- Single threaded
- Ad hoc production deployment
- Issues for backup, recovery, security

Oracle Machine Learning for Python

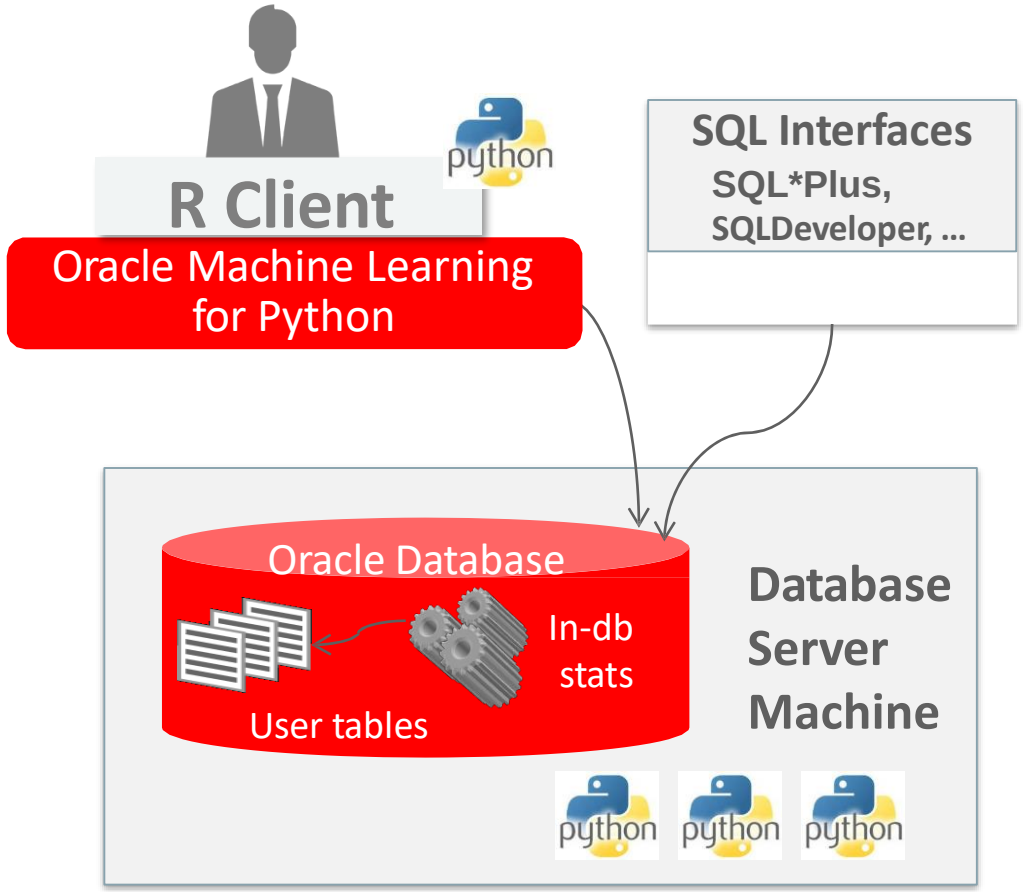
Oracle Advanced Analytics Option to Oracle Database >= 18c

- Use Oracle Database as HPC environment
- Use in-database parallel and distributed machine learning algorithms
- Manage Python scripts and Python objects in Oracle Database
- Integrate Python results into applications and dashboards via SQL
- Produce better models faster with automated machine learning



OAA / OML for Python

- Transparency layer
  - Leverage proxy objects so data remains in database
  - Overload Python functions translating functionality to SQL
  - Use familiar Python syntax to manipulate database data
- Parallel, distributed algorithms
  - Scalability and performance
  - Exposes in-database algorithms from Oracle Data Mining
- Embedded Python execution
  - Manage and invoke Python scripts in Oracle Database
  - Data-parallel, task-parallel, and non-parallel execution
  - Use open source Python packages
- Automated machine learning
  - Feature selection, model selection, hyperparameter tuning



OML4Py Transparency Layer

- Leverages proxy objects for database data: *oml.DataFrame*

```
# DB table Boston
DATA = core.sync(table = 'BOSTON')

# Pandas DataFrame data
DATA = core.create(data, table = 'BOSTON')
```

- Overloads Python functions translating functionality to SQL
- Use familiar Python syntax to manipulate database data

```
DATA.shape

DATA.head()

DATA.describe()

DATA.std()

DATA.skew()


train_dat, test_dat =
   DATA.split()

train_dat.shape

test_dat.shape
```

OAA / OML for Python 1.0

Machine Learning algorithms in-Database

…plus open source Python packages for algorithms in combination with embedded Python execution

Classification

- Decision Tree
- Naïve Bayes
- Generalized Linear Model
- Support Vector Machine
- RandomForest
- Neural Network

Regression

- Generalized Linear Model
- Neural Network
- Support Vector Machine

Clustering

- Expectation Maximization
- Hierarchical k-Means

Attribute Importance

- Minimum Description Length

Anomaly Detection

- 1 Class Support Vector Machine

Market Basket Analysis

- Apriori – Association Rules

Feature Extraction

- Singular Value Decomposition
- Explicit Semantic Analysis

Time Series

- Exponential Smoothing

Supports integrated partitioned models, text mining

Connect to the database

```
import oml
import os
sid = os.environ["ORACLE_SID"]
oml.connect(user="pyquser", password="pyquser",
  dsn="(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)
  (PORT=1521))(CONNECT_DATA=(SID="+sid+")))")
oml.isconnected()
```

```
>>> sid = os.environ["ORACLE_SID"]
>>> oml.connect(user="pyquser", password="pyquser",
...             dsn="(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)
...(PORT=1521))(CONNECT_DATA=(SID="+sid+")))")
>>> oml.isconnected()
True
```

Invoke in-database aggregation function

```
ONTIME_S = oml.sync(table="ONTIME_S")
res = ONTIME_S.crosstab('DEST')
type(res)
res.head()
```

```
>>> ONTIME_S = oml.sync(table="ONTIME_S")
>>> res=ONTIME_S.crosstab('DEST')
>>> type(res)
<class 'oml.core.frame.DataFrame'>
>>> res.head()
  DEST  count
0  ABE    237
1  ABI     34
2  ABQ   1357
3  ABY     10
4  ACK      3
```

Source data is a DataFrame, ONTIME_S, which resides as an Oracle Database table

The crosstab() function has been overloaded to accept OML DataFrame objects and transparently generates SQL for execution in Oracle Database

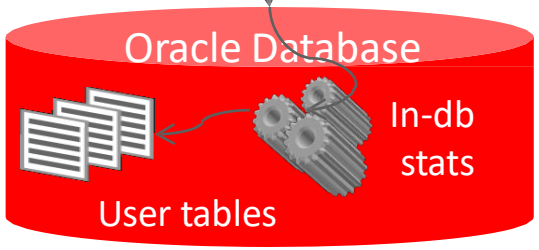
Python user on desktop

Client Python Engine

Transparency Layer

Oracle ML for Python

```
select DEST, count(*)
from ONTIME_S
group by DEST
```



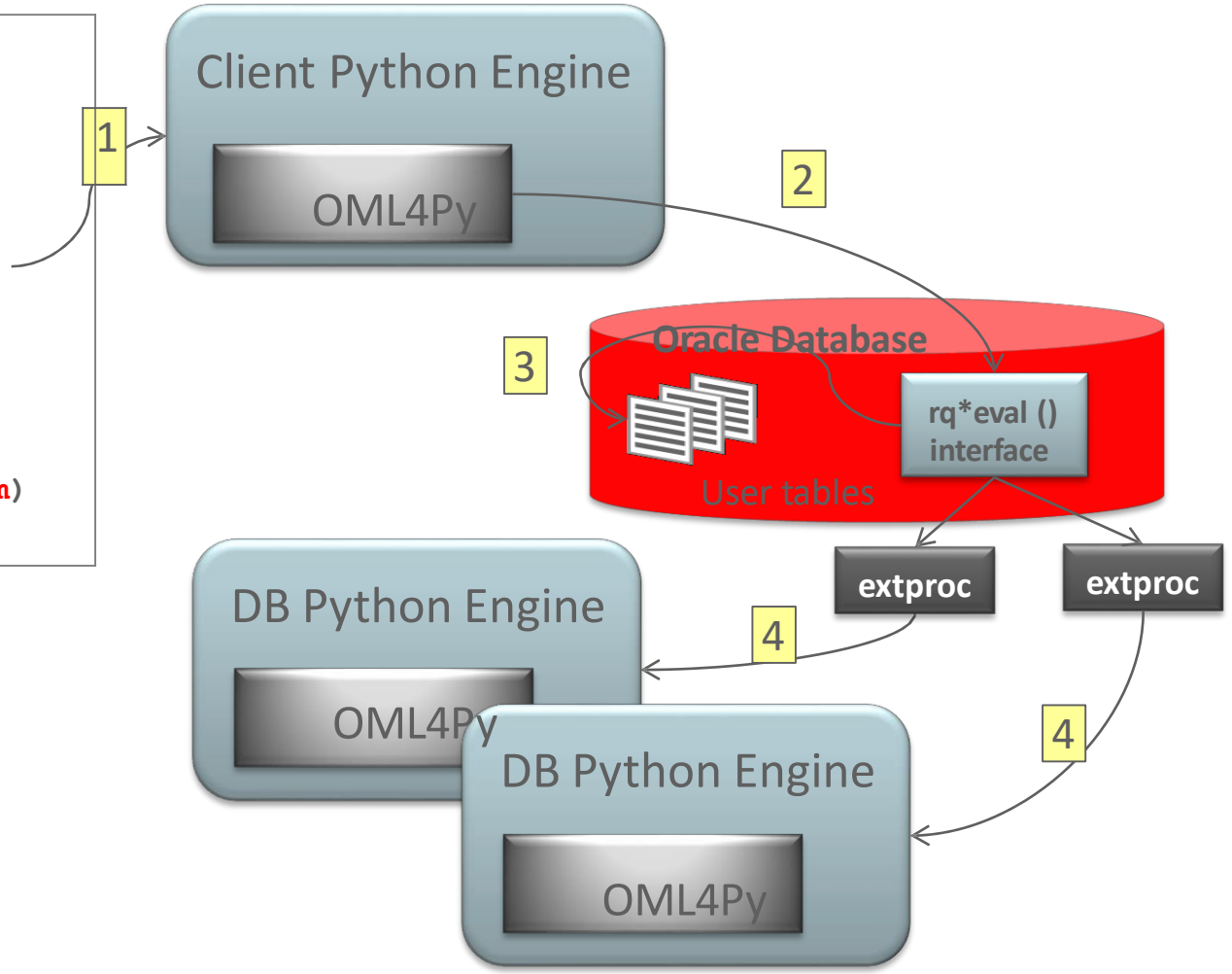
OML4Py Embedded Python

```
def fit(data):
  from sklearn.svm import LinearSVC
  x = data.drop('TARGET',
                axis = 1).values
  y = data['TARGET']
  return LinearSVC().fit(x, y)
```

```
oml.script.create(
           "sk_svc_fit", fit,
           overwrite = True)


oml.script.dir()


clf_mod =
oml.table_apply(train_dat,
     func = 'sk_svc_fit',
     oml_input_type  =
            'pandas.DataFrame')
```

oml.group_apply – partitioned data flow

```
def build_lm(dat):
  from sklearn import linear_model
  lm = linear_model.LinearRegression()
  X = dat[["PETAL_WIDTH"]]
  y = dat[["PETAL_LENGTH"]]
  lm.fit(X, y)
  return lm

index = oml.DataFrame(IRIS['SPECIES'])

mod = oml.group_apply(IRIS[:,["PETAL_LENGTH",
          "PETAL_WIDTH","SPECIES"]], index, func=build_lm)
          sorted(mod.pull().items())
```



Also includes

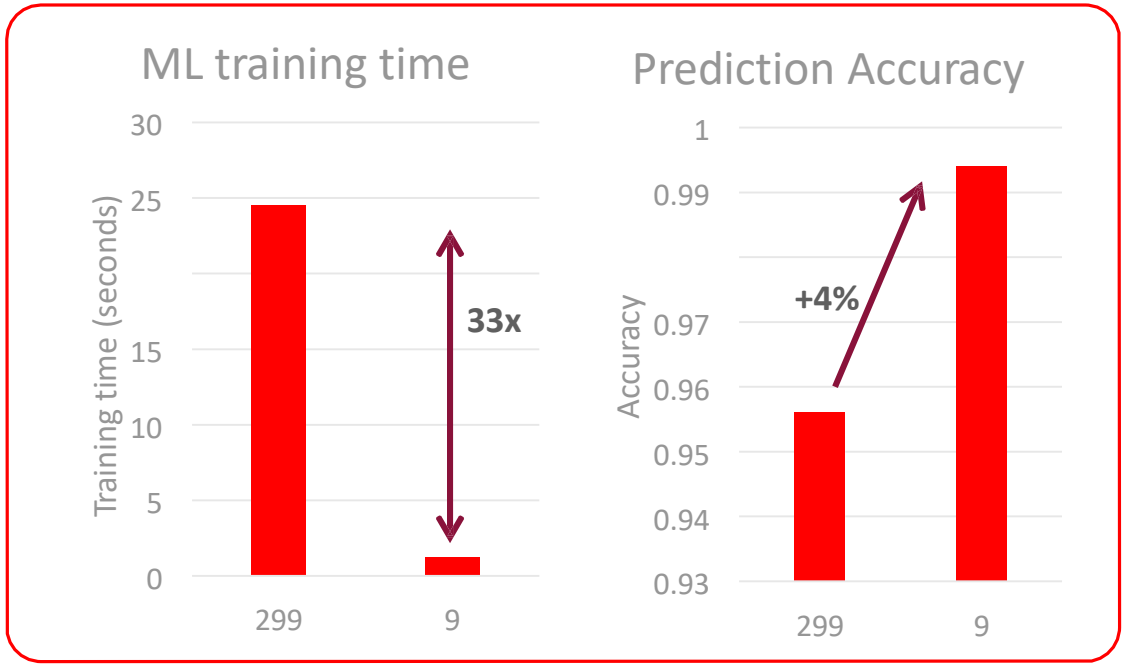
- ore.doEval
- ore.tableApply
- ore.rowApply
- ore.indexApply

AutoML – *new* with OML4Py in Oracle Advanced Analytics

- Goals: increase model quality and data scientist productivity while reducing overall compute time
- Auto Feature Selection
  - Reduce # features by identifying most relevant
  - Improve performance and accuracy
- Auto Model Selection for classification and regression
  - Identify best algorithm to achieve maximum score
  - Find best model many times faster than with exhaustive search techniques
- Auto Tuning of Hyperparameters
  - Significantly improve model accuracy
  - Avoid manual or exhaustive search techniques

Auto Feature Selection: Motivation & Example

- Many real-world datasets have a large number of irrelevant features
- Slows down training
- Goal: Speed-up ML pipeline by selecting most relevant features



OpenML dataset 312, 1925 rows

OpenML dataset 40996 (56000 rows, 784 columns)
Using SVM Gaussian with AutoFS

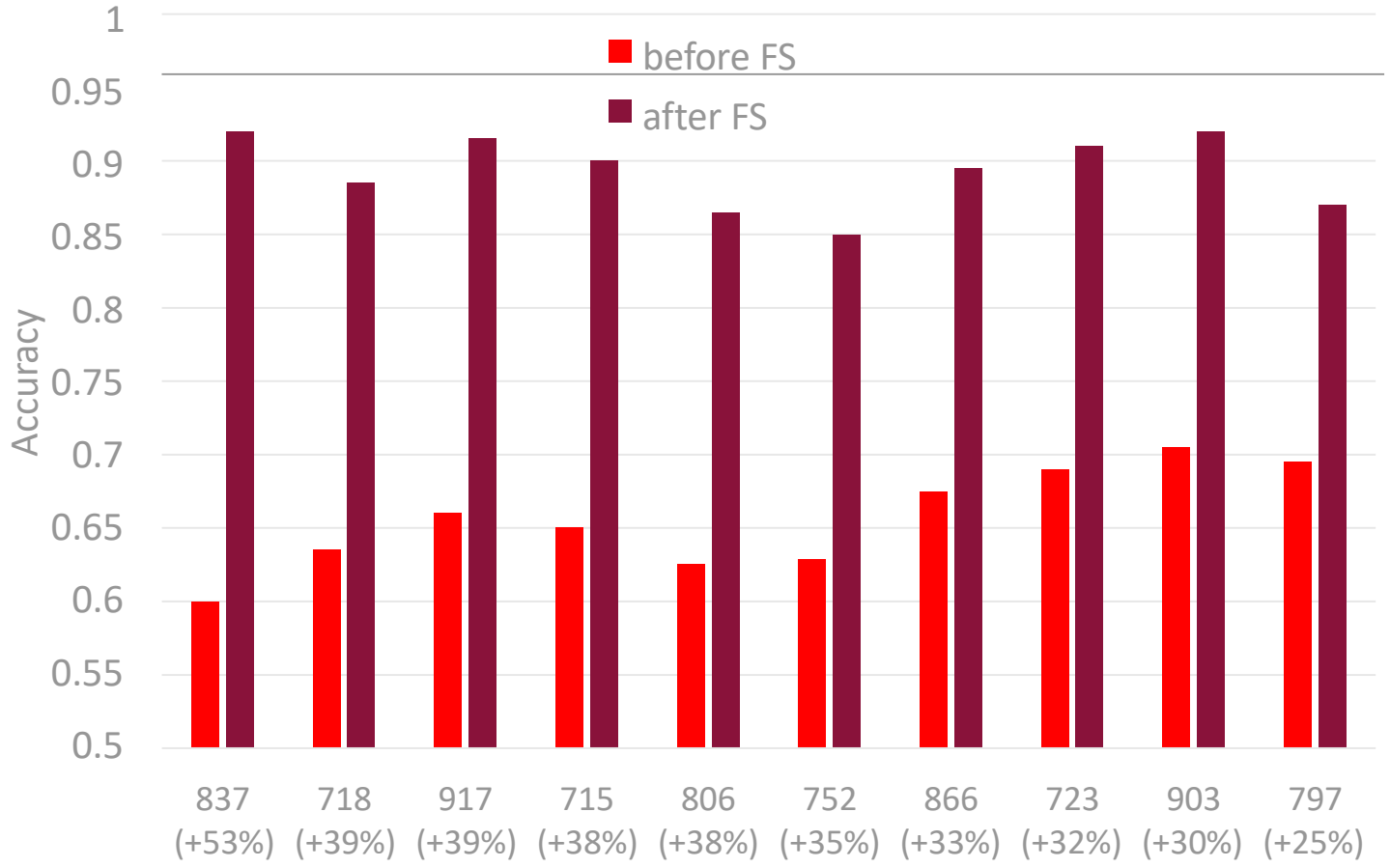
- Accuracy improves from 0.659 to 0.843
- Features reduced from 784 to 309
- Training time reduced 1.3x

AutoFS: Evaluation for OAA SVM Gaussian

- 150 Datasets with more than 500 cases

Avg Feature Reduction
52%

Avg Accuracy Gain
2.5%



AutoFS Example

```
fs = FeatureSelection(
          mining_function = 'classification',
          score_metric = 'accuracy')

selected_features =
     fs.reduce('dt', X_train, y_train)

X_train = X_train[:,selected_features]
```

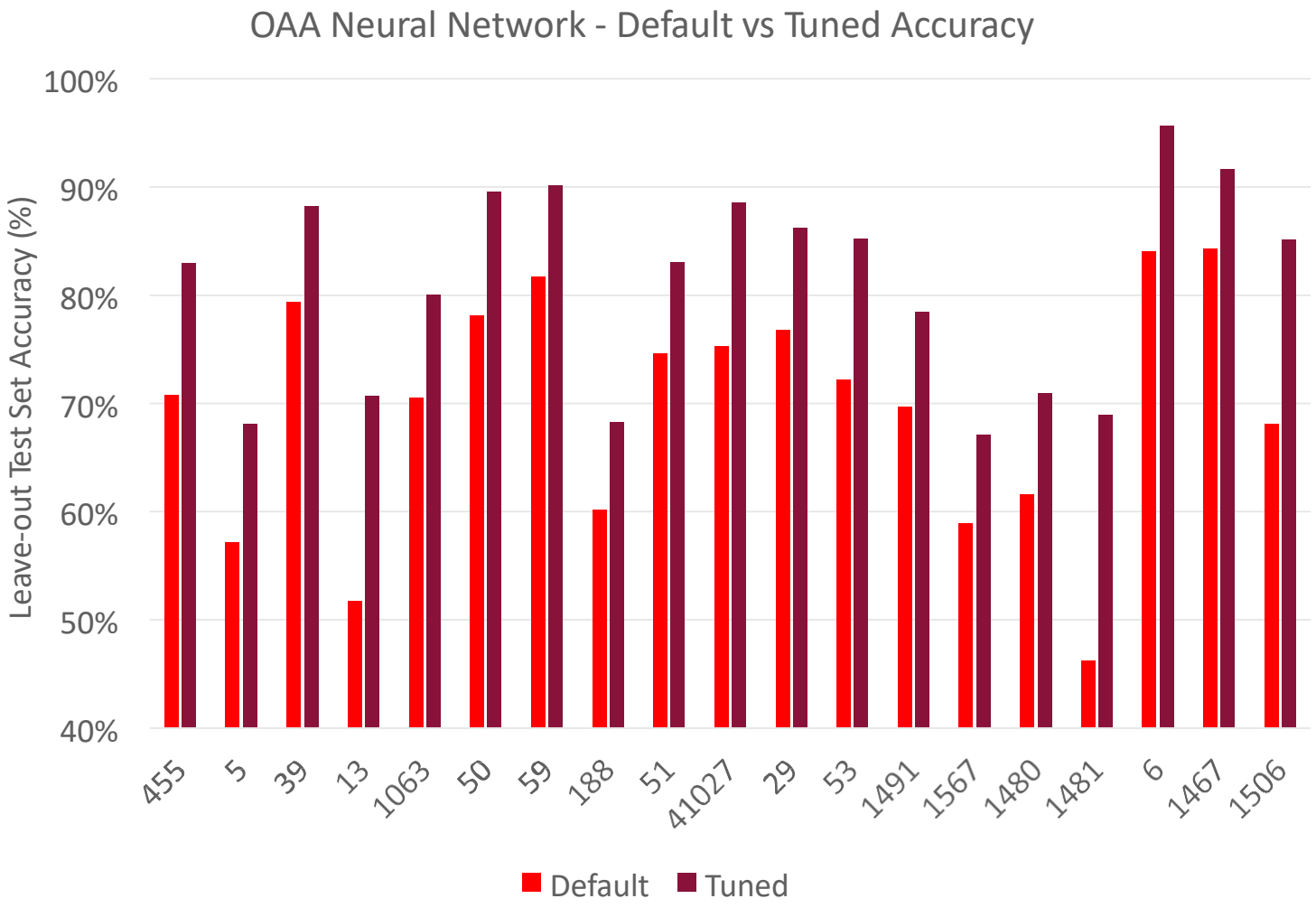
AutoMS: Example

```
ms = ModelSelection(
            mining_function = 'classification',
            score_metric = 'accuracy')

best_model = ms.select(X_train, y_train)

y_pred = best_model.predict(X_test)
```

AutoTune: Evaluation for OAA Neural Network

Avg Improvement of 2.86%

8-22% improvement for several datasets



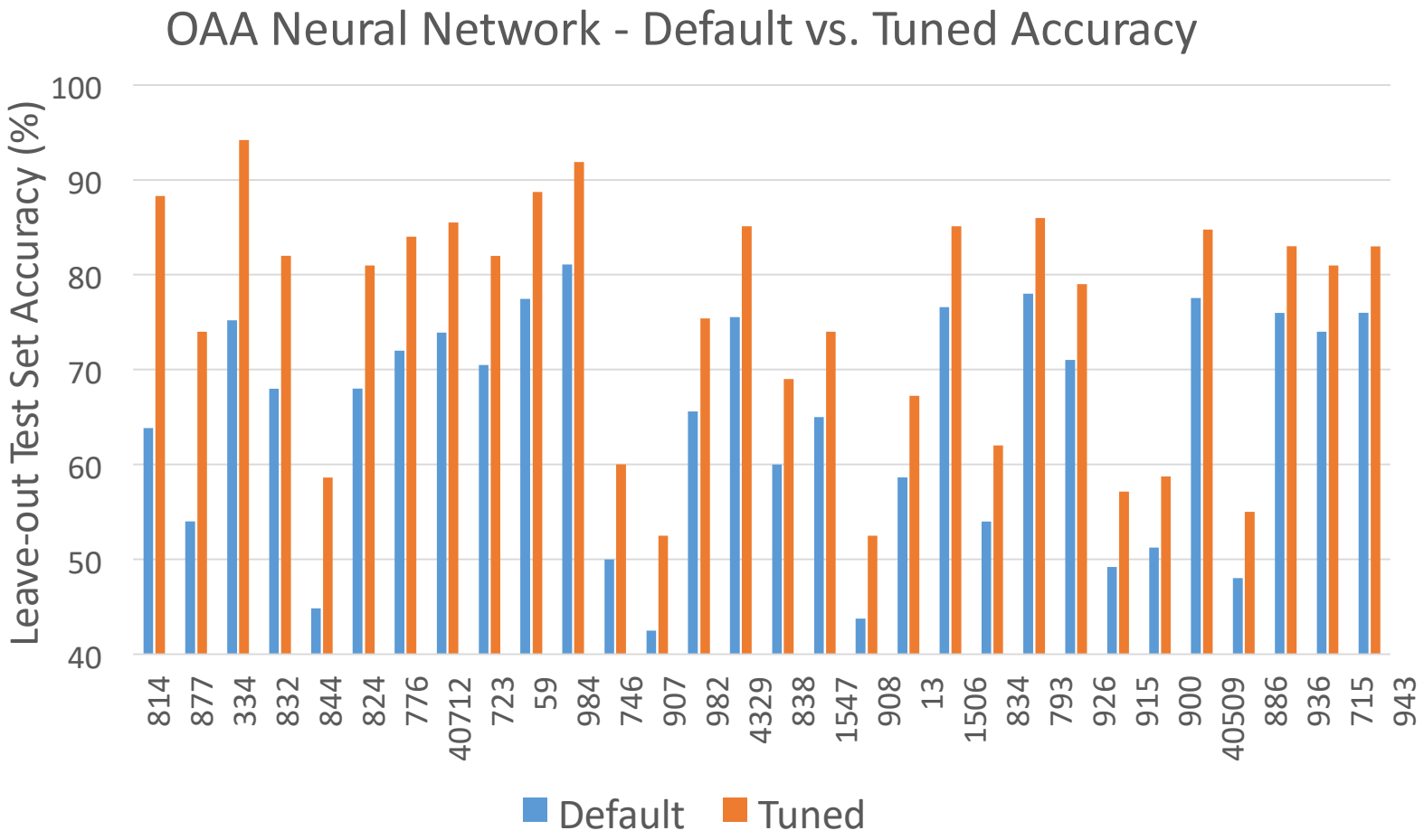
AutoTune Example

```
at = Autotune(
          mining_function = 'classification',
          score_metric = 'accuracy')

evals = at.run('dt', X_train, y_train)

mod = evals['best_model']

y_pred = mod.predict(X_test)
```

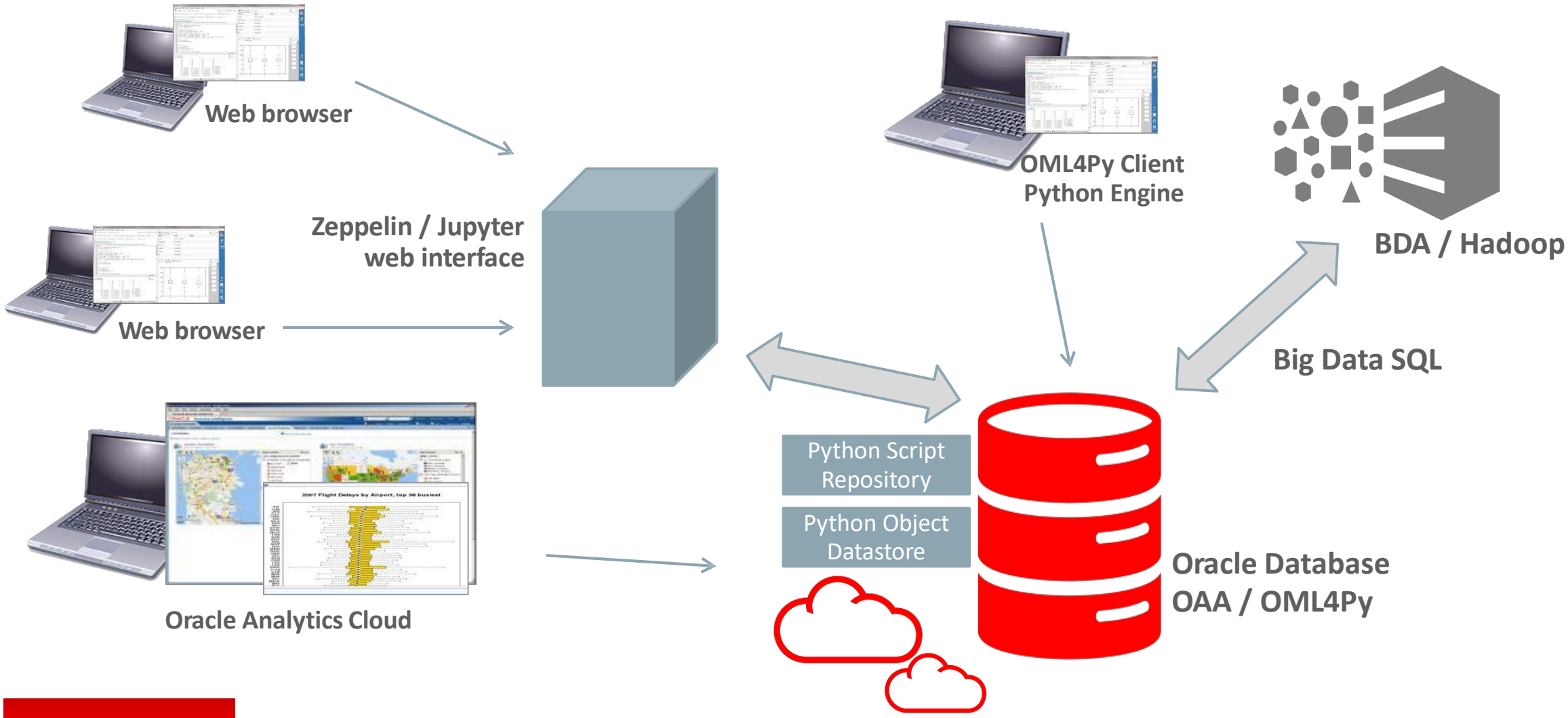
AutoTune: Evaluation for OAA Neural Network

Avg Improvement of 1.68% across ~300 datasets

8-24% improvement for several datasets



OML for Python - Deployment Architecture



Summary - Oracle Machine Learning for Python

- Oracle Database enabled with Python scripting language and environment for the enterprise via Oracle Advanced Analytics option
- Oracle's Python technologies extend Python for enterprise use
  - Data analysis, exploration, and machine learning
  - Streamlined production development
  - Automate key data science steps for greater data scientist productivity, while enhancing accuracy and performance
- Enables performance and scalability leveraging Oracle Database as a high performance compute engine

Hardware and Software
Engineered to Work Together


ORACLE®