ORACLE®

Oracle R Technologies Overview

Massive Predictive Modeling with Oracle R Enterprise

Mark Hornick, Director, Oracle Advanced Analytics

CERN openlab

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 remain at the sole discretion of Oracle.

Agenda

- Overview of R
- Oracle’s R Technologies
  - Oracle R Distribution
  - ROracle
  - Oracle R Enterprise (Oracle Advanced Analytics)
  - Oracle R Advanced Analytics for Hadoop
- Use cases
  - Massive Predictive and Clustering Modeling
  - Face Recognition
  - Densifying Sparse Text via Hadoop
  - Simulations
- Demonstration of Oracle R Enterprise

What is R?

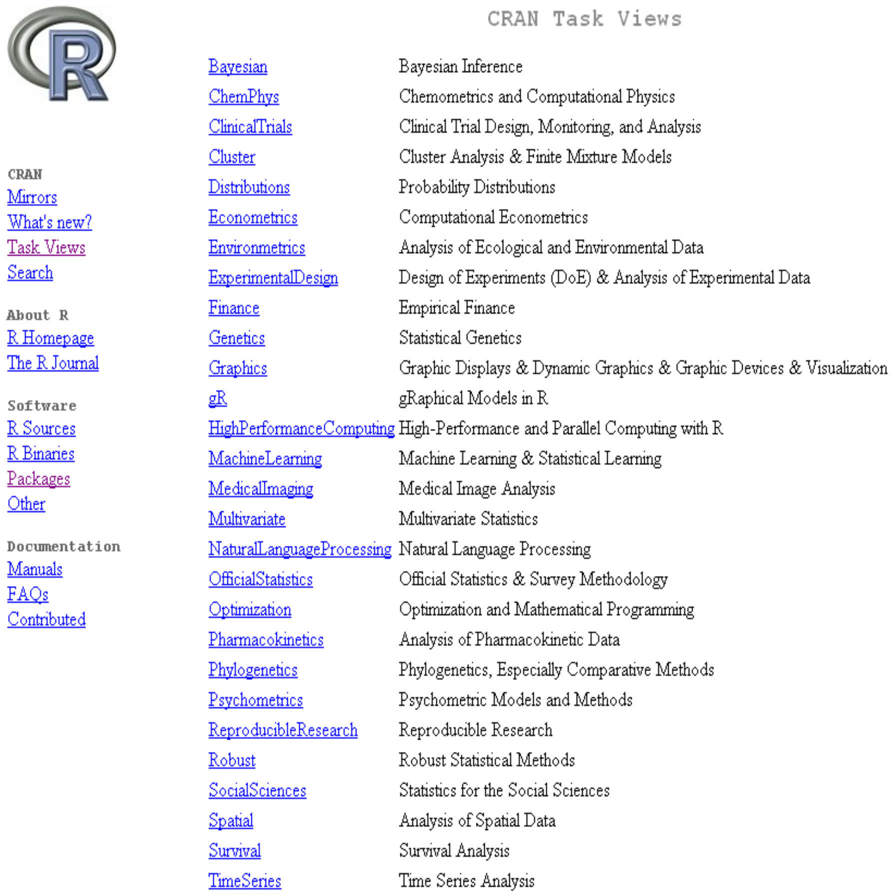
What is R?

- **R is an Open Source language and environment for statistical computing and graphics** **http://www.R-project.org/**

- **Started in 1994 as an alternative to SAS, SPSS, and other proprietary statistical environments**

- **An integrated suite of software facilities for data manipulation, analytical calculations, and graphics**
- **Over 2 million R users worldwide**
  - Widely taught in universities
  - Many corporate analysts know and use R

- **A thriving ecosystem with thousands of open sources packages**



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

CRAN Task Views

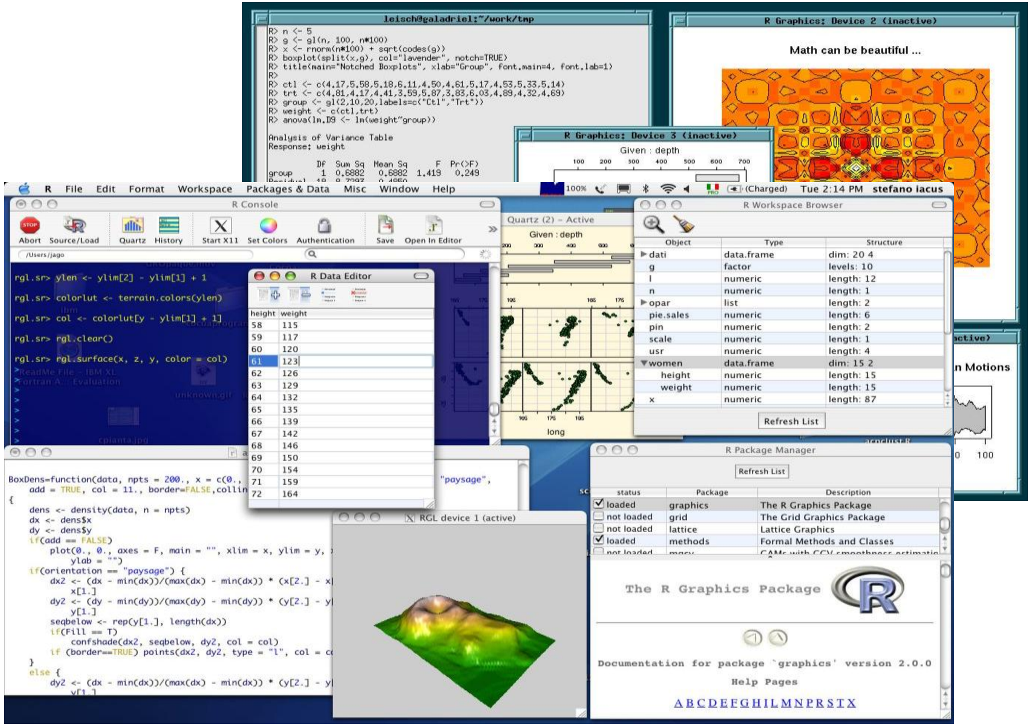
Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
gR	gRaphical Models in R
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis

Why statisticians/data analysts use R

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

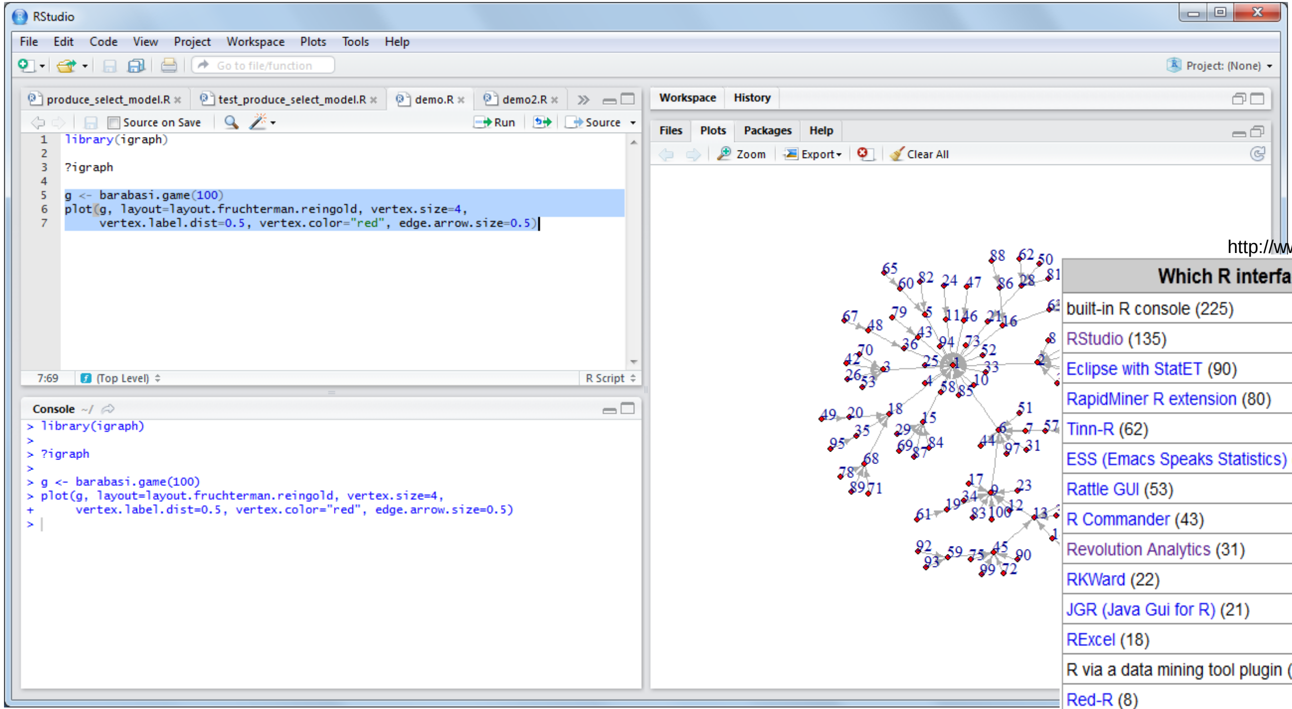
R environment is ..

- Powerful
- Extensible
- Graphical
- Extensive statistics
- OOTB functionality with many ‘knobs’ but smart defaults
- Ease of installation and use
- ***Free***



Third Party Open Source IDEs, e.g., RStudio

Oracle R Enterprise is compatible with Third Party tools



http://www.kdnuggets.com/polls/2011/r-gui-used.htm

Which R interfaces do you use frequently?	
built-in R console (225)	40%
RStudio (135)	24%
Eclipse with StatET (90)	16%
RapidMiner R extension (80)	14.2%
Tinn-R (62)	11%
ESS (Emacs Speaks Statistics) (59)	10.5%
Rattle GUI (53)	9.4%
R Commander (43)	7.7%
Revolution Analytics (31)	5.5%
RKWard (22)	3.9%
JGR (Java Gui for R) (21)	3.7%
RExcel (18)	3.2%
R via a data mining tool plugin (12)	2.1%
Red-R (8)	1.4%
SciViews-R (6)	1.1%
Other (44)	7.8%

ORACLE

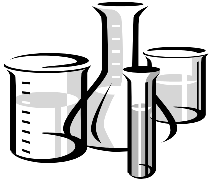
Three Concerns for Enterprise Data

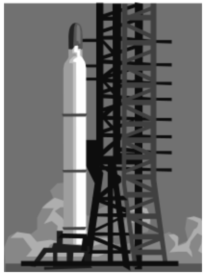
Three concerns for enterprise data analytics

- Scalability

- Performance

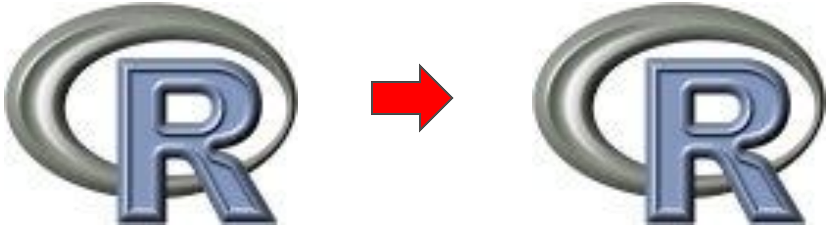
- Production Deployment





A fourth concern…

- Remain in the R language and environment
  - Same paradigm
  - SQL not required
  - Design, code, test, deploy from R



Oracle's R Technologies

- Oracle R Distribution
- ROracle
- Oracle R Enterprise
- Oracle R Advanced Analytics for Hadoop

Software available to R Community for free

Oracle R Distribution

Oracle R Distribution

Ability to dynamically load

+

Intel Math Kernel Library (MKL)

AMD Core Math Library (ACML)

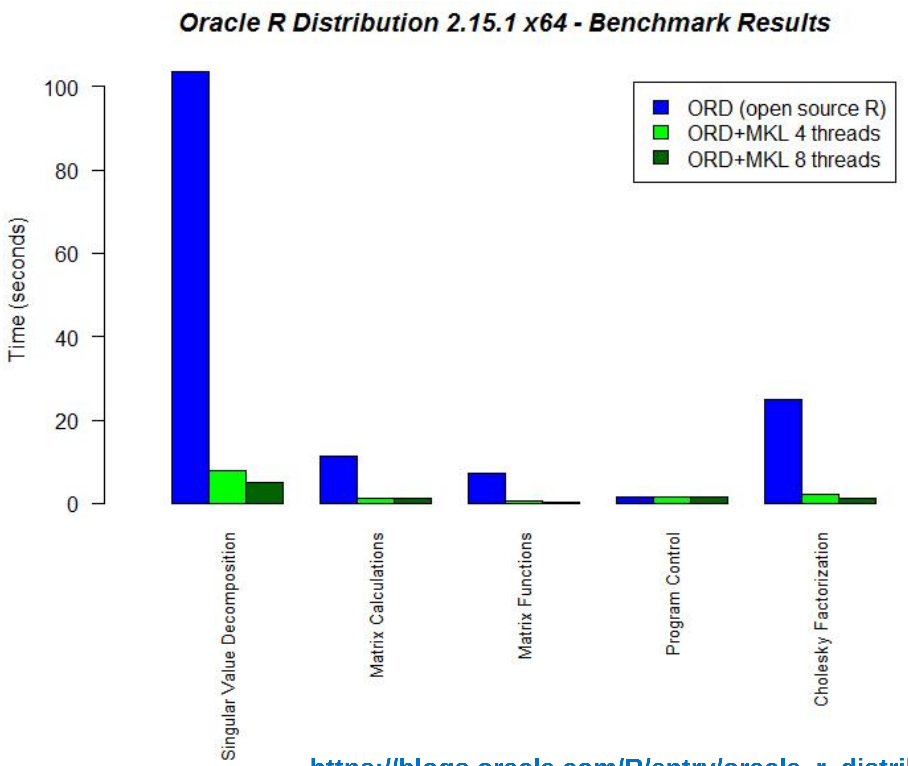
Solaris Sun Performance Library

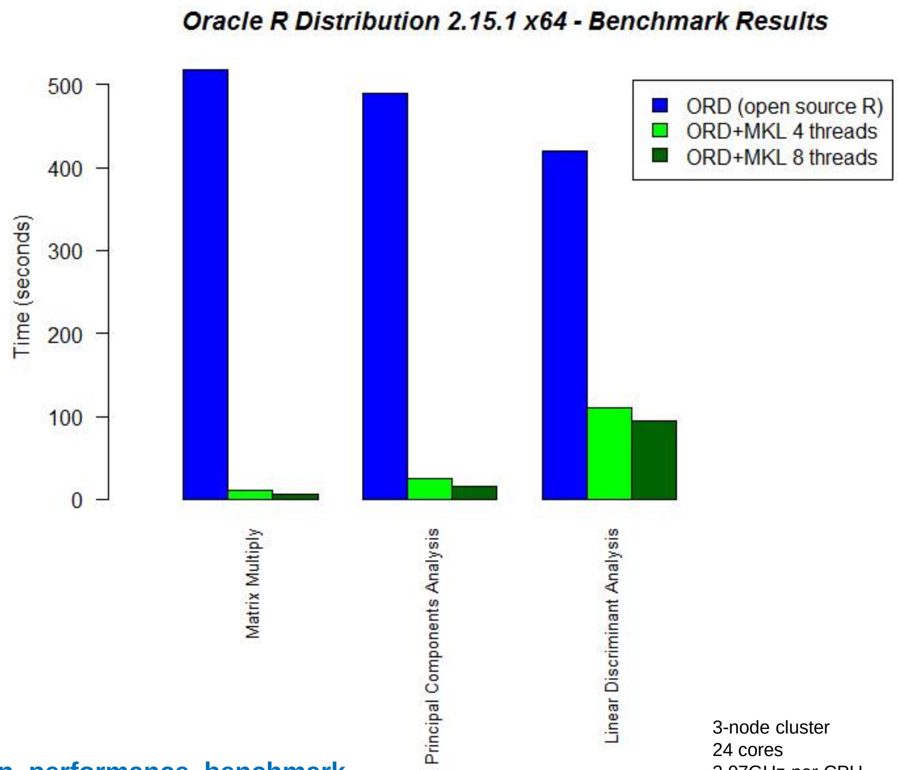
+

Oracle Support

- Oracle's redistribution of open source R
- Enhanced linear algebra performance using Intel's MKL, AMD's ACML, and Sun Performance Library for Solaris
- Improve R scalability at client and at database server for embedded R execution
- Enterprise support for customers of Oracle Advanced Analytics option, Big Data Appliance, and Oracle Linux
- **Free** download
- Oracle makes bug fixes and enhancements available for open source R

Oracle R Distribution (ORD) Performance with MKL





3-node cluster
24 cores
3.07GHz per CPU
47 GB RAM
Linux 5.5

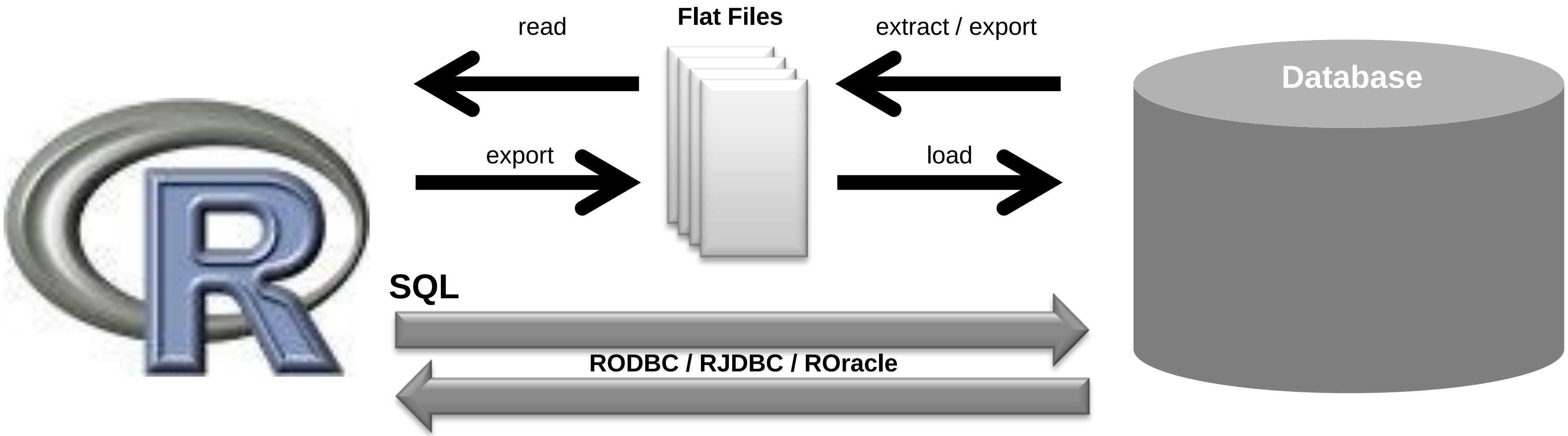
https://blogs.oracle.com/R/entry/oracle_r_distribution_performance_benchmark

Similar results for ORD 3.0.1 https://blogs.oracle.com/R/entry/oracle_r_distribution_3_0

Oracle R Enterprise

Component of the Oracle Advanced Analytics option

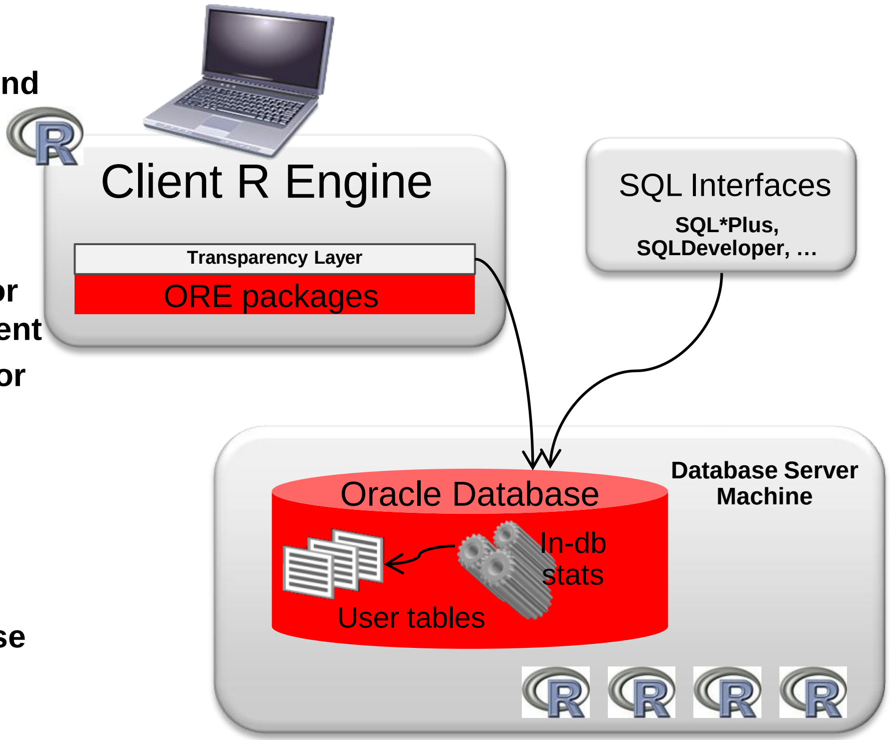
Traditional R and Database Interaction



- R memory limitation – data size, call-by-value
- R single threaded
- Paradigm shift: R → SQL → R
- Access latency, backup, recovery, security
- Ad hoc script execution or “porting” code to target environment

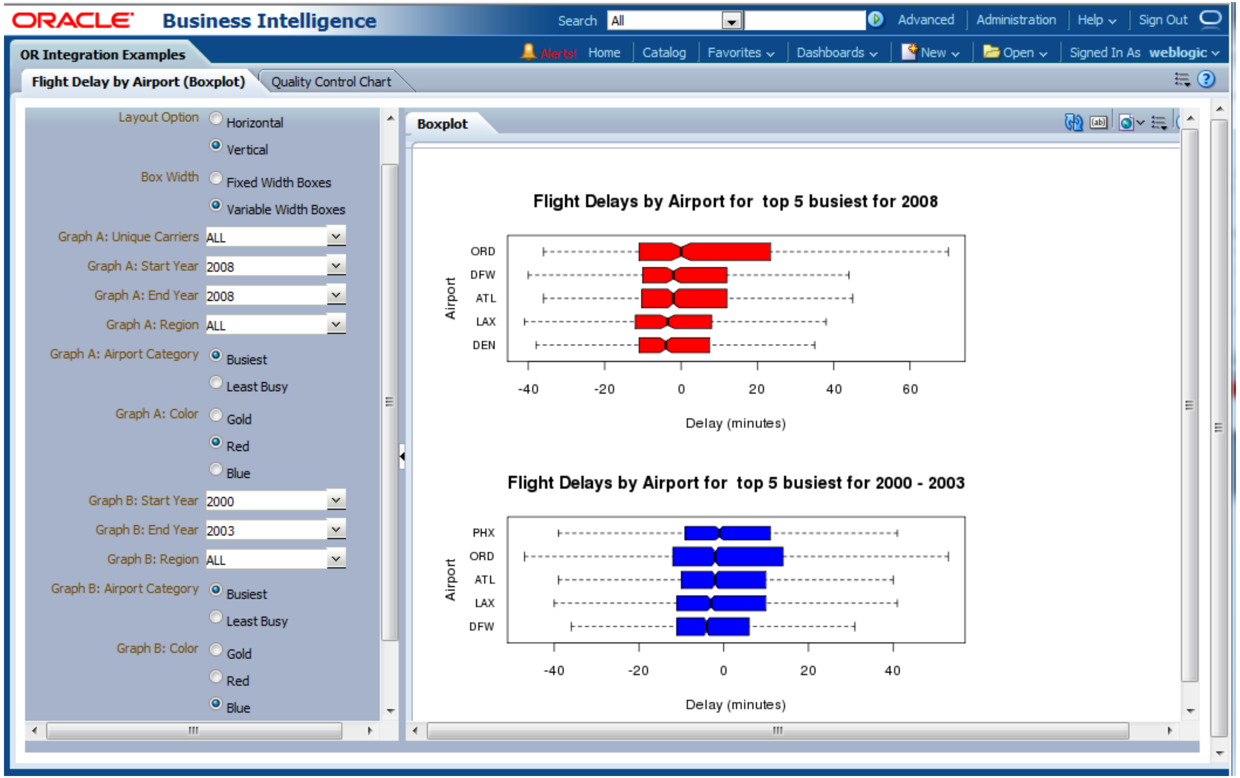
Oracle R Enterprise

- A comprehensive, database-centric environment for end-to-end analytical processes in R, with immediate deployment to production environments
- Operationalize entire R scripts in production applications – eliminate porting R code
- Seamlessly leverage Oracle Database as HPC environment for R scripts, providing data parallelism and resource management
- Execute R scripts through Oracle Database server machine for scalability and performance
- Enable integration and management through SQL
- Avoid reinventing code to integrate R results into existing applications
- Score R models in Oracle Database
- Transparently analyze and manipulate data in Oracle Database through R using versatile and customizable R functions
- Eliminate memory constraint of client R engine
- Get maximum value from your Oracle Database and Exadata
- Integrate R into the IT software stack, e.g. OBIEE



OBIEE Dashboard Integration

Parameterized analytics and graph customization



Improve time to insight

Accommodate diverse consumption paths

Deliver analytics that scale with data volumes, variables, techniques

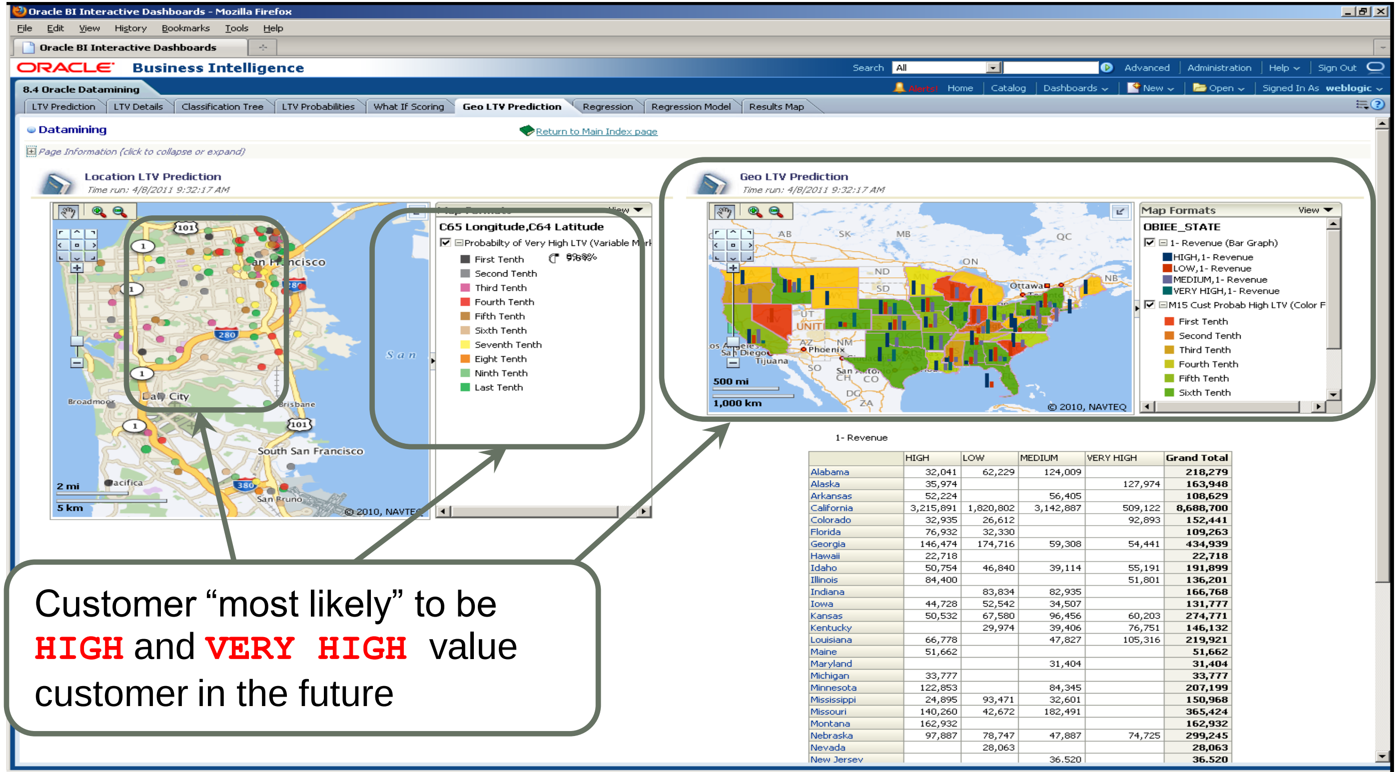
Integrate readily with IT infrastructure and software stack

Leverage CRAN packages at database server

Integrated Business Intelligence

Integrate a range of in-DB SQL & R Predictive Analytics & Graphics

- In-database construction of predictive models that predict customer behavior
- OBIEE's integrated spatial mapping shows where



1- Revenue

	HIGH	LOW	MEDIUM	VERY HIGH	Grand Total
Alabama	32,041	62,229	124,009		**218,279**
Alaska	35,974			127,974	**163,948**
Arkansas	52,224		56,405		**108,629**
California	3,215,891	1,820,802	3,142,887	509,122	**8,688,700**
Colorado	32,935	26,612		92,893	**152,441**
Florida	76,932	32,330			**109,263**
Georgia	146,474	174,716	59,308	54,441	**434,939**
Hawaii	22,718				**22,718**
Idaho	50,754	46,840	39,114	55,191	**191,899**
Illinois	84,400			51,801	**136,201**
Indiana		83,834	82,935		**166,768**
Iowa	44,728	52,542	34,507		**131,777**
Kansas	50,532	67,580	96,456	60,203	**274,771**
Kentucky		29,974	39,406	76,751	**146,132**
Louisiana	66,778		47,827	105,316	**219,921**
Maine	51,662				**51,662**
Maryland			31,404		**31,404**
Michigan	33,777				**33,777**
Minnesota	122,853		84,345		**207,199**
Mississippi	24,895	93,471	32,601		**150,968**
Missouri	140,260	42,672	182,491		**365,424**
Montana	162,932				**162,932**
Nebraska	97,887	78,747	47,887	74,725	**299,245**
Nevada		28,063			**28,063**
New Jersey			36,520		**36,520**

Customer "most likely" to be **HIGH** and **VERY HIGH** value customer in the future

Oracle Database 12c Parallel Distributed Advanced Analytics

Real world proof points

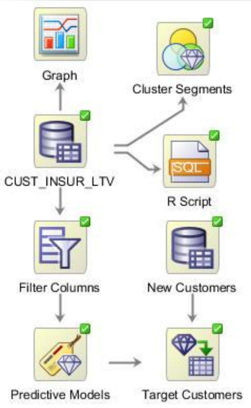
- Linear Regression (`ore.lm`) on Exadata X3-2 half-rack
  - Data set: 2.9 billion rows spanning 12 months of data with over 350 predictors
  - Elapsed time ~5 minutes!
- Logistic Regression (`ore.glm`) on Exadata X3-2 half-rack
  - Data set: 2.9 billion rows spanning 12 months of data with over 350 predictors
  - Elapsed time ~30 minutes!
- Neural networks (`ore.neural`) on T5-4 Solaris
  - Data set: 1 billion rows with 40 columns
  - Elapsed time ~6 minutes with 10 hidden neurons & 421 weights

Processing data at this scale not feasible with vanilla R

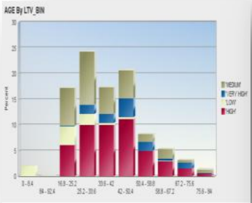
Oracle Advanced Analytics
Option to Oracle Database EE

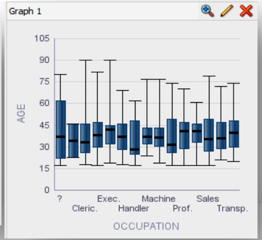
Oracle Advanced Analytics Option

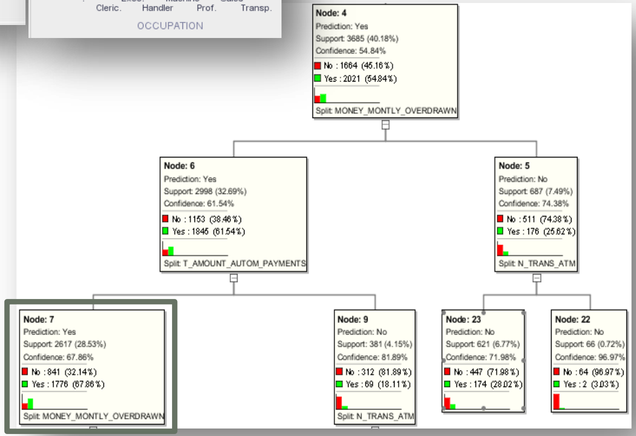
Fastest Way to Deliver Scalable Enterprise-wide Predictive Analytics



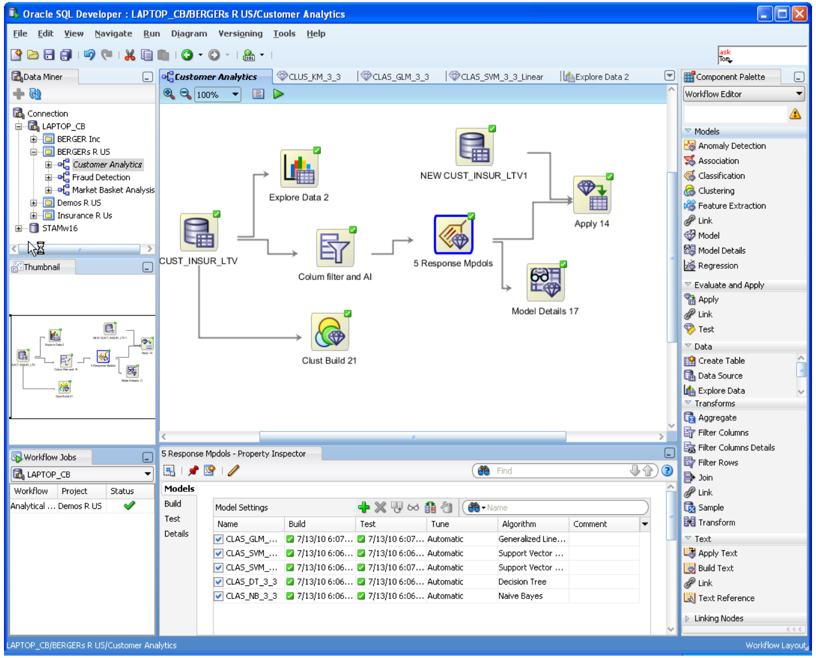
- **Better Decisions with Deeper Insights & Predictive Analytics**
  - Understand and predict customer behavior for churn, fraud, cross-sell, etc. problems
- **Easy to Use**
  - Data analysts: Mining work flow GUI (part of SQL Developer)
  - Data scientists: SQL and R languages supported
  - DBA: SQL integration
- **Comprehensive Analytics on a Simple Architecture**
  - Performance and scalability of the Oracle Database
  - Lowest Total Costs of Ownership; no need for separate analytical servers

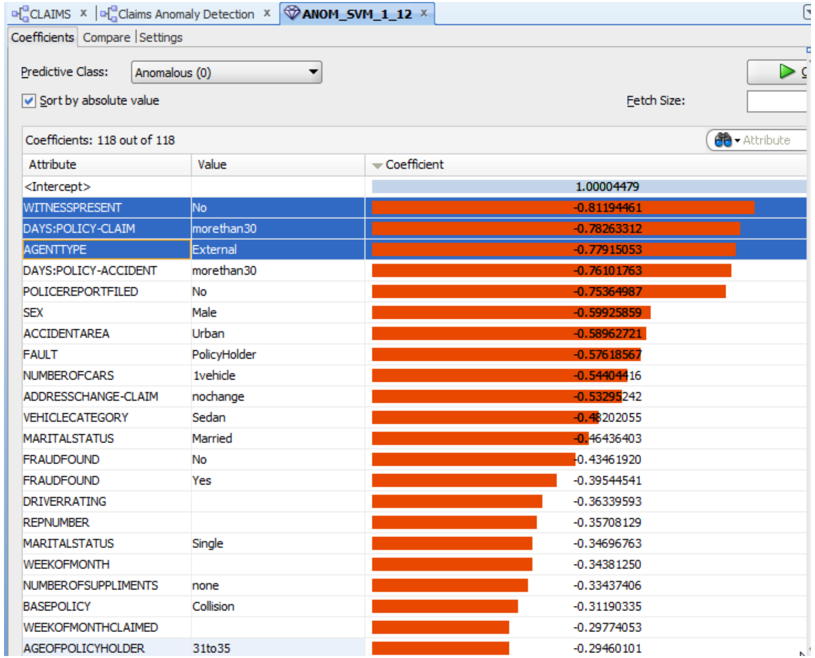






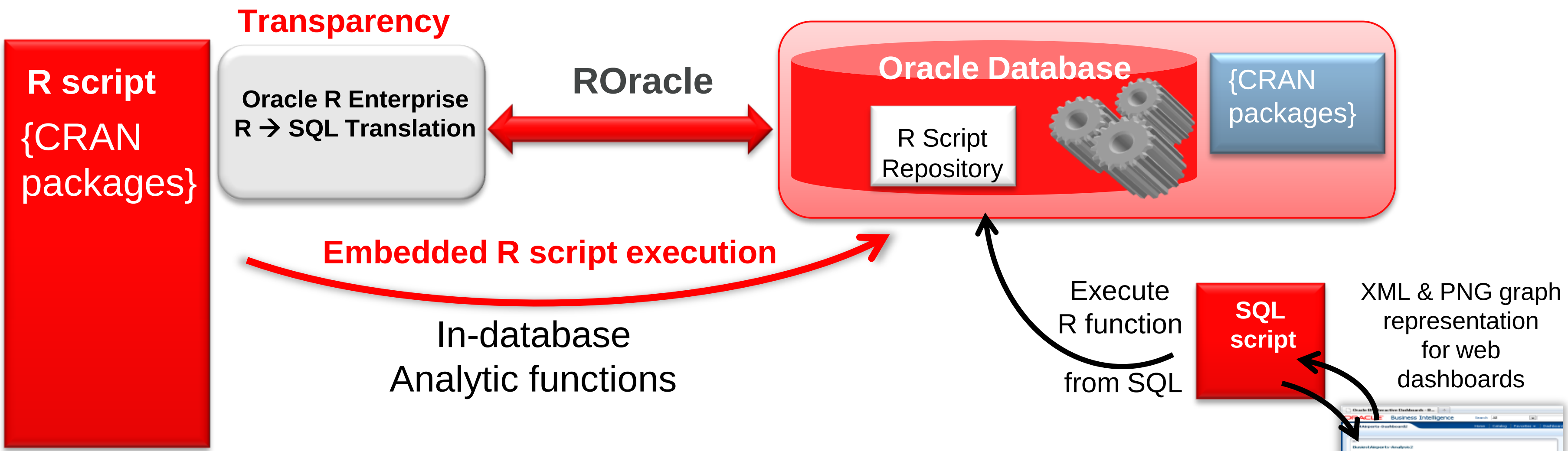
GUI for automated analytics

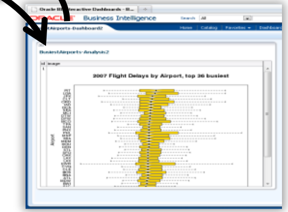




ORACLE

R integration for data analysts/scientists





Benefits

1. Enables development of 'one-stop' R scripts for end-to-end analytical process running in-database
2. SQL-R integration allows immediate operationalization of R scripts
3. SQL-R integration allows any IT software to readily leverage advanced analytics
4. Enables the database to serve as a high performance compute platform for R quants

ORE Transparency Layer

Transparency

- No need to learn a different programming paradigm or environment
- Operate on database data as though they were R objects using R syntax
- Require minimal change to base R scripts for database data
- Implicitly translates R to SQL for in-database execution, performance, and scalability

The Transparency Layer supports in-database data exploration, data preparation, and data analysis en route performing predictive analytics with a mix of in-database and CRAN techniques.

Establish a connection to Oracle Database

```
library(ORE)

ore.connect(user="rquser", sid="orcl",
            host="localhost", password="rquser", all=TRUE)
ore.ls()
```

```
> ore.connect("rquser","orcl","localhost","rquser",1521, all=TRUE)
> ore.ls()
[1] "ALL_2011"          "ALL_2011_DT_RULES" "ALL_2011_PREDS"    "CLAIMS"            "IRIS"
[6] "NARROW"            "ONTIME_S"          "TEST_DF1"          "TEST_DF2"
```

Data Selection

- Column selection

```
df <- ONTIME_S[,c("YEAR","DEST","ARRDELAY")]
class(df)

head(df)
head(ONTIME_S[,c(1,4,23)])
head(ONTIME_S[,-(1:22)])
```

- Row selection

```
df1 <- df[df$DEST=="SFO",]
class(df1)

df2 <- df[df$DEST=="SFO",c(1,3)]
df3 <- df[df$DEST=="SFO" | df$DEST=="BOS",1:3]
head(df1)
head(df2)
head(df3)
```

```
R> df <- ONTIME_S[,c("YEAR","DEST","ARRDELAY")]
R> class(df)
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
R>
R> head(df)
  YEAR DEST ARRDELAY
1 1987  MSP        4
2 1987  SJC        6
3 1987  OAK        7
4 1987  PHX        9
5 1987  CLT        0
6 1987  CVG        4
R> head(ONTIME_S[,c(1,4,23)])
  YEAR DAYOFMONTH TAXIOUT
1 1987          1      NA
2 1987          1      NA
3 1987          1      NA
4 1987          1      NA
5 1987          1      NA
6 1987          1      NA
R> head(ONTIME_S[,-(1:22)])
  TAXIOUT CANCELLED CANCELLATIONCODE I
1      NA         0             <NA>
2      NA         0             <NA>
3      NA         0             <NA>
4      NA         0             <NA>
5      NA         0             <NA>
6      NA         0             <NA>
```

```
R> df1 <- df[df$DEST=="SFO",]
R> class(df1)
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
R>
R> df2 <- df[df$DEST=="SFO",c(1,3)]
R> df3 <- df[df$DEST=="SFO" | df$DEST=="BOS",1:3]
R> head(df1)
  YEAR DEST ARRDELAY
1 1987  SFO       24
2 1987  SFO       68
3 1987  SFO       -3
4 1987  SFO        5
5 1987  SFO       37
6 1987  SFO       11
R> head(df2)
  YEAR ARRDELAY
1 1987       24
2 1987       68
3 1987       -3
4 1987        5
5 1987       37
6 1987       11
R> head(df3)
  YEAR DEST ARRDELAY
1 1987  SFO       24
2 1987  SFO       68
3 1987  SFO       -3
4 1987  SFO        5
5 1987  SFO       37
6 1987  BOS       NA
```

ORACLE

Summarize Data

```
res <- summary(ONTIME_S[,1:13])
class(res)    # table
res
```

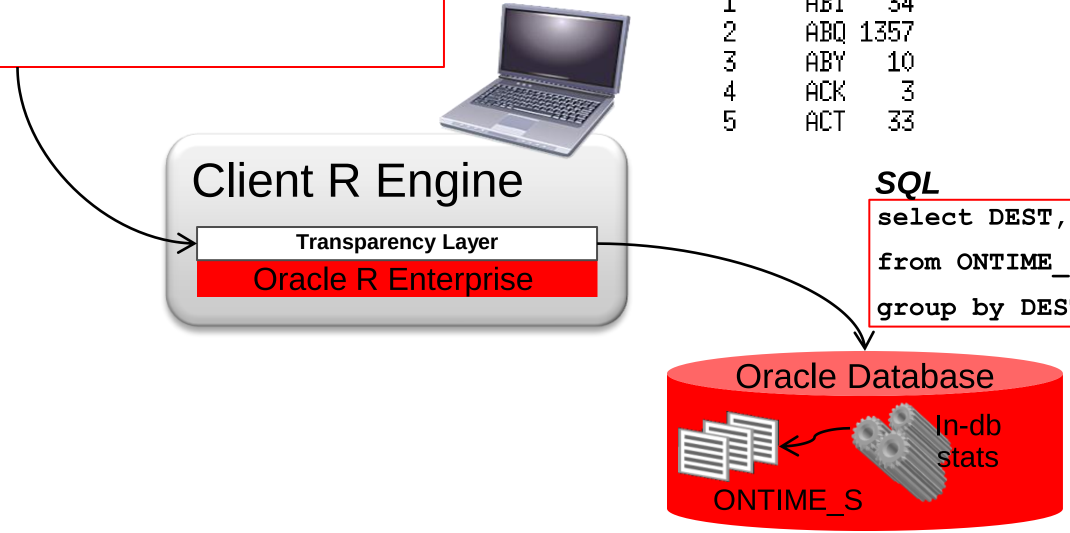
```
> res <- summary(ONTIME_S[,1:13])
> class(res)
[1] "table"
> res
      YEAR           MONTH              MONTH2          DAYOFMONTH        DAYOFMONTH2        DAYOFWEEK         DEPTIME
 Min.   :1996   Min.   : 1.000   M7      :11246   Min.   : 1.00   D11      :  4489   Min.   :1.000   Min.   :   1
 1st Qu.:1999   1st Qu.: 4.000   M8      :11234   1st Qu.: 8.00   D21      :  4417   1st Qu.:2.000   1st Qu.: 932
 Median :2002   Median : 7.000   M3      :11024   Median :16.00   D7       :  4388   Median :4.000   Median :1332
 Mean   :2002   Mean   : 6.514   M6      :10934   Mean   :15.75   D2       :  4347   Mean   :3.932   Mean   :1347
 3rd Qu.:2005   3rd Qu.: 9.000   M10     :10929   3rd Qu.:23.00   D28      :  4345   3rd Qu.:6.000   3rd Qu.:1736
 Max.   :2008   Max.   :12.000   M5      :10928   Max.   :31.00   D23      :  4327   Max.   :7.000   Max.   :2617
                                 (Other):63374                    (Other):103356                    NA's   :2766
   CRSDEPTIME        ARRTIME        CRSARRTIME     UNIQUECARRIER      FLIGHTNUM         TAILNUM
 Min.   :   0   Min.   :   1   Min.   :   0   WN     :20187   Min.   :   1   #NAME?  :  1814
 1st Qu.: 925   1st Qu.:1116   1st Qu.:1115   DL     :15654   1st Qu.: 514   UNKNOW  :   883
 Median :1325   Median :1521   Median :1520   AA     :14954   Median :1146   0       :   538
 Mean   :1329   Mean   :1492   Mean   :1489   UA     :13464   Mean   :1597   ¿NKNO¿  :   273
 3rd Qu.:1725   3rd Qu.:1918   3rd Qu.:1912   US     :12425   3rd Qu.:1994   N510    :    69
 Max.   :2359   Max.   :2722   Max.   :2400   NW     :10570   Max.   :9599   (Other) :125882
                NA's   :3066                  (Other) :42415                  NA's    :   210
```

Aggregate Data

R

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

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



SQL

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

Data preparation – recoding and binning

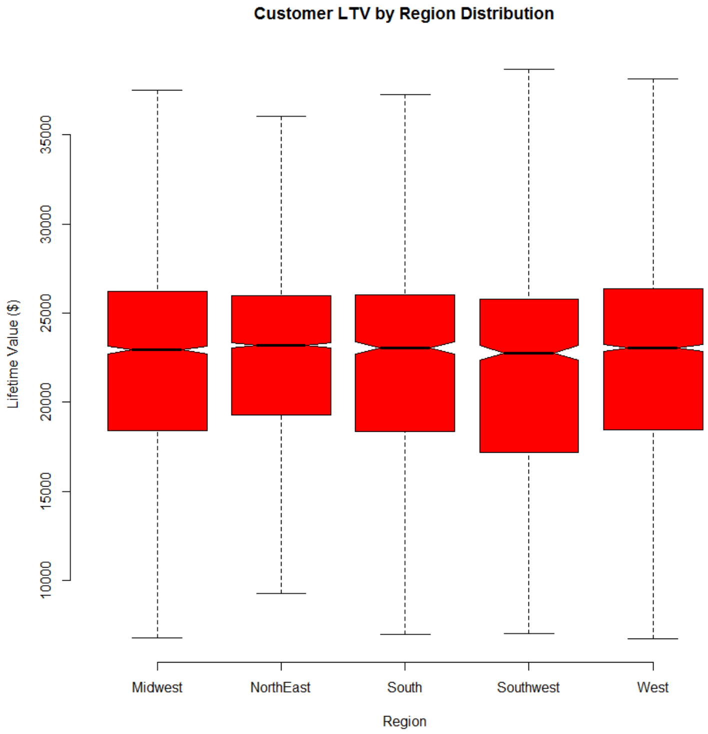
Using transform ( )

```
ONTIME <- transform(ONTIME_S,
      DIVERTED = ifelse(DIVERTED == 0, 'Not Diverted',
                 ifelse(DIVERTED == 1, 'Diverted', '')),

      CANCELLATIONCODE =
                 ifelse(CANCELLATIONCODE == 'A', 'A CODE',
                 ifelse(CANCELLATIONCODE == 'B', 'B CODE',
                 ifelse(CANCELLATIONCODE == 'C', 'C CODE',
                 ifelse(CANCELLATIONCODE == 'D', 'D CODE', 'NOT CANCELLED')))),

      ARRDELAY = ifelse(ARRDELAY > 200, 'LARGE',
                 ifelse(ARRDELAY >= 30, 'MEDIUM', 'SMALL')),

      DEPDELAY = ifelse(DEPDELAY > 200, 'LARGE',
                 ifelse(DEPDELAY >= 30, 'MEDIUM', 'SMALL')),

      DISTANCE_ZSCORE =(DISTANCE - mean(DISTANCE, na.rm=TRUE))/sd(DISTANCE, na.rm=TRUE))
head(ONTIME)
```

Visualize Data

Overloaded graphics functions for in-database statistics

```
dat <- LTV
value <- CUST_LIFETIME_VALUE$LTV
part <- dat$REGION
bd <- split(value, part)
boxplot(bd, notch = TRUE, col = "red", cex = 0.5,
        outline = FALSE, axes = FALSE,
        main = "Customer LTV by Region Distribution",
        ylab = "Lifetime Value ($)", xlab = "Region")
axis(1, at=1:length(levels(part)), labels=levels(part))
axis(2)
```



ORE Analytics Packages and Functions

High performance in-database predictive techniques available through ORE packages

OREdm package

- Support Vector Machine
- Generalized Linear Model
- K-Means clustering
- OC clustering
- Naïve Bayes
- Decision Trees
- Association Rules
- Attribute Importance

OREmodels package

- Neural Networks
- Linear Regression
- Stepwise Regression
- Generalized Linear Model

R Interface to In-Database Statistical Functions

- Special Functions
  - Gamma function
  - Natural logarithm of the Gamma function
  - Digamma function
  - Trigamma function
  - Error function
  - Complementary error function
- Tests
  - Chi-square, McNemar, Bowker
  - Simple and weighted kappas
  - Cochran-Mantel-Haenzel correlation
  - Cramer's V
  - Binomial, KS, t, F, Wilcox
- Base SAS equivalents
  - Freq, Summary, Sort
  - Rank, Corr, Univariate
- Density, Probability, and Quantile Functions
  - Beta distribution
  - Binomial distribution
  - Cauchy distribution
  - Chi-square distribution
  - Exponential distribution
  - F-distribution
  - Gamma distribution
  - Geometric distribution
  - Log Normal distribution
  - Logistic distribution
  - Negative Binomial distribution
  - Normal distribution
  - Poisson distribution
  - Sign Rank distribution
  - Student's t distribution
  - Uniform distribution
  - Weibull distribution
  - Density Function
  - Probability Function
  - Quantile

ORACLE

ORE Embedded R Script Execution

Embedded R Execution

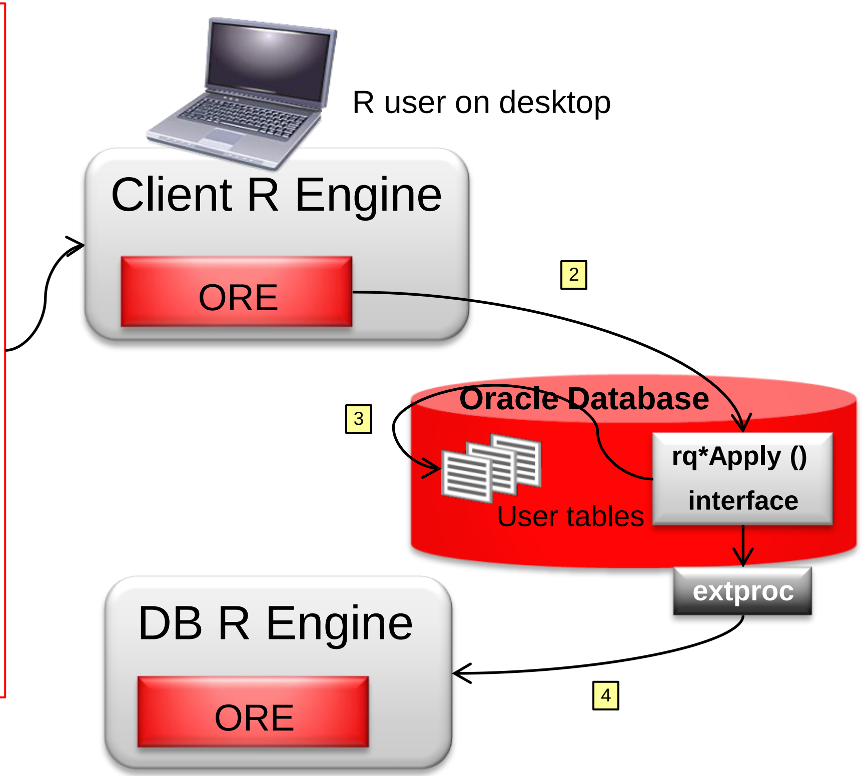
- Ability to execute R code on the database server
- Execution controlled and managed by Oracle Database
- Eliminates loading data to the user's R engine and result write-back to Oracle Database
- Enables data- and task-parallel execution of R functions
- Enables SQL access to R: invocation and results
- Supports use of open source CRAN packages at the database server
- R scripts can be stored and managed in the database
- Schedule R scripts for automatic execution

Motivation – why embedded R execution?

- Facilitate application use of R script results
  - Develop/test R scripts interactively with R interface
  - Invoke R scripts directly from SQL for production applications
  - R Scripts stored in Oracle Database
- Improved performance and throughput
  - Oracle Database data- and task-parallelism
  - Compute and memory resources of database server, e.g., Exadata
  - More efficient read/write of data between Oracle Database and R Engine
  - Parallel simulations
- Image generation at database server
  - Available to OBIEE and BI Publisher, or any such consumer
  - Rich XML, image streams

ore.tableApply with parameter passing

```
build.GLM.model <- function(dat, family) {
     mod <- glm(ARRDELAY ~ DISTANCE + DEPDELAY,
                  data=dat, family=family)
     coef(mod)
     }

class(ONTIME_S)  # ore.frame

modCoef <- ore.tableApply(
  ONTIME_S[,c("ARRDELAY","DISTANCE","DEPDELAY")],
  build.GLM.model,
  family = gaussian());
modCoef
```



ore.tableApply using CRAN package

```
dat.ore <- ore.push(iris)
library(e1071)

build.NB.model <- function(dat) {
    library(e1071)
    dat$Species <- as.factor(dat$Species)
    naiveBayes(Species ~ ., dat)
 }

mod <- ore.tableApply(dat.ore, build.NB.model)
class(mod)
mod
local.mod <- ore.pull(mod)
```

```
R> dat.ore <- ore.push(iris)
R> library(e1071)
Loading required package: class
R>
R> build.NB.model <- function(dat) {
+     library(e1071)
+     dat$Species <- as.factor(dat$Species)
+     naiveBayes(Species ~ ., dat)
+  }
R>
R> mod <- ore.tableApply(dat.ore, build.NB.model)
R> class(mod)
[1] "ore.object"
attr(,"package")
[1] "OREembed"
R> mod

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Conditional probabilities:
            Sepal.Length
Y             [,1]      [,2]
  setosa     5.006 0.3524897
  versicolor 5.936 0.5161711
  virginica  6.588 0.6358796
```

ore.tableApply with batch scoring returning ore.frame

```
score.NB.model <- function(dat, mod) {
     library(e1071)
     dat$Species <- as.factor(dat$Species)
     dat$PRED <- predict(mod, newdata = dat)
     dat
   }
IRIS <- ore.push(iris)
IRIS_PRED <- IRIS[1,]
IRIS_PRED$PRED <- "A"
res <- ore.tableApply(
   IRIS, score.NB.model,
   mod = local.mod,
   FUN.VALUE = IRIS_PRED)
class(res)
head(res)
```

```
R> score.NB.model <- function(dat, mod) {
+       library(e1071)
+       dat$Species <- as.factor(dat$Species)
+       dat$PRED <- predict(mod, newdata = dat)
+       dat
+     }
R> IRIS <- ore.push(iris)
R> IRIS_PRED <- IRIS[1,]
R> IRIS_PRED$PRED <- "A"
R> res <- ore.tableApply(
+     IRIS, score.NB.model,
+     mod = local.mod,
+     FUN.VALUE = IRIS_PRED)
R> class(res)
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
R> head(res)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species   PRED
1          5.1         3.5          1.4         0.2  setosa setosa
2          4.9         3.0          1.4         0.2  setosa setosa
3          4.7         3.2          1.3         0.2  setosa setosa
4          4.6         3.1          1.5         0.2  setosa setosa
5          5.0         3.6          1.4         0.2  setosa setosa
6          5.4         3.9          1.7         0.4  setosa setosa
```

ore.rowApply – data parallel scoring

```
res <- ore.rowApply(
   IRIS ,
   score.NB.model,
   mod = local.mod,
   FUN.VALUE = IRIS_PRED,
   rows=10)
class(res)
table(res$Species, res$PRED)
```

```
R> res <- ore.rowApply(
+    IRIS ,
+    score.NB.model,
+    mod = local.mod,
+    FUN.VALUE = IRIS_PRED,
+    rows=10)
R> class(res)
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
R> table(res$Species, res$PRED)

             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         47         3
  virginica       0          3        47
```

Goal: Score data in batch (rows=10) using data from input ore.frame

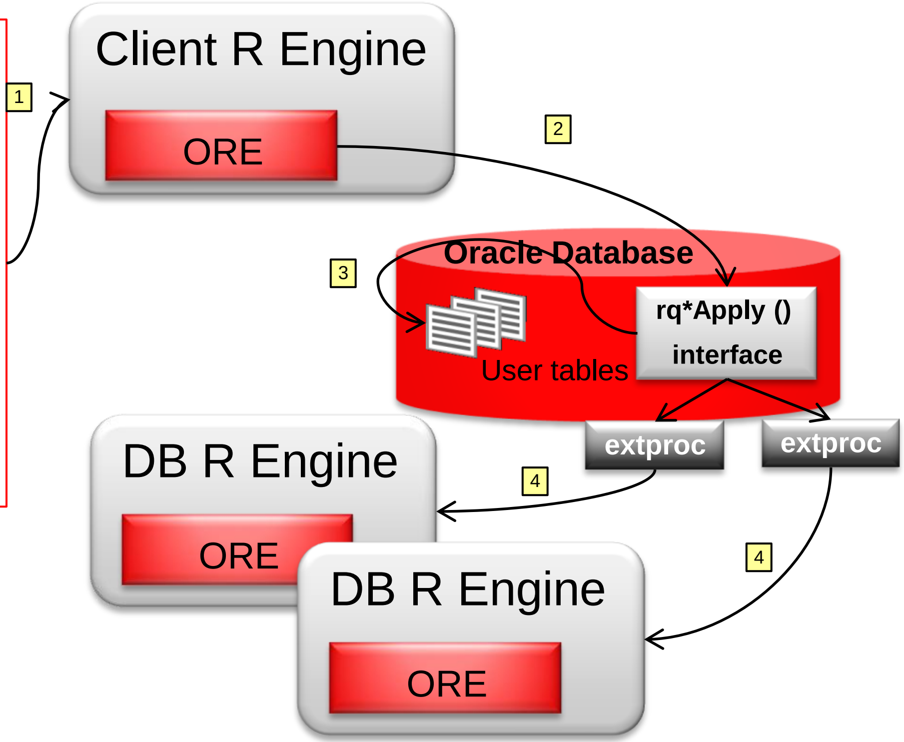
Data set loaded into R memory at database R Engine and passed to function

Return value specified using IRIS_PRED as *example* representation.

Result returned as ore.frame

ore.groupApply – partitioned data flow

```
build.LM.model <- function(dat) {
      lm(ARRDELAY ~ DISTANCE + DEPDELAY, dat)
}

modList <- ore.groupApply(X=ONTIME_S,
                          INDEX=ONTIME_S$DEST,
                          build.LM.model);
class(modList)
modList_local <- ore.pull(modList)
summary(modList_local$BOS) ## return model for BOS
```

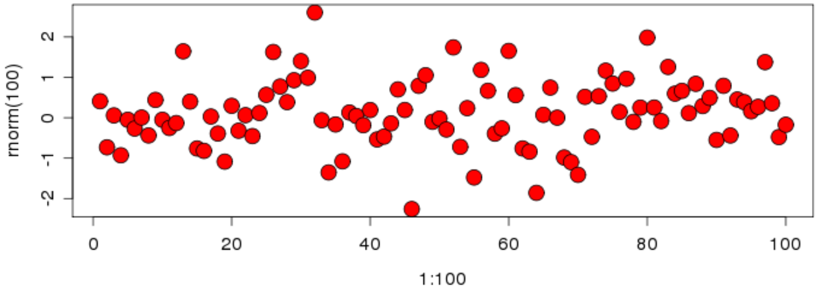


Production Deployment – same R function, multiple uses

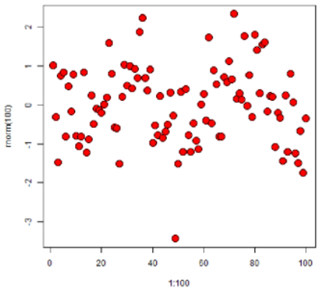
SQL

```
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
```



Results

‘PNG’ result

	ID	IMAGE
1	1	(BLOB)

‘select 1 id, 1 val from dual’ result

	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	0.1

‘XML’ result

```
SQL> set long 20000
set pages 1000
begin
  sys.rqScriptCreate('Example6',
 'function(){
            res <- 1:10
            plot( 1:100, rnorm(100), pch = 21,
                  bg = "red", cex = 2 )
            res
            }');
SQL> end;
/
select    value
from      table(rqEval( NULL,'XML','Example6'));
SQL>   2    3    4    5    6    7    8    9   10
PL/SQL procedure successfully completed.

SQL>   2
VALUE
--------------------------------------------------------------------------------
```


```
<root><R-data><vector_obj> <ROW-vector_obj><value>1</value></ROW-vector_obj><ROW
-vector_obj><value>2</value></ROW-vector_obj><ROW-vector_obj><value>3</value></R
OW-vector_obj><ROW-vector_obj><value>4</value></ROW-vector_obj><ROW-vector_obj><
value>5</value></ROW-vector_obj><ROW-vector_obj><value>6</value></ROW-vector_obj
><ROW-vector_obj><value>7</value></ROW-vector_obj><ROW-vector_obj><value>8</valu
e></ROW-vector_obj><ROW-vector_obj><value>9</value></ROW-vector_obj><ROW-vector_
obj><value>10</value></ROW-vector_obj></vector_obj> </R-data><images><image><img
 src="data:image/pngbase64"><![CDATA[iVBORw0KGgoAAAANSUhEUgAAAeAAAAHgCAIAAADytin
CAAAgAElEQVR4nOzdZ1xT1x8G8CcMB6jgQq0IDnDVulsRBSKyZQjIUnCDKDhq3bvuValbcRYFFRFRUBF
ExYWrarGKA3GAgw0udvJ/wV8aTG5ESG4C/L4fXug9Jzd+PGL/c3HvuORw+nw9CCCHyR0HWAQghhIhGBZo
QQuQUFWhCCJFTVKAJIURUYEmhBA5RQWaEELkFBVoQgiRU1SgCSFET1GBJoQQQ0UUFmhBC5BQVaEIIkVN
UoAkhRE5RgSaEED1FBZoQQuQUFWhCCJFTVKAJIUR0UYEmhBA5RQWaEELkFBVoQgiRU1SgCSFET1GBJoQ
Q0UUFmhBC5BQVaEIIkVNUoAkhRE5RgSaEED1FBZoQQuQUFWhCCJFTVKAJIUR0UYEmhBA5RQWaEELkFBV
nQoiRU1SgCSFET1GBJoQQ0UUFmhBC5BQVaEIIkVNUoAkhRE5RgSaEED1FBZoQQuQUFWhCCJFTVKAJIUR
```

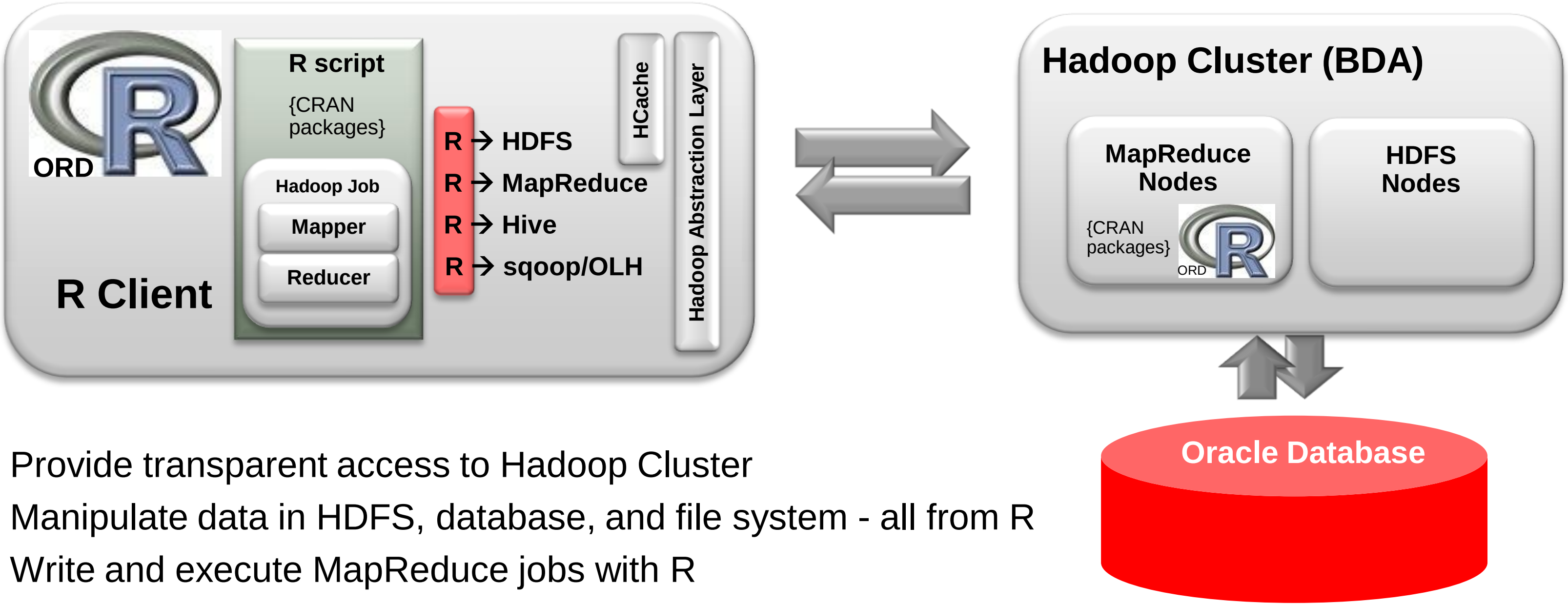
Oracle R Advanced Analytics for Hadoop

Component of the Big Data Connectors Software Suite, option for BDA

Goals

- Expand user population that can build models on Hadoop
- Accelerate rate at which business problems are tackled
- Deliver analytics that scale
  - Data volumes
  - Variables
  - Techniques

Oracle R Advanced Analytics for Hadoop

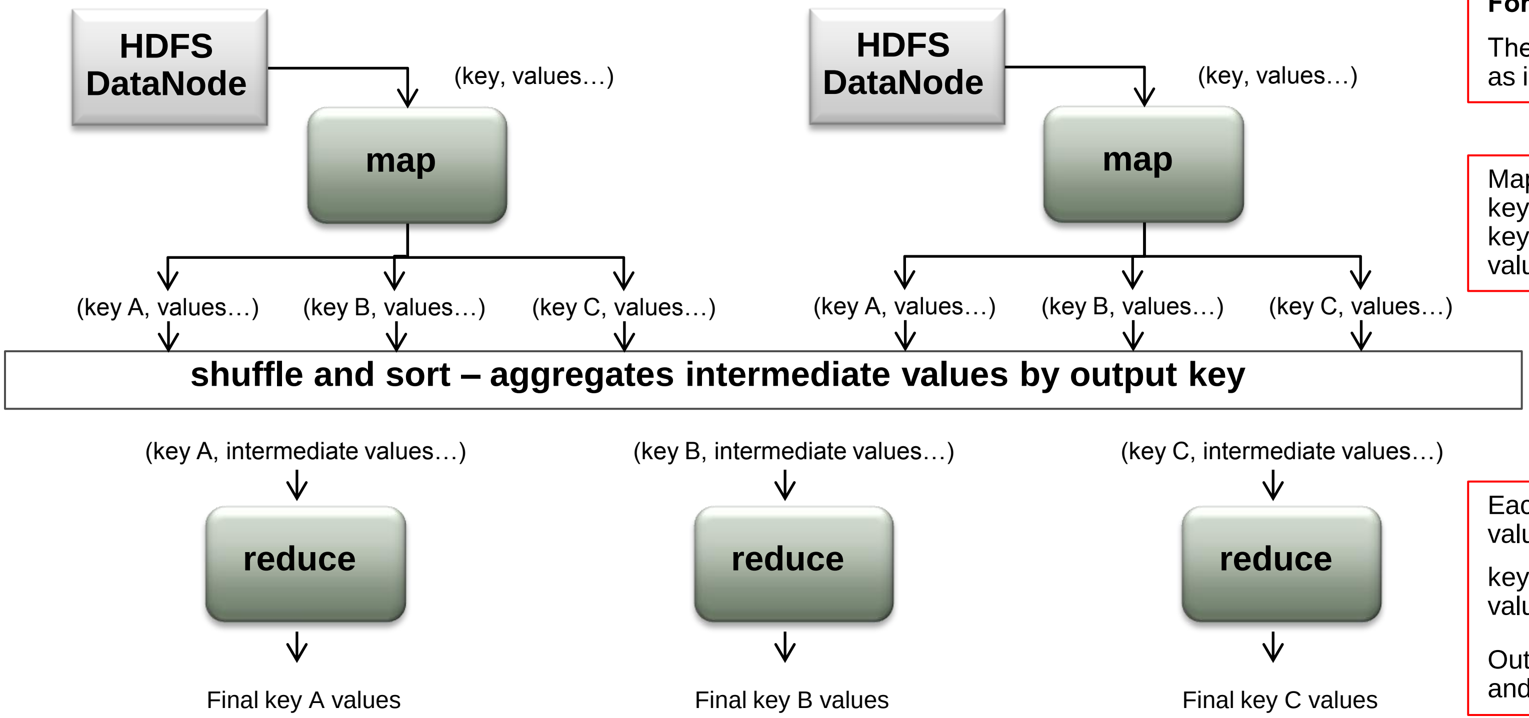


- Provide transparent access to Hadoop Cluster
- Manipulate data in HDFS, database, and file system - all from R
- Write and execute MapReduce jobs with R
- Leverage CRAN R packages to work on HDFS-resident data
- Move from lab to production without requiring knowledge of Hadoop internals, Hadoop CLI, or IT infrastructure

ORAAH Analytics Functions

Function	Description
orch.cor	Correlation matrix computation
orch.cov	Covariance matrix computation
orch.kmeans	Perform k-means clustering on a data matrix stored as an HDFS file. Score data using orch.predict.
orch.lm	Fits a linear model using tall-and-skinny QR (TSQR) factorization and parallel distribution. The function computes the same statistical parameters as the Oracle R Enterprise ore.lm function. Score data using orch.predict.
orch.lmf	Fits a low rank matrix factorization model using either the jellyfish algorithm or the Mahout alternating least squares with weighted regularization (ALS-WR) algorithm.
orch.neural	Provides a neural network to model complex, nonlinear relationships between inputs and outputs, or to find patterns in the data. Score data using orch.predict.
orch.nmf	Provides the main entry point to create a nonnegative matrix factorization model using the jellyfish algorithm. This function can work on much larger data sets than the R NMF package, because the input does not need to fit into memory.
orch.princomp	Principal components analysis of HDFS data. Score data using orch.predict.
orch.sample	Sample HDFS data by percentage or explicit number of rows specification

Map Reduce Example – Graphically Speaking



For "Word Count"

There's no key, only value as input to mapper

Mapper output is a set of key-value pairs where key is the word and value is the count=1

Each reducer receives values for each word

key is the word
value is a set of counts

Outputs key as the word and value as the sum

Mapper and reducer code in ORAAH for “Word Count”

```
corpus <- scan("corpus.dat", what=" ",quiet= TRUE, sep="\n")
corpus <- gsub("([/\\\":,#.@-])", " ", corpus)
input  <- hdfs.put(corpus)
res    <- hadoop.exec(dfs.id = input,
            mapper = function(k,v) {
               x <- strsplit(as.character(v[[1]]), " ")
               x <- unlist(x)
               x <- x[x!='']
               orch.keyvals(x,rep(1,length(x)))
            },
            reducer = function(k,vv) {
               orch.keyval(k, sum(vv$val))
            },
            config = new("mapred.config",
              job.name      = "wordcount",
              map.output    = data.frame(key='a', val=0),
              reduce.output = data.frame(key='a', val=0),
              reduce.tasks  = 30)
          )
res
hdfs.get(res)
```

Load the R data.frame into HDFS

Specify and invoke map-reduce job

Split words and output each word

Sum the count of each word

Mapper and reducer code in JAVA for “Word Count”

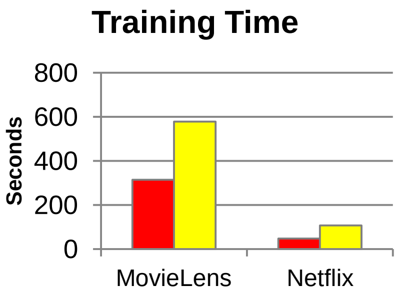
```java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WordMapper extends MapReduceBase
    implements Mapper<LongWritable, Text, Text,
   IntWritable> {
  public void map(LongWritable key, Text value,
      OutputCollector<Text, IntWritable> output,
   Reporter reporter)
      throws IOException {
     String s = value.toString();
     for (String word : s.split("\\W+")) {
       if (word.length() > 0) {
         output.collect(new Text(word), new
   IntWritable(1));
       }
     }
   }
 }
```

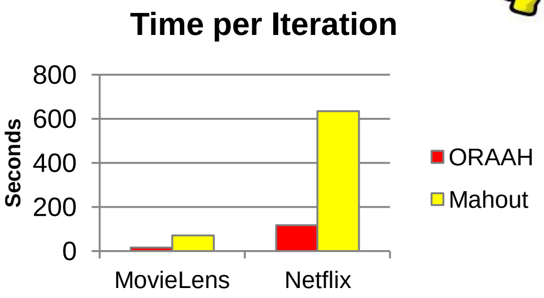
```java
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class SumReducer extends MapReduceBase implements
    Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterator<IntWritable>
   values,
      OutputCollector<Text, IntWritable> output,
   Reporter reporter)
      throws IOException {
    int wordCount = 0;
    while (values.hasNext()) {
      IntWritable value = values.next();
      wordCount += value.get();
    }
    output.collect(key, new IntWritable(wordCount));
  }
}
```

Oracle R Advanced Analytics for Hadoop

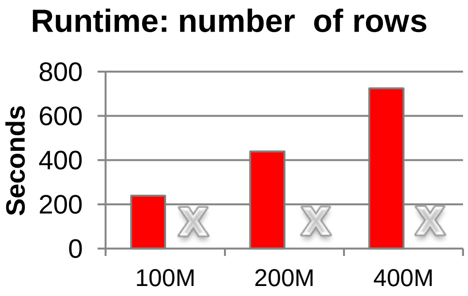
Real world proof points with Oracle Big Data Appliance and hadoop

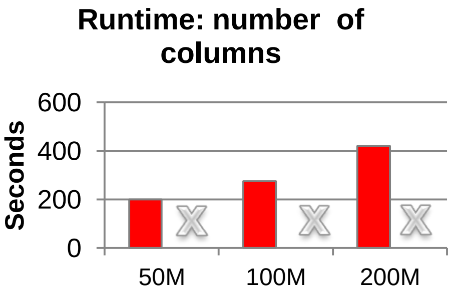
- Low Rank Matrix Factorization
  - Performance

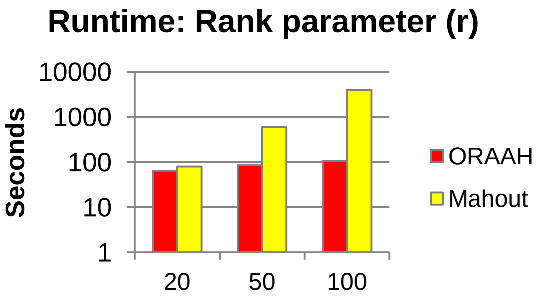




  - Scalability







X ***Mahout crashes at ~20M***

ORACLE

Oracle Big Data Platform

Oracle Big Data Appliance

Optimized for Hadoop, R, and NoSQL Processing

- Hadoop
- Oracle R Distribution
- Oracle NoSQL Database
- Applications

Oracle Big Data Connectors

- Oracle R Advanced Analytics for Hadoop + …
- Oracle Data Integrator

Oracle Exadata

"System of Record" Optimized for DW/OLTP

- Oracle R Enterprise
- Oracle Data Mining
- Data Warehouse
- Oracle Database

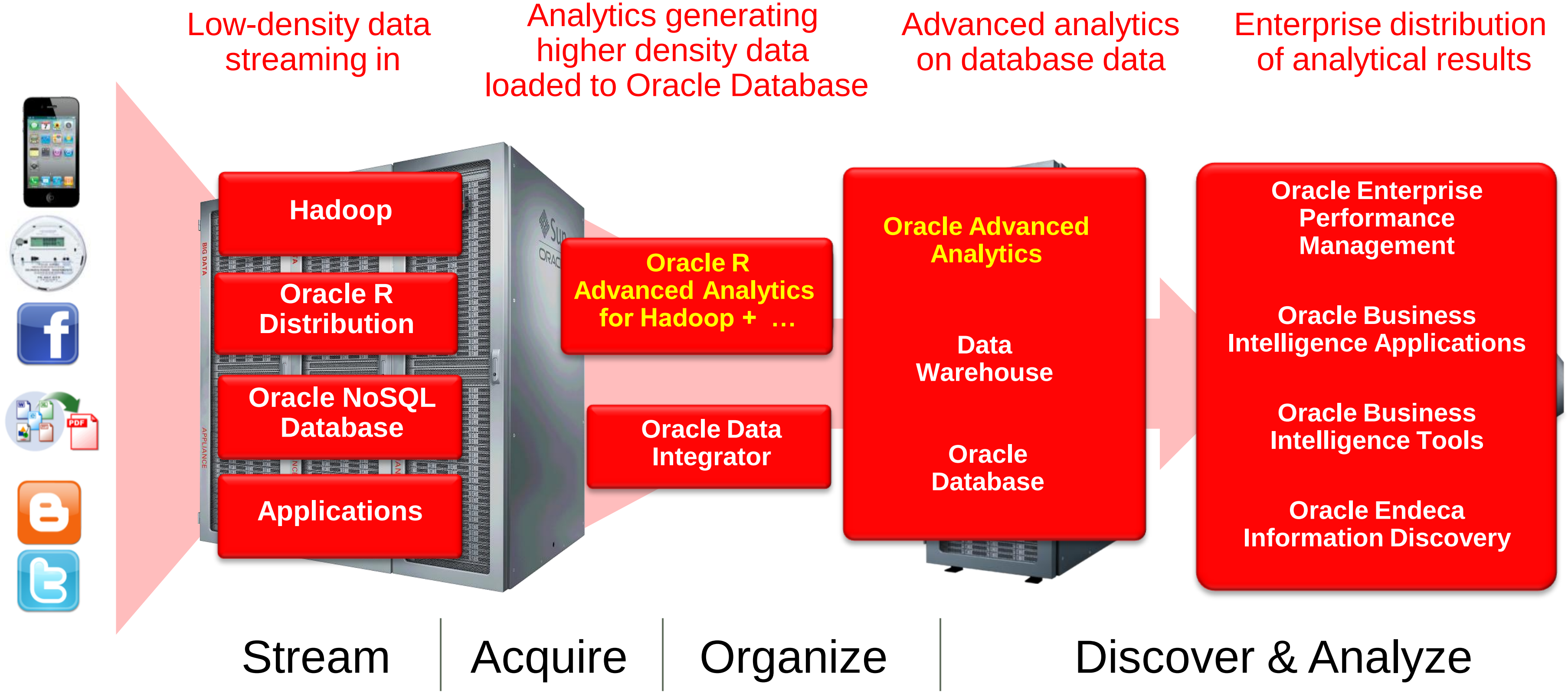
Oracle Exalytics

Optimized for Analytics & In-Memory Workloads

- Oracle Enterprise Performance Management
- Oracle Business Intelligence Applications
- Oracle Business Intelligence Tools
- Oracle Endeca Information Discovery

Stream | Acquire | Organize | Discover & Analyze

Oracle Big Data Platform



ORACLE

Use Cases

Massive Predictive Modeling

Sensor Data Analysis

- Model each customer's usage to understand behavior and predict individual usage and overall aggregate demand
- 200 thousand households, each with a utility "smart meter"
- 1 reading / meter / hour
- 200K x 8760 hours / year ➔ 1.752B readings
- 3 years worth of data ➔ 5.256B readings
- Each customer has 2628 readings
- If each model takes 10 seconds to build, 555.6 hours (23.2 days) …with 128 DOP ➔ 4.3 hours

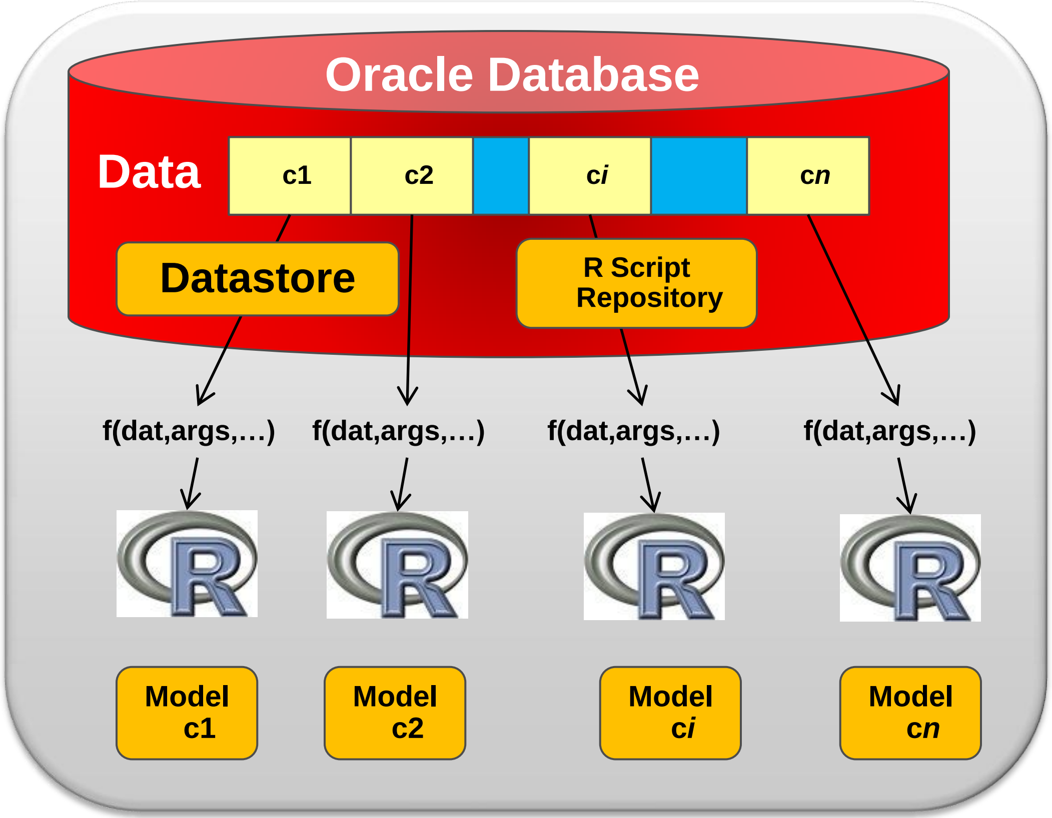
Database-centric architecture

Smart meter scenario

f(dat,args,…) {

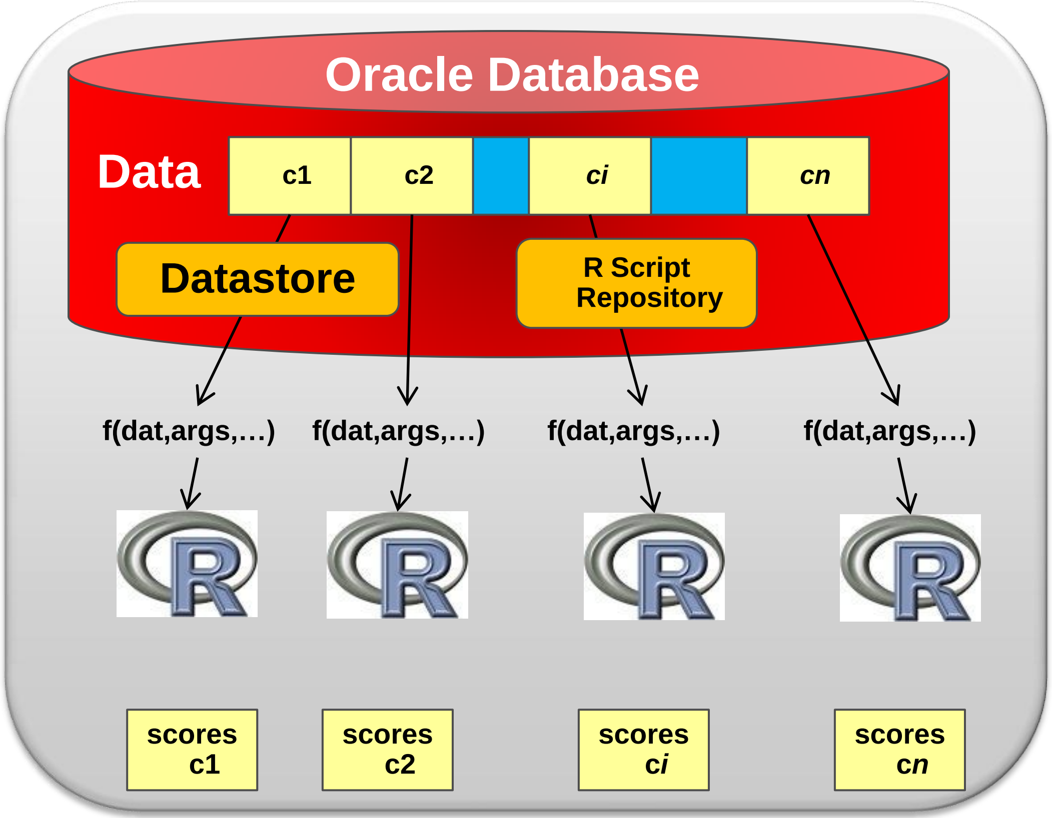
R Script
build
model

}



Database-centric architecture

Smart meter scenario



f(dat,args,…) {

R Script

score data

}

Build 200K models stored in database, partition on CUST_ID

```
ore.groupApply (CUST_USAGE_DATA,
                CUST_USAGE_DATA$CUST_ID,
    function(x, ds.name) {
      cust_id <- x$CUST_ID[1]
      mod <- lm(Consumption ~ . -CUST_ID, x)
      mod$effects <- mod$residuals <- mod$fitted.values <- NULL
      name <- paste("mod", cust_id,sep="")
      assign(name, mod)
      ds.name1 <- paste(ds.name,".",cust_id,sep="")
      ore.save(list=paste("mod",cust_id,sep=""), name=ds.name1, overwrite=TRUE)
      TRUE
    },
    ds.name="myDatastore", ore.connect=TRUE, parallel=TRUE
  )
```

14 lines

Score 200K customers in database, partition on CUST_ID

```
ore.groupApply(CUST_USAGE_DATA_NEW,
               CUST_USAGE_DATA_NEW$CUST_ID,
    function(dat, ds.name) {
        cust_id <- dat$CUST_ID[1]
        ds.name1 <- paste(ds.name,".",cust_id,sep="")
        ore.load(ds.name1)
        name <- paste("mod", cust_id,sep="")
        mod <- get(name)
        prd <- predict(mod, newdata=dat)
        prd[as.integer(rownames(prd))] <- prd
        res <- cbind(CUST_ID=cust_id, PRED = prd)
        data.frame(res)
    },
    ds.name="myDatastore", ore.connect=TRUE, parallel=TRUE,
    FUN.VALUE=data.frame(CUST_ID=numeric(0), PRED=numeric(0))
  )
```

16 lines

ORACLE

Massive Clustering Modeling

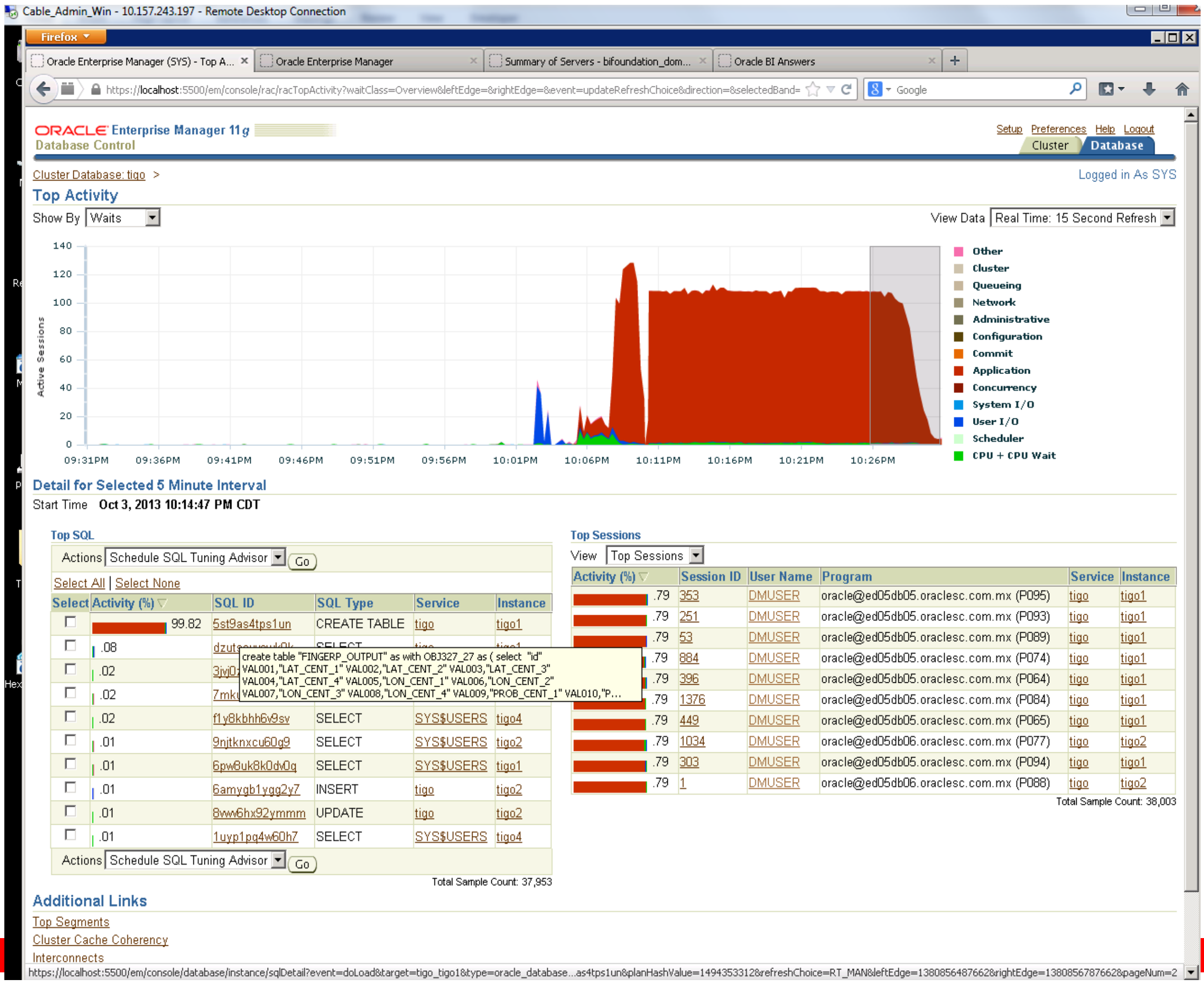
Massive Clustering Model Building

From customer POC

- Identify number of clusters (between 2 and 5) that best describes customer behavior
- Data
  - 5.64M customers with total of 1.8B transactions over past year (~320 records/customer)
- Approach
  - Execute ore.groupApply that spawns parallel computations for each customer's transactions – building multiple clustering models per customer (
  - Build 5 k-means models and basic computations to select the top transaction types and volumes for each set of customer transactions
- Return value
  - For each customer, produce columns containing centroids for clusters found to be optimal
  - Top 8 transaction types and volumes output as new columns
  - Output automatically converted to an ore.frame / table by ore.groupApply
- Timing
  - Execution built a total of 28M open-source k-means models (+ auxiliary functions) in 25.25 minutes, with a very high utilization of available hardware.

Machine Utilization

> 100 cores in use



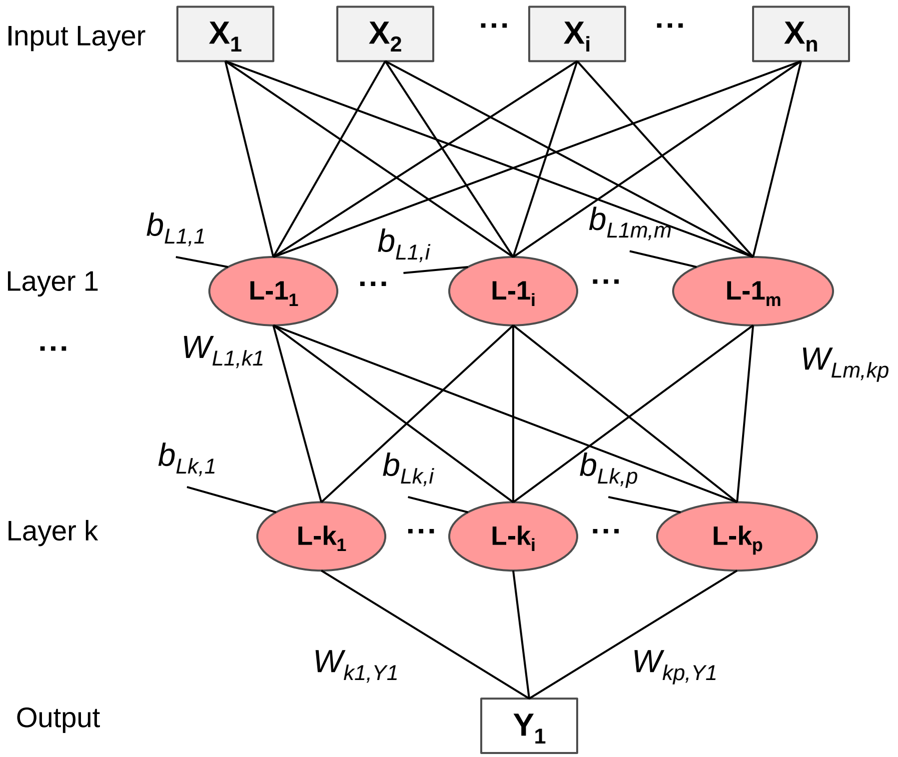
Artificial Neural Networks

Artificial Neural Networks

- Mathematical model inspired by biological neural networks in some sense mimicking the functioning of a brain
  - Consists of an interconnected group of artificial neurons (nodes)
  - Non-linear statistical data modeling tools
  - Model complex nonlinear relationships between input and output variables
- Find patterns in data
  - Function approximation: regression, including time series prediction, fitness approx, modeling
  - Classification: pattern / sequence recognition, novelty detection, sequential decision making
  - Data processing: including filtering, clustering, blind source separation and compression
  - Robotics: including directing manipulators, computer numerical control
- Applicable to neuroinformatics, neurorobotics
- **ore.neural**: L-BFGS (Limited-memory BFGS) algorithm used to solve underlying unconstrained nonlinear optimization problem
  - ore.parallel option used by ore.neural to determine preferred DOP to use within ORE server

ore.neural Architecture Specification

- Input Layer
  - Numerical or categorical
  - No automatic normalization of data
  - Supports up to 1000 actual columns (due to database table limit)
  - No fixed limit on interactions
  - No fixed limit on cardinality of categorical variables
- Hidden Layers
  - Any number of hidden layers - *k*
  - All nodes from previous layer are connected to nodes of next
  - Activation function applies to one layer
    - Bipolar Sigmoid default for hidden layers
- Output Layer
  - Currently single numeric target or binary categorical
  - Linear activation function default, all others also supported
- Calculate number of weights
  - (# input units) x (# L1 nodes) + (# L1 nodes bias) +
    (# L1 nodes) x (# L2 nodes) + (# L2 nodes bias) +
    …
    (# Lk nodes) x (# output nodes)
- Initialize weights
  - Change initialization with random seed
  - Set lower and upper bound, typically -0.25, 0.25



Unique aspects of ore.neural

- Hidden layer structure complexity
- #Activation functions - 15
- Support for categorical variables and transformations of all variables – predictors and targets
- Support for logistic regression through entropy activation function
- No competitive CRAN package available for neural networks
- Scalability on several dimensions including HYPER SPARSE data sets
  - Scale-up and Scale-out
- Works with data sets that do not fit in memory
  - SAS HPNeural requires complete data set to fit into distributed memory before it can solve any HP* models

Face Recognition

http://cbcl.mit.edu/projects/cbcl/software-datasets/FaceData1Readme.html

- Is this image a face or not?
- Data: 6,977 training, 24,045 test, 363 columns
  - 19x19 pixel image
  - From Center for Biological and Computational Learning at MIT
- 5 Neural Network Layers of size – *to explore scalability*
  - (1000L, 500L, 250L, 200L, 50L)
- T5-8 took ~10 minutes to calculate 1,048,051 weights
- 3.8% error rate on test set

- GLM produced a 8.43% error rate
  - not surprising since only 362 weights, compared to neural 1M+

Face Recognition

```
fit <- ore.neural(is_face~.,FACES_TRAIN,
                  hiddenSizes = c(1000L, 500L, 250L, 200L, 50L),
                  activations = c("tanh","bSigmoid","bSigmoid","bSigmoid","bSigmoid","sigmoid")))
```

```
user  system elapsed
  3.727   0.058 593.042
> fit$nIterations
[1] 28
> fit$nObjEvaluations
[1] 41
> fit$nThreads
[1] 1024
> fit$nUpdates
[1] 20
> fit$nWeights
[1] 1048051
> fit$solutionStatus
[1] "Optimal"
```

```
> fit
Number of input units          361
Number of output units         1
Number of hidden layers        5
Objective value                1.083448E+00
Solution status                Optimal
Hidden layer [1]               number of neurons 1000, activation 'tanh'
Hidden layer [2]               number of neurons 500, activation 'bSigmoid'
Hidden layer [3]               number of neurons 250, activation 'bSigmoid'
Hidden layer [4]               number of neurons 200, activation 'bSigmoid'
Hidden layer [5]               number of neurons 50, activation 'bSigmoid'
Output layer                   number of neurons 1, activation 'sigmoid'
Optimization solver            L-BFGS
Scale Hessian inverse          1
Number of L-BFGS updates       20
```

Densifying Sparse Data via Hadoop

Densifying Twitter Data

“Sparse” social data

Tweets about
a bank’s products

Big Data Appliance

Oracle R Advanced Analytics for Hadoop

“Dense” predictive variables

Which customers tweeted about which bank products

Oracle Database

Build predictive model for Lifetime Value

Which customers tweeted about which bank products

Tweets – using format from twitteR

"text","favorited","replyToSN","created","truncated","replyToSID","id","replyToUID","statusSource","screenName","retweetCount","retweeted","longitude","latitude"

"**Doing a great job #SavingsAlpha #BankOfOracle #SavingsBeta**",FALSE,NA,2014-01-01 00:00:00,FALSE,NA,3.430311e+17,NA,"HootSuite","MEE.COMER.CU1142",0,FALSE,NA,NA

"**Where can I get #SavingsBeta #BankOfOracle**",FALSE,NA,2014-01-01 03:40:28,FALSE,NA,3.430311e+17,NA,"Vitrue Accounts","LAURINDA.ROWLAND.CU1144",0,FALSE,NA,NA

"**I'm a fan of #BOOCD #SavingsBeta #SavingsAlpha**",FALSE,NA,2014-01-01 07:20:57,FALSE,NA,3.430311e+17,NA,"web","ANNETT.MCMULLEN.CU1145",0,FALSE,NA,NA

"**I'm a fan of #BankOfOracle #SavingsBeta #SavingsAlpha**",FALSE,NA,2014-01-01 11:01:26,FALSE,NA,3.430311e+17,NA,"TweetCaster for Android","THELMA.DELONG.CU1146",0,FALSE,NA,NA

"**Where can I get #CheckingPlusPlus**",FALSE,NA,2014-01-01 14:41:55,FALSE,NA,3.430311e+17,NA,"TweetDeck","CRISELDA.HAWKINS.CU1147",1,FALSE,NA,NA

"

Workflow

- Load tweets into HDFS
- Convert sparse tweets into dense counts of specific hash tags using ORAAH on BDA
- Move dense data to Oracle Database for processing with ORE
- Merge/join customer hash tag counts with customer data
- Build predictive model for lifetime value (LTV)
- Score new customers to identify likely to have high LTV
- Flag those customers who are currently at a lower LTV than predicted

Processed Tweets – "densified"

```
> head(tag_summary2)
  bankoforacle boacd checkingplusplus savingsalpha savingsbeta             screenname
1            3     3                1            6           2       AJA.BROOKS.CU290
2            4     0                3            1           4     ALANA.BEARD.CU12607
3            4     1                5            4           4       ALBERTO.LE.CU425
4            2     1                0            3           5   ALEIDA.RAMSEY.CU11958
5            5     0                1            1           4    ALFONSO.WOODY.CU5213
6            3     1                0            3           5 ALLEN.ELDRIDGE.CU13612
...
```

- Join with Life Time Value (LTV) data based on "screenname"

Load Tweets into R and HDFS

```
tweetsBOO.id <- hdfs.upload("/home/mh/datasets/TweetsBankOfOracle-100K.txt",
                            dfs.id="tweetsBOO",
                            header=FALSE,overwrite=TRUE,key.sep='\1',
                            value.sep=',') # use bogus key.sep for no key

hdfs.meta(tweetsBOO.id, names=c("text","favorited","replyToSN","created",
                                "truncated","replyToSID","id","replyToUID",
                                "statusSource","screenName","retweetCount",
                                "retweeted","longitude","latitude"))

hdfs.meta(tweetsBOO.id,pristine=TRUE)
hdfs.meta(tweetsBOO.id, quote='"')
```

Specify Mapper

```
mapHashTags <- function (k,v) {
  x <- strsplit(v$text, " ")
  x <- x[x!='']
  importantTags <- tolower(importantTags)
  for(twt in 1:length(x)) {
    for(tag in x[[twt]]) {
      if(substr(tag,1,1) == "#") {
        tagL <- tolower(tag)
        if(tagL %in% importantTags) {
          orch.keyval(v[twt,"screenName"],tagL)
        }}}}}
```

Specify Reducer

```
reduceHashTags <- function(screenName,tags) {
            importantTags <- tolower(importantTags)
            tags          <- factor(tags$val,levels=importantTags)
            tagCounts     <- as.data.frame(t(as.matrix(table(tags))))
            orch.keyval(screenName,tagCounts)
}
```

Invoke MapReduce Job

```
importantTags <- c("#BankOfOracle","#BOACD","#CheckingPlusPlus",
                   "#SavingsAlpha", "#SavingsBeta" )
tag.summary <- hadoop.exec(tweetsBOO.id,
                 mapper  = mapHashTags,
                 reducer = reduceHashTags,
                 export  = orch.export(importantTags=importantTags),
                 config  = new("mapred.config",
                      job.name      = "TwitterScreenNameHashTags",
                      map.tasks     = 100,
                      reduce.tasks  = 50,
                      map.output    = data.frame(key='a', val='a'),
                      reduce.output = data.frame(key='a',BankOfOracle=0,
                                                  BOACD=0, CheckingPlusPlus=0,
                                                 SavingsAlpha=0 ,SavingsBeta=0)))
hdfs.get(tag.summary)
```

Simulations

Simulation with ore.indexApply

```
simulation <- function(index, n) {
  set.seed(index)
  x <- rnorm(n)
  res <- data.frame(t(matrix(summary(x))))
  names(res) <- c("min","q1","median","mean","q3","max")
  res$id <- index
  res
}
(res <- simulation(1,1000))
```

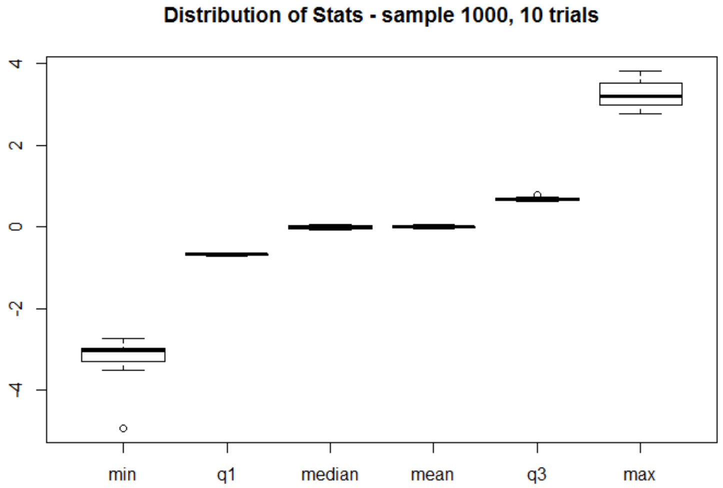
```
> (res <- simulation(1,1000))
     min      q1   median     mean     q3  max id
1 -3.008 -0.6974 -0.03532 -0.01165 0.6884 3.81  1
```

Simulation with ore.indexApply

```
stats <- ore.pull(ore.indexApply(10, simulation, n=1000,
                                 FUN.VALUE=res[1,], parallel=TRUE))
Library(reshape2)
melt.stats <- melt(stats, id.vars="id")
boxplot(value~variable, data=melt.stats,
        main="Distribution of Stats - sample 1000, 10 trials")
```

\> stats

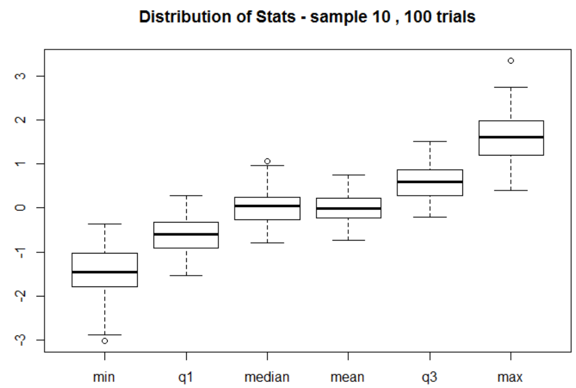
	min	q1	median	mean	q3	max	id
1	-3.498	-0.6556	0.022300	0.017400	0.6919	3.402	5
2	-3.282	-0.7268	-0.028480	-0.041260	0.6634	2.978	8
3	-3.056	-0.6845	0.032340	0.006397	0.6767	3.519	3
4	-3.008	-0.6974	-0.035320	-0.011650	0.6884	3.810	1
5	-2.722	-0.6313	0.050140	0.062000	0.7711	3.009	2
6	-3.012	-0.6774	-0.003001	0.011370	0.7275	3.541	10
7	-4.919	-0.6851	-0.069410	-0.025270	0.6484	3.238	6
8	-2.973	-0.6581	-0.022490	0.003048	0.6780	2.967	7
9	-2.840	-0.6661	-0.039790	-0.034430	0.6350	3.174	4
10	-3.041	-0.6486	0.031070	0.005885	0.6623	2.763	9

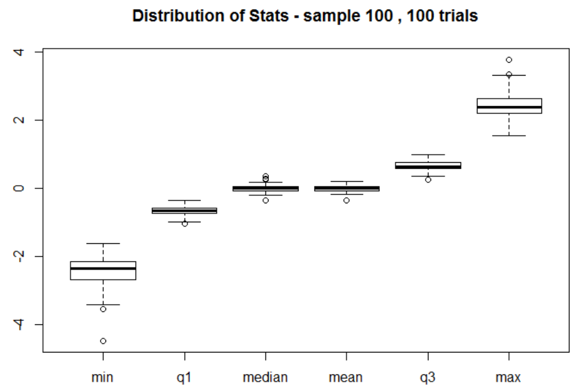


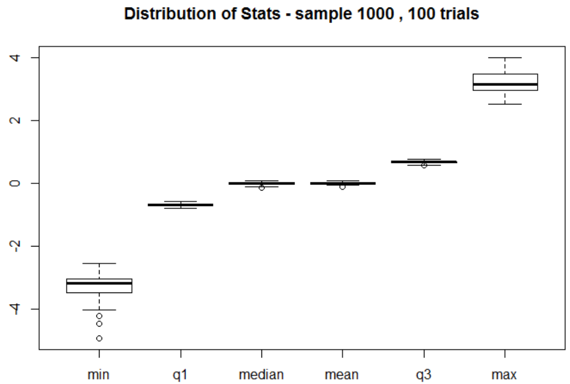
Simulation with ore.indexApply

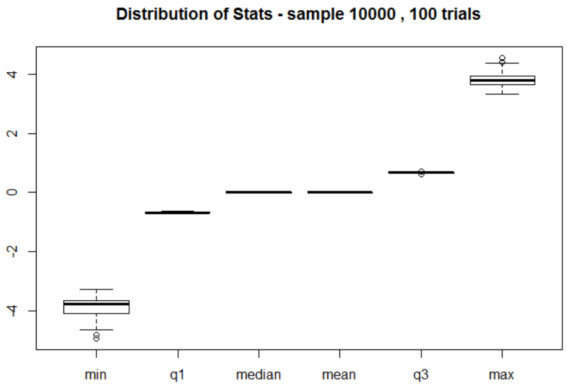
```
num.trials <- 100
for(n in 10^(1:6)){
  t1 <- system.time(stats <- ore.pull(ore.indexApply(num.trials, simulation,
  n=n,
                                 FUN.VALUE=res[1,], parallel=TRUE)))[3]
  cat("n=",n,", time=",t1,"\n")
  melt.stats <- melt(stats, id.vars="id")
  boxplot(value~variable, data=melt.stats,
          main=paste("Distribution of Stats - sample",n,",",
                     num.trials, "trials"))
  gc()
}
```

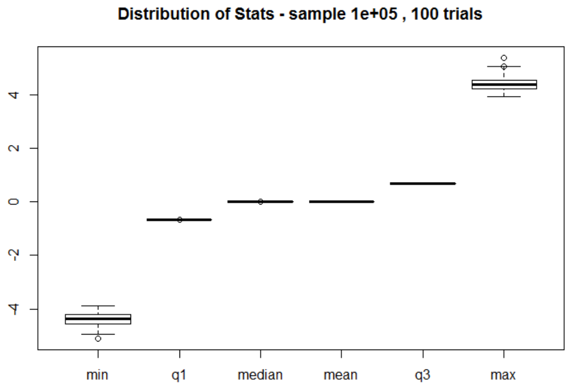
Plot Results

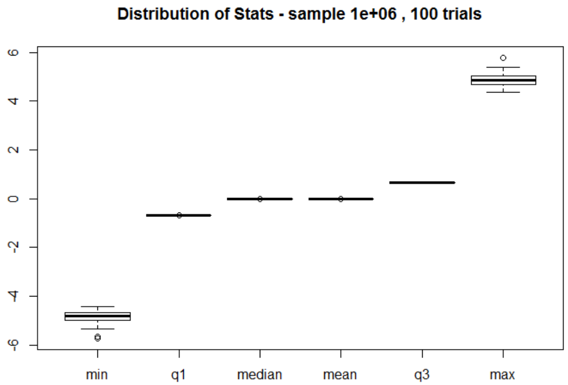












Simulation with ore.indexApply - igraph

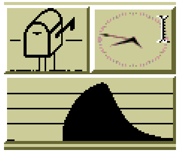
```
library(igraph)
simulation <- function(index, n, p.or.m) {
  library(igraph)
  set.seed(index)
  g <- erdos.renyi.game(n, p.or.m)
  max.clique.size <- clique.number(g)
  res <- data.frame(id = index, max_clique_size=max.clique.size)
  res
}
(res <- simulation(1, 100, 0.3))
```

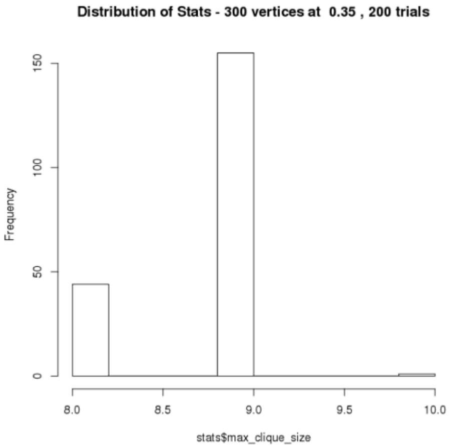
```
> (res <- simulation(1, 100, 0.3))
  id max_clique_size
1  1               6
```

Simulation with ore.indexApply - igraph

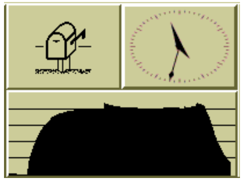
```
n <- 300
p.or.m <- 0.35
num.trials <- 200
stats <- ore.pull(ore.indexApply(num.trials, simulation,
                                 n=n, p.or.m=p.or.m,
                                 FUN.VALUE=res[1,],
                                 parallel=TRUE))

hist(stats$max_clique_size,
     main=paste("Distribution of Stats -",
     n,"vertices at ",p.or.m,",",
     num.trials,"trials"))
```

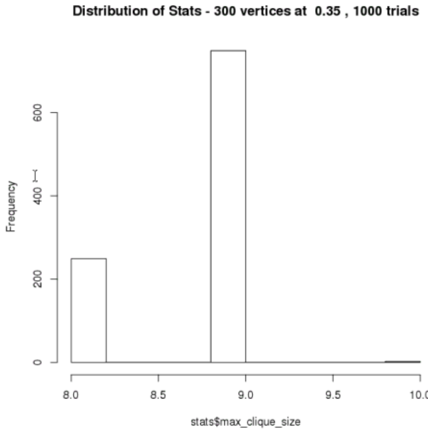
```
R> system.time(
+ stats <- ore.pull(ore.indexApply(num.trials, simulation, n=n, p.or.m=p.or.m,
+                                  FUN.VALUE=res[1,], parallel=TRUE))
+ )[3]
elapsed
169.278
```






```
R> n <- 300
R> p.or.m <- 0.35
R> num.trials <- 1000
R> system.time(
+ stats <- ore.pull(ore.indexApply(num.trials, simulation, n=n, p.or.m=p.or.m,
+                                  FUN.VALUE=res[1,], parallel=TRUE))
+ )[3]
elapsed
794.897
```





Demonstration

Resources

- **Book:** Using R to Unlock the Value of Big Data
- **Blog:** https://blogs.oracle.com/R/
- **Forum:** https://forums.oracle.com/forums/forum.jspa?forumID=1397
- **Oracle R Distribution**
- **ROracle**
  Oracle R Enterprise
- **Oracle R Advanced Analytics for Hadoop**

http://oracle.com/goto/R

ORACLE®