ORACLE

Session 5: Oracle Machine Learning for R

Machine Learning Algorithms

Mark Hornick, Senior Director
Oracle Machine Learning Product Management

November 2020

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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Copyright © 2020 Oracle and/or its affiliates.

Agenda

1. Package Overview
2. OREdm package
3. OREmodels package
4. OREpredict package
5. OREeda package

Copyright © 2020 Oracle and/or its affiliates.

OML4R Analytics Packages

List package contents:
lsf.str("package:OREdm")
ls("package:OREdm")
help(package = OREdm)

OREbase

OREdm

- Oracle Data Mining algorithms exposed through R interface
- Attribute Importance, Decision Trees, GLM, KMeans, O-Cluster, Naïve Bayes, SVD, SVM, NMF, Association Rules, Explicit Semantic Analysis

OREeda

- Functions for exploratory data analysis for Base SAS equivalents

OREgraphics

OREmodels

- ore.lm, ore.stepwise, ore.neural, ore.glm, ore.randomForest

OREpredict

- Score R models in the database

OREstats

- In-database statistical computations exposed through R interface

OREXml

Copyright © 2020 Oracle and/or its affiliates.

High performance in-database ML algorithms

OREdm

Support Vector Machine
GLM
Naïve Bayes
Decision Trees
k-Means clustering
O-Cluster clustering
Expectation Maximization
Explicit Semantic Analysis
Singular Value Decomposition
Association Rules
Attribute Importance

OREmodels

Random Forest

Principal Component Analysis (overloaded)

Singular Value Decomposition (overloaded)

Neural Networks

Linear Regression

Stepwise Regression

Generalized Linear Model

Copyright © 2020 Oracle and/or its affiliates.

OREdm Package

Copyright © 2020 Oracle and/or its affiliates.

OREdm Features

Function signatures conform to R norms
- Use formula for specifying target and predictor variables
- Use ore.frame objects as input data set for build and predict
- Creates R objects for models and ore.frames for prediction results
- Use parameter names similar to corresponding R functions
- Function parameters provide explicit default values to corresponding ODM settings, where applicable

As in R, models are treated as transient objects
- Automatically delete ODM model when corresponding R object no longer exists
- Can be explicitly saved using datastore, via ore.save

Implicit variable selection for specific models

Automatic data preparation available

Supports *partitioned* models based on values of one or more columns

Enables text column analytics for select algorithms

Copyright © 2020 Oracle and/or its affiliates.

Algorithms supporting Implicit Variable Selection

Decision tree performs automatic variable selection as part of the building process itself

- Variables that are not used in the tree are effectively removed
- Takes into account the relationship of the variables with the target

Naïve Bayes performs automatic variable selection when ADP is enabled

- Takes into account relationship of each variable with target

GLM performs variable selection (and creation) when the corresponding settings for feature selection/creation are used

Copyright © 2020 Oracle and/or its affiliates.

Automatic Data Preparation (ADP)

Automatic variable transformation is handled by auto data preparation for OREdm algorithms

- Auto data preparation takes into account the algorithm and data characteristics to prepare data
- Each algorithm may have different preparation requirements

Binning: ADP for Naïve Bayes and Decision Tree use the supervised binning transformation in the dbms_data_mining_transform package to generate bins prior to model building (that take into account the target)

Normalization: ADP for SVM and GLM uses normalization transformations in the dbms_data_mining_transform package to generate a variety of normalization parameters prior to model building

Simplest approach - turn on ADP when building a model and inspect results after

- If user needs more control over preparation stages before model building, transform the data explicitly using OML4R transparency layer

Copyright © 2020 Oracle and/or its affiliates.

Partition Models

Automates a typical machine learning task for data scientists

Builds an ensemble model composed of multiple sub-models, one built for each partition of data

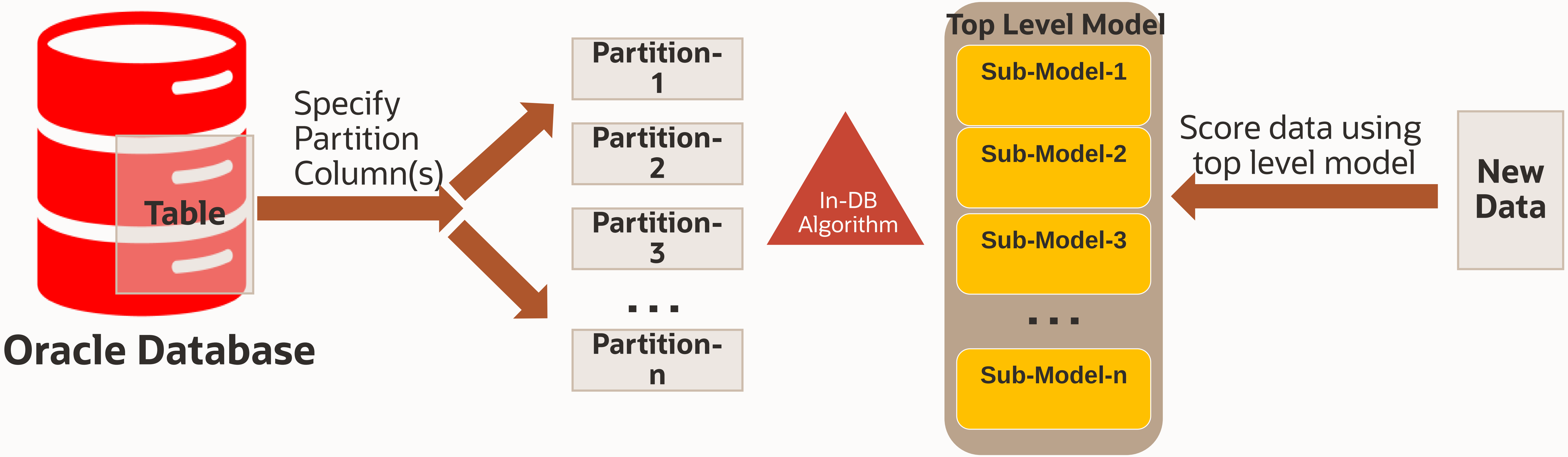
- Potentially achieve better accuracy through multiple targeted models – managed and used as one

Set parameter `ODMS_PARTITION_COLUMNS` to the name(s) of the partition column(s)

- For example, `odm.setting = list(odms_partition_columns = "part")`

Simplifies scoring by allowing user to provide the top level model only

- Proper sub-model chosen by system based on row of data to be scored



Copyright © 2020 Oracle and/or its affiliates.

Extensible R Algorithm Models

Creates an *Extensible R Algorithm* model using Oracle Data Mining in Oracle Database 12.2 or later

Extensible R Algorithm enables build, score, and view of R model using the user-provided R scripts stored in R Script Repository

Supports classification, regression, clustering, feature_extraction, attribute_importance, and association mining functions

Predict method executes the *score.function* specified for the model build and returns an ore.frame containing the predictions along with the columns specified by the *supplemental.cols* argument

- Function *predict* applicable to classification, regression, clustering, and feature_extraction models, only

Copyright © 2020 Oracle and/or its affiliates.

ore.odmRAlg

Extensible R Algorithm Models

```
ore.odmRAlg(data,
         mining.function = c("classification", "regression", "clustering",
                   "feature_extraction", "attribute_importance", "association"),
         formula = NULL,
         build.function,
         build.parameter = NULL,
         score.function = NULL,
         detail.function = NULL,
         detail.value = NULL,
         odm.setting = NULL)

predict(object, newdata, supplemental.cols = NULL,
        type = c("class","raw"), na.action = na.pass,...)

summary(object,...)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

data – ore.frame object used for model building

mining.function – A scalar string to specify the type of mining function: classification, regression, clustering, feature_extraction, attribute_importance, and association

formula – An R formula object or a string representing a formula in characters. This formula can be named or take the default name 'formula'. This name is used to pass the specified formula to the R *build.function*. If formula is NULL, the user-specified R *build.function* does not take a formula.

build.function – The name of the user-defined R function in the R Script Repository used to build the model. The R function uses the first argument for input data, optionally the second argument for weight numeric vector when parameter *odms_row_weight_column_name* is specified in *odm.setting*, and matches the remaining arguments by name with the values from *build.parameter*. The R function returns an R model.

build.parameter – A list containing *build.function* parameters excluding input data and weight vector if applicable. The list element names must match the name of *build.function* script input parameter names. Only scalar numeric and character values are valid as parameters.

score.function– The name of the user-defined R function in the R Script Repository used to score the model. The script takes two arguments: model and new data. It returns a *data.frame* containing prediction results. For regression, the results are predicted values. In classification, clustering, and feature exaction, the results are probabilities for each class, cluster, and feature, respectively. Rows of the results match the rows of input data.

detail.function – The name of the user-defined R function in the R Script Repository used to obtain model details and return them in a data.frame

detail.value – A data.frame object used to specify the data types of columns in the returned *data.frame* from the *detail.function*

odm.setting – A list to specify Oracle Data Mining parameter settings. This argument is applicable to building models in Database 12.2 or later. Each list element's name/value refers to the parameter setting name/value. The setting value must have type numeric or character. When parameter *ODMS_PARTITION_COLUMNS* is set to the names of the partition columns, a partition model is created from the input data.

Copyright © 2020 Oracle and/or its affiliates.

ore.odmRAlg – model object

ore.odmRAlg object

• **name**	name of model in database
• **mining.function**	The type of the data mining function for the model.
• **details**	An ore.frame returned by the R detail.function script
• **settings**	data.frame with settings used to build model
• **attributes**	data.frame of variable/columns used to build model
• **formula**	formula used to build the model
• **call**	specific invocation of the function with arguments

Copyright © 2020 Oracle and/or its affiliates.

ore.odmRAlg

Extensible R Algorithm Models

```
IRIS <- ore.push(iris)
ore.scriptCreate("glm_build", function(data, form, family) {
                                 glm(formula = form, data = data, family = family)})
ore.scriptCreate("glm_score", function(mod, data) {
                                  res <- predict(mod, newdata = data); data.frame(res)})
ore.scriptCreate("glm_detail", function(mod) {
                                  data.frame(name=names(mod$coefficients), coef=mod$coefficients)})

ralg.mod <- ore.odmRAlg(IRIS, mining.function = "regression",
                        formula = c(form="Sepal.Length ~ ."),
                        build.function = "glm_build", build.parameter = list(family="gaussian"),
                        score.function = "glm_score",
                        detail.function = "glm_detail", detail.value = data.frame(name="a", coef=1))
summary(ralg.mod)
ralg.mod$details

predict(ralg.mod, newdata = head(IRIS), supplemental.cols = "Sepal.Length")
```

Copyright © 2020 Oracle and/or its affiliates.

OREdm Algorithms

Algorithm	Main R Function	Mining Type / Function
Association Rules	ore.odmAssocRules	Association Rules
Minimum Description Length	ore.odmAI	Attribute Importance for Classification or Regression
Decision Tree	ore.odmDT	Classification
Expectation Maximization (12.2)	ore.odmEM	Clustering
Explicit Semantic Analysis (12.2)	ore.odmESA	Feature Extraction
Generalized Linear Models	ore.odmGLM	Classification Regression
K-Means	ore.odmKMeans	Clustering
Naïve Bayes	ore.odmNB	Classification
Non-negative Matrix Factorization	ore.odmNFM	Feature Extraction
Orthogonal Partitioning	ore.odmOC	Clustering
Singular Value Decomposition	ore.odmSVD	Feature Extraction
Support Vector Machine	ore.odmSVM	Classification Regression Anomaly Detection

Copyright © 2020 Oracle and/or its affiliates.

[Doc link]

Attribute Importance

Compute the relative importance of predictor variables for predicting a response (target) variable

Gain insight into relevance of variables to guide manual variable selection or reduction, with the goal to reduce predictive model build time and/or improve model accuracy

Attribute Importance uses a Minimum Description Length (MDL) based algorithm that ranks the relative importance of predictor variables in predicting a specified response (target) variable

Pairwise only – each predictor with the target

Supports categorical target (classification) and numeric target (regression)

Copyright © 2020 Oracle and/or its affiliates.

ore.odmAI

Attribute Importance

```
ore.odmAI(
  formula,                    # formula specifying attributes for model build
  data,                       # ore.frame of the training dataset
  auto.data.prep = TRUE,      # Setting to perform automatic data preparation
  na.action = na.pass,        # Allows missing values (na.pass), or removes rows with
                              #    missing values (na.omit)
  odm.setting = NULL)         # A list to specify Oracle Data Mining parameter settings
)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

formula

- Form `response ~ terms` where 'response' is the numeric or character response vector and 'terms' is a series of terms, i.e., column names, to include in the analysis
- Multiple terms are specified using '+' between column names
- Use `response ~ .` if all columns in 'data' should be used for model building. Functions can be applied to 'response' and 'terms' to realize transformations. To exclude columns, use '-' before each column name to exclude.

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

auto.data.prep

- If TRUE, automatically performs the data transformations required by the algorithm
- Transformation instructions are embedded in the in-database model

Types of transformations

- Binning
  - reduces cardinality of continuous and discrete data
  - improve resource utilization and model build response time dramatically without significant loss in model quality
  - can improve model quality by strengthening relationships between attributes
- Normalization
  - reduces range of numerical data, e.g., between 0 and 1

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

na.action

- By default, allows missing values ('na.pass'), or removes rows with missing values ('na.omit')

odm.setting

- Use to build a *partition model*
- Set parameter `ODMS_PARTITION_COLUMNS` to the name(s) of the partition column(s)
- For example, `odm.setting = list(odms_partition_columns = "part")`

Copyright © 2020 Oracle and/or its affiliates.

ore.odmAI - Example

Attribute Importance

```
LONGLEY <- ore.push(longley)
head(LONGLEY)
ore.odmAI(Employed ~ ., LONGLEY)

STATE <-
   ore.push(as.data.frame(state.x77))
head(STATE)
ore.odmAI(Murder ~ ., STATE)
```

```
R> LONGLEY <- ore.push(longley)
R> head(LONGLEY)
     GNP.deflator     GNP Unemployed Armed.Forces Population Year Employed
1947         83.0 234.289      235.6        159.0    107.608 1947   60.323
1948         88.5 259.426      232.5        145.6    108.632 1948   61.122
1949         88.2 258.054      368.2        161.6    109.773 1949   60.171
1950         89.5 284.599      335.1        165.0    110.929 1950   61.187
1951         96.2 328.975      209.9        309.9    112.075 1951   63.221
1952         98.1 346.999      193.2        359.4    113.270 1952   63.639
R> ore.odmAI(Employed ~ ., LONGLEY)

Call:
ore.odmAI(formula = Employed ~ ., data = LONGLEY)

Importance:
             importance rank
Year          0.4901166    1
Population    0.4901166    1
GNP           0.4901166    1
GNP.deflator  0.4901166    1
Armed.Forces  0.3648186    2
Unemployed    0.1318046    3
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmAI - Example

Attribute Importance

```
R> STATE <- ore.push(as.data.frame(state.x77))
R> head(STATE)
           Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
California      21198   5114        1.1    71.71   10.3    62.6    20 156361
Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
R> ore.odmAI(Murder ~ ., STATE)

Call:
ore.odmAI(formula = Murder ~ ., data = STATE)

Importance:
            importance rank
Life Exp    0.10872845    1
HS Grad     0.06915643    2
Illiteracy  0.05760828    3
Frost       0.05051389    4
Area       -0.04538736    5
Income     -0.06720964    6
Population -0.12554537    7
```

Copyright © 2020 Oracle and/or its affiliates.

Attribute Importance - results

importance

- Relative metric indicating how much the variable contributes to predicting the target
- Values > 0 contribute to prediction
- Values <= do not contribute or add noise

rank

- Ordering of variables / attributes from most significant to least

Copyright © 2020 Oracle and/or its affiliates.

Naïve Bayes

Doc link

Classification algorithm – simple probabilistic classifier

Relies on Bayes' theorem $$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

Assumes independence of predictors

- May not be the case, but works well in practice

Conditional probabilities between each predictor and target multiplied to obtain prediction

Copyright © 2020 Oracle and/or its affiliates.

ore.odmNB & predict.ore.odmNB

Naïve Bayes

```
ore.odmNB(
  formula,                    # formula specifying attributes for model build
  data,                       # ore.frame of the training dataset
  auto.data.prep = TRUE,      # Setting to perform automatic data preparation
  class.priors = NULL,        # Numeric vector with named elements for target class priors
  na.action = na.pass,        # Allows missing values (na.pass), or removes rows with
                              #    missing values (na.omit)
  odm.setting = NULL)         # A list to specify Oracle Data Mining parameter settings

predict(
  object,                     # Object of type "ore.naiveBayes"
  newdata,                    # Data used for scoring
  supplemental.cols = NULL,   # Columns to retain in output
  type = c("class","raw"),    # "raw" – cond. a-posterior probs for each class returned
                              # "class" - class with max prob
  na.action = na.pass)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

class.priors

- Optional user-specified priors for the target classes
- Specifying prior probabilities offsets distribution differences between training data and real population (scoring data)

Use when one target value dominates in frequency

- For example
  - telephone marketing campaign positive responses may be < 2%
  - occurrence of fraud in credit card transactions may be < 1%.
- A classification model built with so few positive cases may not be able to distinguish characteristics of the two classes, resulting in a model that predicts the frequent class every time → use stratified sampling to balance the data set and set priors
- Such models may be accurate, but not be very useful
- Do not rely solely on accuracy when judging the quality of a classification model

Stratified sampling and anomaly detection are alternatives to compensating for data distribution issues

Copyright © 2020 Oracle and/or its affiliates.

ore.odmNB – Example

Naïve Bayes

Login to database for transparent access via OML4R

```
library(ORE)
ore.connect("rquser","orcl","localhost","rquser",all=TRUE)

data(titanic3,package="PASWR")
```

Push data to db for transparent access

```
t3 <- ore.push(titanic3)
t3$survived <- ifelse(t3$survived == 1, "Yes", "No")
```

Recode column from 0/1 to No/Yes keeping data in database

```
n.rows <- nrow(t3)
set.seed(seed=6218945)
random.sample <- sample(1:n.rows, ceiling(n.rows/2))
t3.train <- t3[random.sample,]
t3.test  <- t3[setdiff(1:n.rows,random.sample),]
```

Sample keeping data in database

Create priors for model building

```
priors <- c(Yes=0.1, No=0.9)

nb  <- ore.odmNB(survived ~ pclass+sex+age+fare+embarked,
                 t3.train, class.priors=priors)
```

Build model using R formula with transparency layer data

Score data using ore.frame with OREdm model object.

```
nb.res  <- predict (nb, t3.test,"survived")
```

Display first 10 rows of data frame using transparency layer

```
head(nb.res,10)
with(nb.res, table(survived,PREDICTION, dnn = c("Actual","Predicted")))
```

Compute confusion matrix using transparency layer

```
library(verification)
res <- ore.pull(nb.res)
perf.auc <- roc.area(ifelse(res$survived == "Yes", 1, 0), res$"'Yes'")
auc.roc <- signif(perf.auc$A, digits=3)
auc.roc.p <- signif(perf.auc$p.value, digits=3)
roc.plot(ifelse(res$survived == "Yes", 1, 0), res$"'Yes'", binormal=T, plot="both",
         xlab="False Positive Rate",
         ylab="True Postive Rate", main= "Titanic survival ODM NB model ROC Curve")
text(0.7, 0.4, labels= paste("AUC ROC:", signif(perf.auc$A, digits=3)))
text(0.7, 0.3, labels= paste("p-value:", signif(perf.auc$p.value, digits=3)))

summary(nb)
ore.disconnect()
```

Retrieve result from database for using verification package

View model object summary

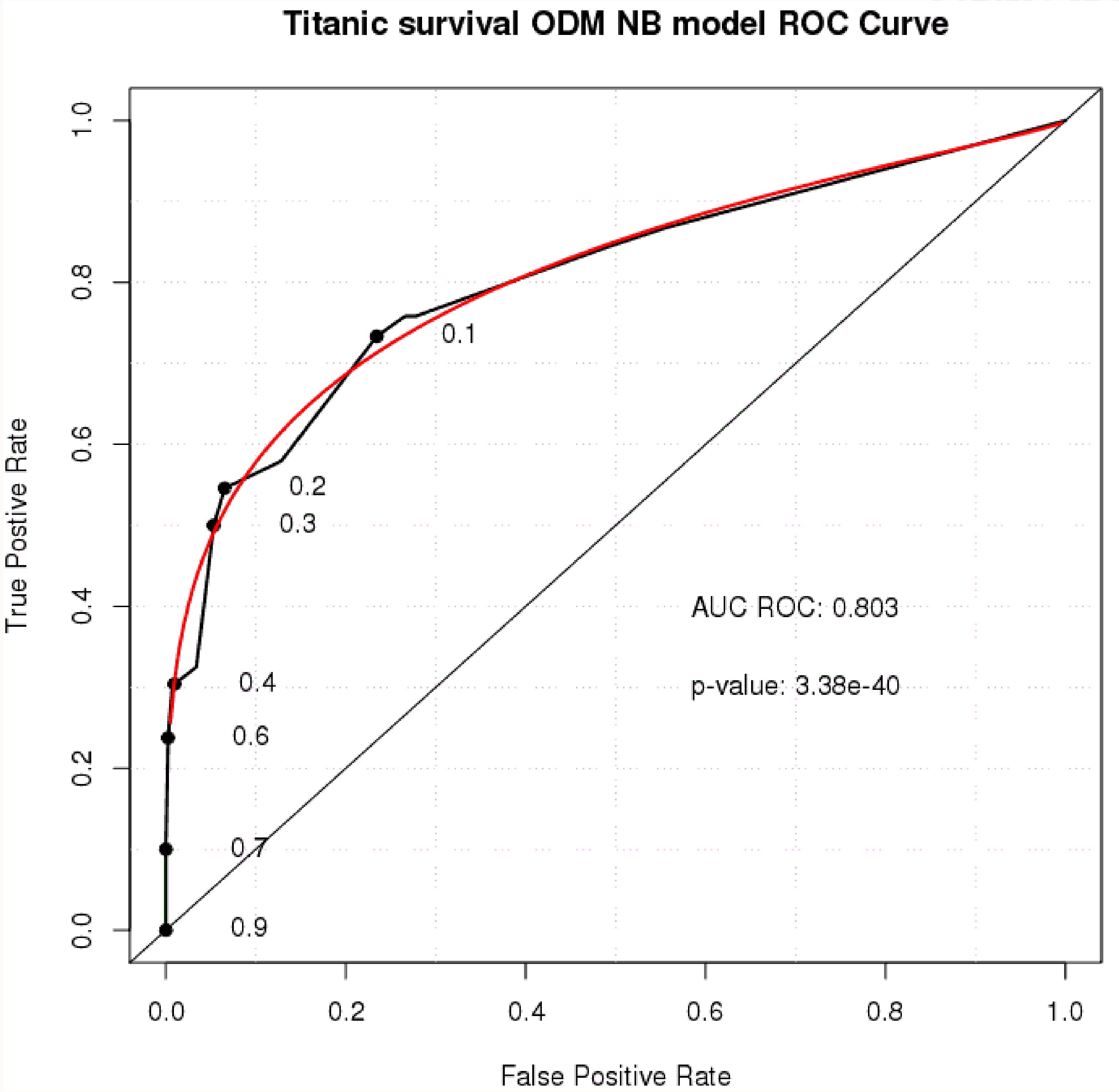
Disconnect from database

Model, train and test objects are automatically removed when session ends or R objects are removed

Copyright © 2020 Oracle and/or its affiliates.

ROC Curve

```
R> summary(nb)

Call:
ore.odmNB(formula = survived ~ pclass + sex + age + fare + embarked,
    data = t3.train, class.priors = priors)

Settings:
          value
prep.auto    on

Apriori:
 No Yes
0.9 0.1

Tables:
$embarked
    'Cherbourg' 'Queenstown', 'Southampton'
No     0.1569620                 0.8430380
Yes    0.3178295                 0.6821705

$fare
    ( ; 51.931249600000001), [51.931249600000001; 51.931249600000001] (51.931249600000001;  )
No                                                         0.91370558                0.08629442
Yes                                                        0.67307692                0.32692308

$pclass
    '1st', '2nd'     '3rd'
No     0.3417722 0.6582278
Yes    0.6346154 0.3653846

$sex
       female      male
No  0.1670886 0.8329114
Yes 0.6769231 0.3230769


Levels:
[1] "No"  "Yes"
```



Copyright © 2020 Oracle and/or its affiliates.

Naïve Bayes – model object

ore.odmNB object

- **name** of the model
- **settings** used to build the model
- **attributes** used to build the model: name, type (numerical or categorical), data type, data length (size), precision and scale for numeric data, and whether the variable is the target
- **apriori** table with class distribution for the dependent variable
- **tables** is a list with one for each predictor variable with conditional probabilities
- **levels** is a vector of unique target class values

Copyright © 2020 Oracle and/or its affiliates.

Support Vector Machine

Doc link

Suite of algorithms, adaptable for use with a variety of problems and data

By swapping one *kernel* for another, SVM can fit diverse problem spaces

Concept

- Data records with N attributes can be thought of as points in N-dimensional space
- SVM attempts to separate the points into subsets with homogeneous target values, by hyperplanes in the linear case, and in the non-linear case (Gaussian) by non-linear separators
- SVM finds the vectors that define the separators giving the widest separation of classes (the “support vectors”).

SVM solves regression problems by defining an N-dimensional “tube” around the data points, determining the vectors giving the widest separation

SVM can emulate some traditional methods, such as linear regression and neural networks, but goes far beyond those methods in flexibility, scalability, and speed

- For example, SVM can act like a neural net in calculating predictions, while a neural net might mistake a local change in direction as a point of minimum error, SVM works to find the global point of minimum error

Copyright © 2020 Oracle and/or its affiliates.

ore.odmSVM

Support Vector Machine

```
ore.odmSVM(
  formula,                                # specifies attributes for model build
  data,                                   # ore.frame containing the training dataset
  mining.function,                        # Type of model: "classification", "regression " or
"anomaly.detection"
  auto.data.prep = TRUE,                  # Setting to perform automatic data preparation
  class.priors = NULL,                    # Data frame containing target class priors
  active.learning = TRUE,                 # Setting for enabling active learning
  complexity.factor = "system.determined",# Setting for complexity factor for SVM
  conv.tolerance = 0.0001,                # Setting for convergence tolerance for SVM
  epsilon = "system.determined",          # Setting for epsilon for SVM Regression
  kernel.function = "system.determined",# Setting for kernel function (SVMS_GAUSSIAN or SVMS_LINEAR)
  std.dev = "system.determined",          # Setting for standard deviation for SVM Gaussian kernel
  outlier.rate = 0.1,                     # Setting for desired rate of outliers in dataset (1class SVM)
  na.action = na.pass                     # Allow missing values in rows by default, or na.omit
  odm.setting = NULL,                     # A list to specify ODM parameter settings
  ctx.setting = NULL                      # A list to specify Oracle Text attribute-specific settings
)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

class.priors

active.learning - enabled by default

- optimization method to control model growth and reduce model build time
- without active learning, SVM models grow as the size of the build data set increases, which effectively limits SVM models to small and medium size training sets (less than 100,000 cases
- with active learning, SVM models can be built on very large training sets.
- active learning forces the SVM algorithm to restrict learning to the most informative training examples and not to attempt to use the entire body of data. In most cases, the resulting models have predictive accuracy comparable to that of a standard (exact) SVM model
- active learning provides a significant improvement in both linear and Gaussian SVM models, whether for classification, regression, or anomaly detection. However, active learning is especially advantageous for the Gaussian kernel, because nonlinear models can otherwise grow to be very large and can place considerable demands on memory and other system resources

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

complexity.factor

- regularization setting that balances complexity of the model against model robustness to achieve good generalization on new data
- data-driven approach to automatically determine the complexity factor

conv.tolerance

- convergence tolerance criterion for completing the model training process, default .001

epsilon

- regularization setting for regression, similar to complexity factor
- specifies the allowable residuals, or noise, in the data

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

kernel.function – linear or Gaussian

- a kernel is a function that transforms the input data to a high-dimensional space where the problem is solved. Kernel functions can be linear or nonlinear.
- algorithm automatically uses the kernel function that is most appropriate to the data if not specified
- linear kernel when # attributes > 100 in training data, else Gaussian kernel
  - # attributes reflects categorical columns exploded to numeric attributes

kernel.cache.size

- memory allocated to Gaussian kernel cache maintained in memory to improve model build time, default 50 MB

std.dev

- controls spread of Gaussian kernel function

outlier.rate

- for anomaly detection
- expected outlier rate in anomaly detection, default 0.1

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

odm.setting – A list to specify Oracle Data Mining parameter settings. This argument is applicable to building a model in Database 12.2 or later. Each list element's name and value refer to the parameter setting name and value, respectively. The setting values must be numeric or string. To perform text mining, parameter `ODMS_TEXT_POLICY_NAME` must be set to a text policy name. When parameter `ODMS_PARTITION_COLUMNS` is set to the name(s) of the partition column(s), a partition model with a sub-model in each partition is created from the input data.

ctx.setting – A list to specify Oracle Text attribute-specific settings. This argument is applicable to building model in Database 12.2 or later. The name of each list element refers to the text column while the list value specifies the text transformation.

(See ODM documentation for specific settings options.)

Copyright © 2020 Oracle and/or its affiliates.

predict.ore.odmSVM

Support Vector Machine

```
predict (
  object,
  newdata,
  supplemental.cols = NULL, # Columns to retain in the output
  type = c("class","raw"),  # "raw" - cond. a-posterior probs for each class returned,
                            #   else class with max prob (TBD for compaitbility with e1071)
  na.action = na.pass,...)  # allow missing vlaues in rows by default, or na.omit
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

supplemental.cols

- Columns from **`newdata`** to include as columns in the **`ore.frame`** prediction result
- Use to include specific columns in the prediction result for easier analysis

type = c("class","raw"), if a classification model…

- "raw" provides probability for each class returned
- "class" returns the class with the maximum probability
- default c("class","raw") returns both

Copyright © 2020 Oracle and/or its affiliates.

ore.odmSVM – Example

Support Vector Machine

```
x <- seq(0.1, 5, by = 0.02)
y <- log(x) + rnorm(x, sd = 0.2)
dat <-ore.push(data.frame(x=x, y=y))

# Regression
svm.mod <- ore.odmSVM(y~x,dat,"regression", kernel.function="linear")
summary(svm.mod)
coef(svm.mod)
svm.res <- predict(svm.mod,dat,supplemental.cols="x")
head(svm.res,6)
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmSVM – Example

Support Vector Machine

```
# Set up data set
m <- mtcars
m$gear <- as.factor(m$gear)
m$cyl  <- as.factor(m$cyl)
m$vs   <- as.factor(m$vs)
m$ID   <- 1:nrow(m)
MTCARS <- ore.push(m)
```

```
# Classification
svm.mod  <- ore.odmSVM(gear ~ .-ID,
                       MTCARS,"classification",
            kernel.function="linear")
summary(svm.mod)
coef(svm.mod)
svm.res  <- predict (svm.mod, MTCARS,"gear")
head(svm.res)
svm.res  <- predict (svm.mod, MTCARS,"gear",type="raw")
head(svm.res)
svm.res  <- predict (svm.mod, MTCARS,"gear",type="class")
head(svm.res)
with(svm.res, table(gear,PREDICTION))  # confusion matrix

# Anomaly Detection
svm.mod  <- ore.odmSVM(~ .-ID, MTCARS,"anomaly.detection",
            kernel.function="system.determined")
summary(svm.mod)
svm.res  <- predict (svm.mod, MTCARS, "ID")
head(svm.res)
table(svm.res$PREDICTION)
```

Copyright © 2020 Oracle and/or its affiliates.

SVM – model object

ore.odmSVM object

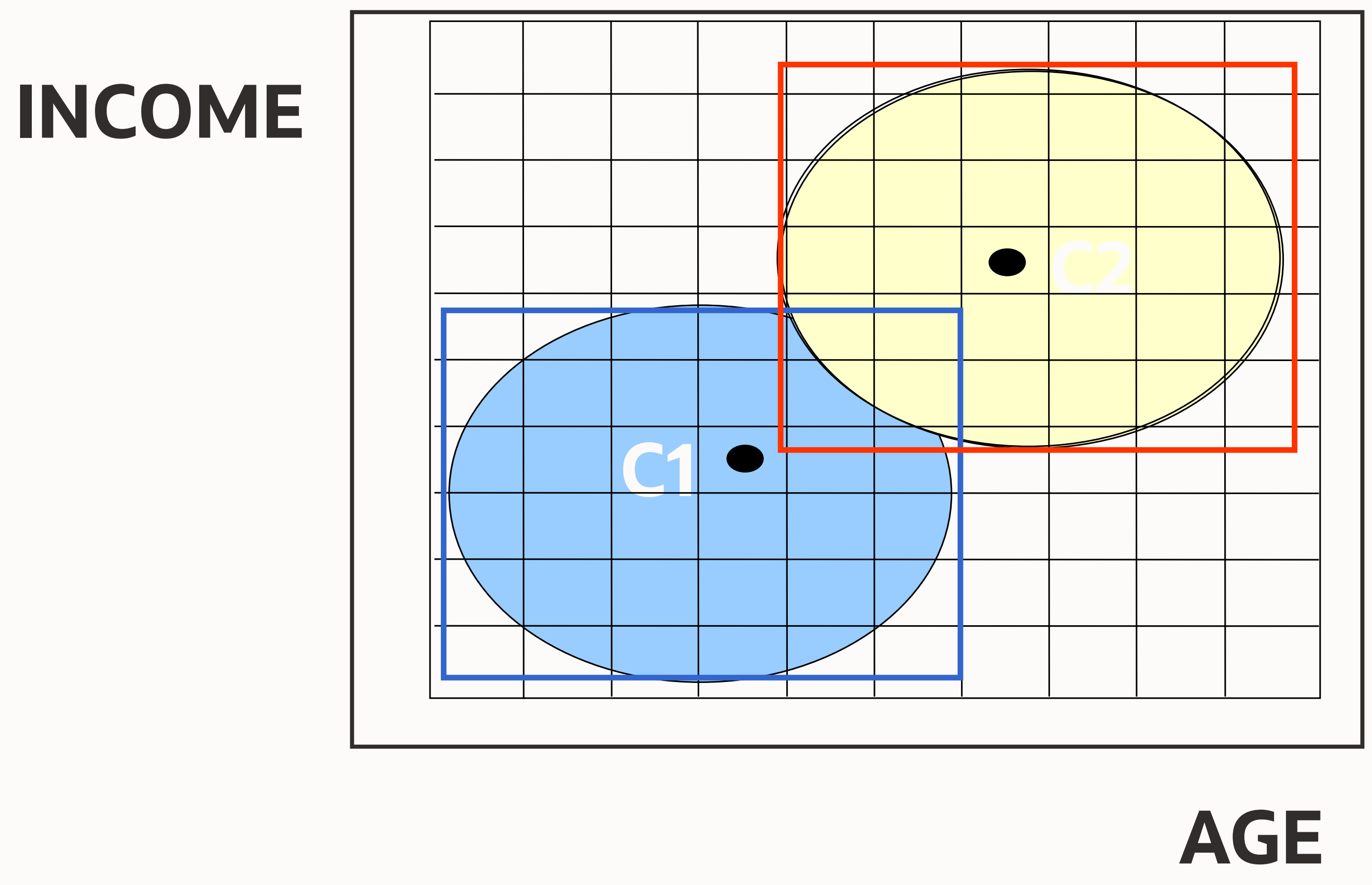
- **name** of the model
- **settings** used to build the model
- **attributes** used to build the model: name, type (numerical or categorical), data type, data length (size), precision and scale for numeric data, and whether the variable is the target
- **fit.values** is an ore.frame of the actual column and predicted column. For regression, the columns are 'ACTUAL' and 'PREDICTED'. For classification, the columns are 'ACTUAL','PREDICTED','PROBABILITY'. For anomaly detection, the columns are 'PREDICTED' and 'PROBABILITY'.
- **residuals** for regression models, an ore.numeric vector containing the residual values (PREDICTED - ACTUAL).
- **formula** is the symbolic description of the model fitted
- **call** is the invocation parameters of the function

If built with a linear kernel, the following are also returned

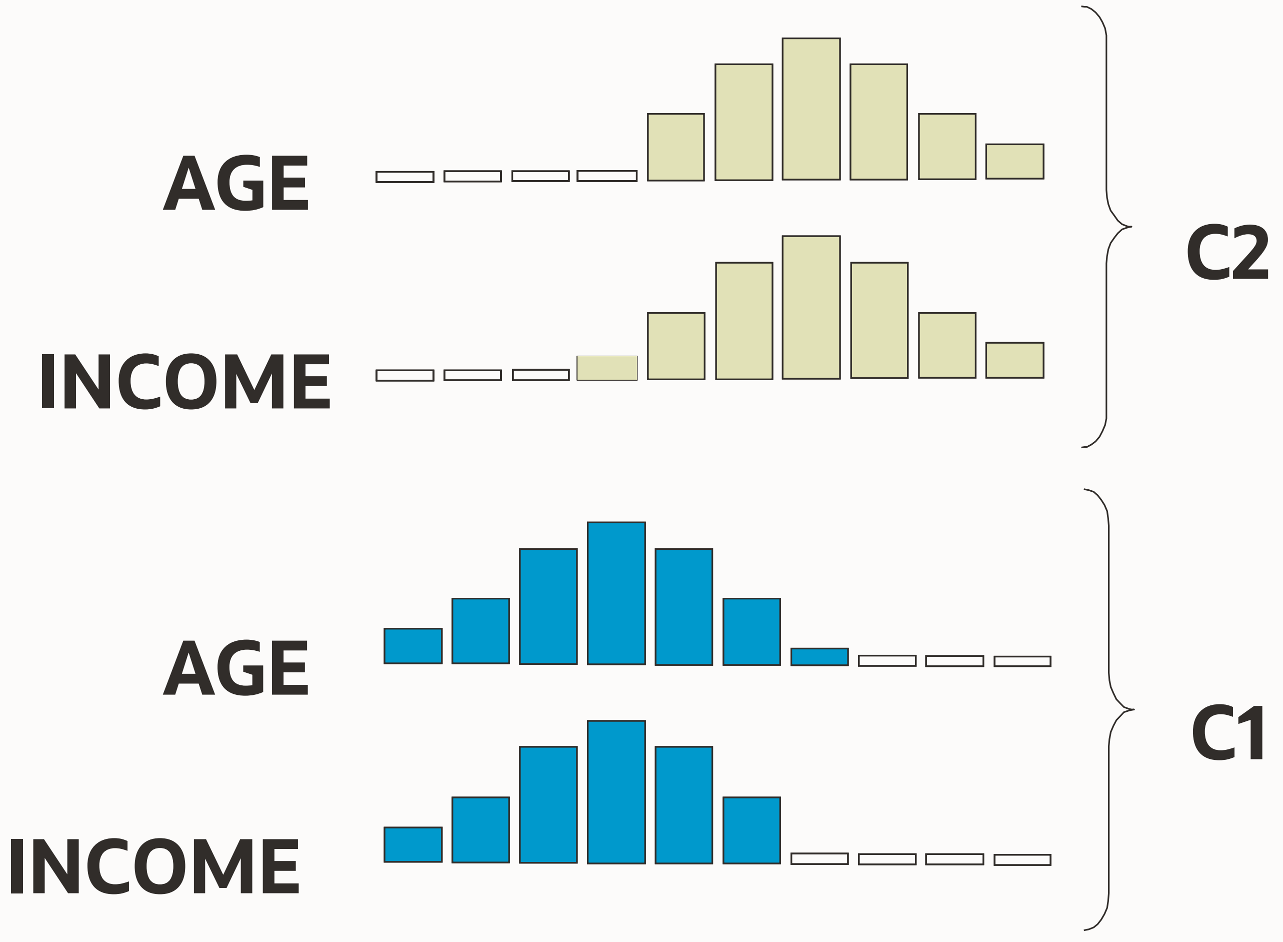
- **coefficients** of the SVM model, one for each predictor variable. If auto.data.prep is set to TRUE, these coefficients will be in the transformed space (after automatic outlier-aware normalization is applied)

Copyright © 2020 Oracle and/or its affiliates.

Cluster Description



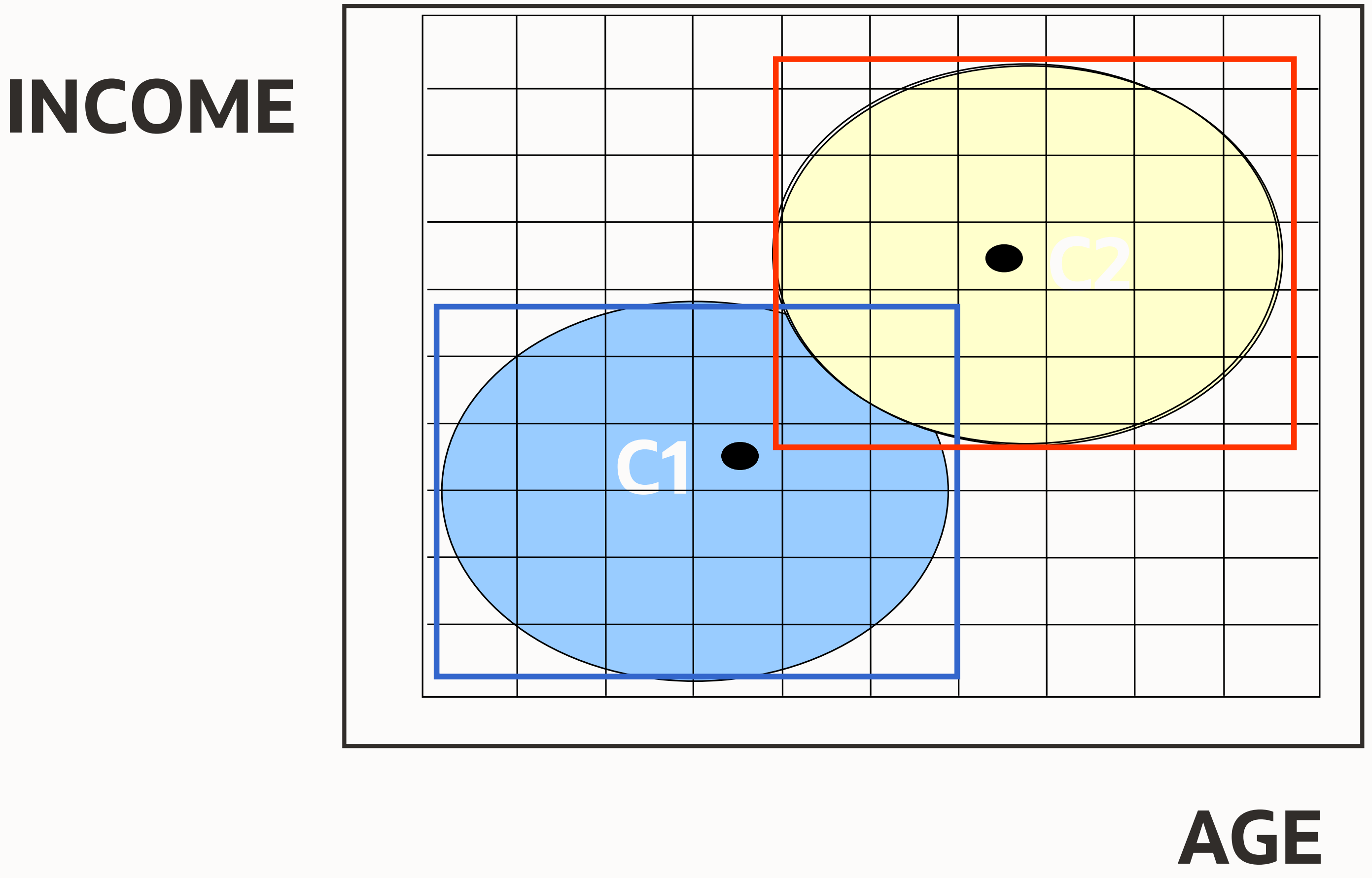
Centroids



Histograms

Copyright © 2020 Oracle and/or its affiliates.

Cluster Rules



Cluster 1:

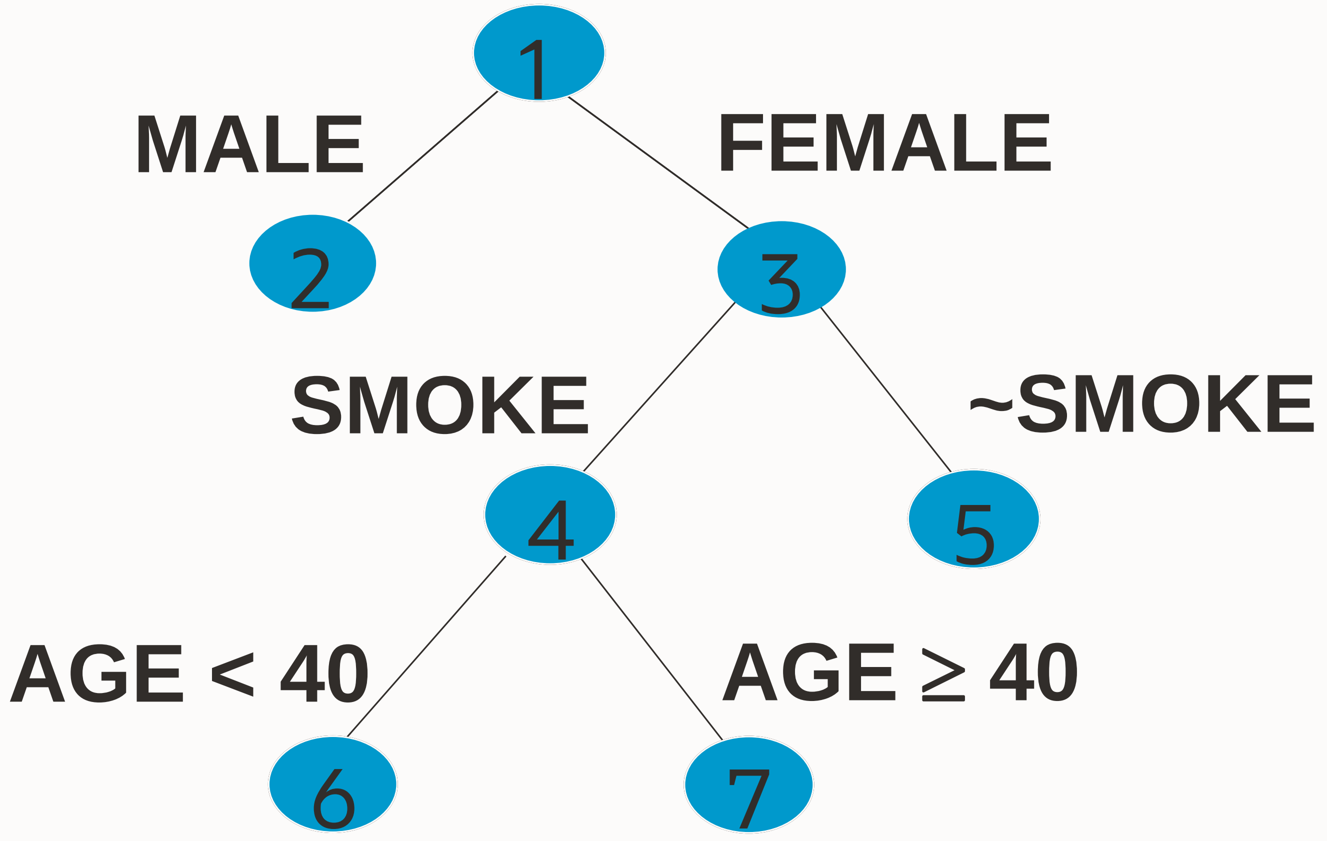
$0 < age \leq 35$ AND $0 < income \leq 50K$

Cluster 2:

$30 < age \leq 55$ AND $40K < income \leq 80K$

Copyright © 2020 Oracle and/or its affiliates.

Clustering Hierarchy



Binary tree

- balanced
- unbalanced

Splitting predicates

Copyright © 2020 Oracle and/or its affiliates.

Doc link

K-Means clustering

Identify distinct segments of a population

Explain the common characteristics of members of a cluster

Determine what distinguishes members of one cluster from members of another cluster

Partitions a set of observations into *k* partitions, or clusters

Each observations belongs to the cluster with the nearest *centroid* or *center*, which is the *mean* of the observations variables

Distance can be computed in various ways,
e.g., Euclidean or cosine

Copyright © 2020 Oracle and/or its affiliates.

ore.odmKMeans

K-Means Clustering

```
ore.odmKMeans(
  formula,
  data,
  auto.data.prep = TRUE,                # Setting to perform automatic data preparation
  num.centers = 10,                     # number of clusters
  block.growth = 2,                     # Numeric growth factor for memory to hold cluster data
  conv.tolerance = 0.01,                # Numeric convergence tolerance setting
  distance.function = "euclidean",      # Distance function: cosine, euclidean, or fast.cosine
  iterations = 3,                       # Maximum number of iterations
  min.pct.attr.support = 0.1,           # Minimum percent required for variables to appear in rules
  num.bins = 10,                        # Number of histogram bins
  split.criterion = "variance",         # Split clusters by variance or size
  na.action = na.pass,                  # Allow missing values in rows by default, or na.omit
  odm.setting = NULL)                   # A list to specify ODM parameter settings
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

num.centers – number of clusters to create, > 1, default 10

block.growth – numeric growth factor for memory to hold cluster data, [1..5], default 2

conv.tolerance – numeric convergence tolerance setting, (0..0.5], default 0.01

distance.function

- distance function between instances and centroids
- options: cosine, euclidean, or fast.cosine
- default: euclidean

iterations – maximum number of iterations, [1..20], default 3

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

min.pct.attr.support

- minimum percent required for variables to appear in rules, [0,1], default 0.1
- The fraction of attribute values that must be non-null for variable to be included in rule description for cluster
- Setting the parameter value too high in data with missing values can result in very short or even empty rules

num.bins

- number of histogram bins, > 0, default 10
- specifies the number of bins in the variable histogram produced by k-Means
- bin boundaries for each variable are computed globally on entire training data set
- binning method is equi-width
- all attributes have same number of bins except variables with a single value, which have only one bin

split.criterion

- split clusters by variance or size, default variance
- use size for more balanced clusters, e.g., with text mining

Copyright © 2020 Oracle and/or its affiliates.

ore.odmKMeans

K-Means Clustering

```
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
colnames(x) <- c("x", "y")
X <- ore.push (data.frame(x))
km.mod1 <- ore.odmKMeans(~., X, num.centers=2, num.bins=5)
summary(km.mod1)
rules(km.mod1)
clusterhists(km.mod1)
histogram(km.mod1)
```

```
R> summary(km.mod1)

Call:
ore.odmKMeans(formula = ~., data = X, num.centers = 2, num.bins = 5)

Settings:
                           value
clus.num.clusters              2
block.growth                   2
conv.tolerance              0.01
distance               euclidean
iterations                     3
min.pct.attr.support         0.1
num.bins                       5
split.criterion         variance
prep.auto                     on

Centers:
            x            y
2  1.05630476 1.0455933541
3 -0.01131291 0.0001622473
```

Copyright © 2020 Oracle and/or its affiliates.

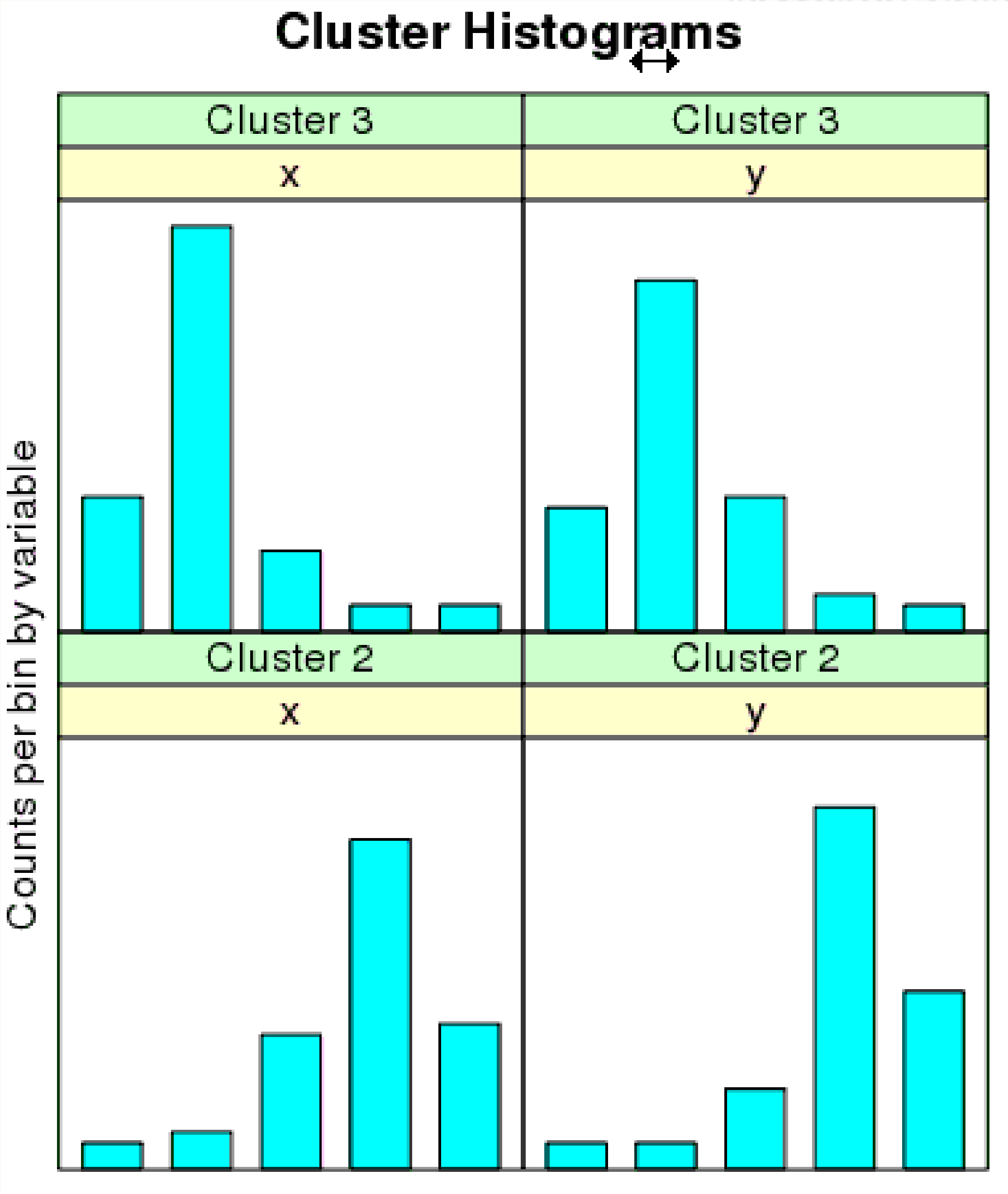
ore.odmKMeans – results

```
R> rules(km.mod1)
   rhs.cluster.id rhs.support rhs.conf lhs.support lhs.conf lhs.var lhs.var.num.val lhs.var.chr.val lhs.var.support lhs.var.conf
1               1         100      1.0          90      0.9       x      -0.2084763            <NA>              90         0.20
2               1         100      1.0          90      0.9       x       1.9020637            <NA>              90         0.20
3               1         100      1.0          90      0.9       y      -0.2588969            <NA>              91         0.20
4               1         100      1.0          90      0.9       y       1.6761630            <NA>              91         0.20
5               2          50      0.5          45      0.9       x       0.3191587            <NA>              49         0.25
6               2          50      0.5          45      0.9       x       1.9020637            <NA>              49         0.25
7               2          50      0.5          45      0.9       y       0.7086331            <NA>              45         0.50
8               2          50      0.5          45      0.9       y       1.6761630            <NA>              45         0.50
9               3          50      0.5          45      0.9       x      -0.7361113            <NA>              45         0.80
10              3          50      0.5          45      0.9       x       0.3191587            <NA>              45         0.80
11              3          50      0.5          45      0.9       y      -0.7426619            <NA>              49         0.60
12              3          50      0.5          45      0.9       y       0.7086331            <NA>              49         0.60
$`1`
  rhs.cluster.id rhs.support rhs.conf lhs.support lhs.conf lhs.var lhs.var.support lhs.var.conf   predicate
2              1         100        1          90      0.9       x              90          0.2   <= 1.9021
1              1         100        1          90      0.9       x              90          0.2  >= -0.2085
4              1         100        1          90      0.9       y              91          0.2   <= 1.6762
3              1         100        1          90      0.9       y              91          0.2  >= -0.2589

$`2`
  rhs.cluster.id rhs.support rhs.conf lhs.support lhs.conf lhs.var lhs.var.support lhs.var.conf  predicate
6              2          50      0.5          45      0.9       x              49         0.25  <= 1.9021
5              2          50      0.5          45      0.9       x              49         0.25  >= 0.3192
8              2          50      0.5          45      0.9       y              45         0.50  <= 1.6762
7              2          50      0.5          45      0.9       y              45         0.50  >= 0.7086

$`3`
   rhs.cluster.id rhs.support rhs.conf lhs.support lhs.conf lhs.var lhs.var.support lhs.var.conf   predicate
10              3          50      0.5          45      0.9       x              45          0.8   <= 0.3192
9               3          50      0.5          45      0.9       x              45          0.8  >= -0.7361
12              3          50      0.5          45      0.9       y              49          0.6   <= 0.7086
11              3          50      0.5          45      0.9       y              49          0.6  >= -0.7427
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmKMeans – results

```
R> clusterhists(km.mod1)
   cluster.id variable bin.id lower.bound upper.bound                         label count
1           1        x      1  -0.7361113  -0.2084763 -7.361E-01 : -2.085E-01    10
2           1        x      2  -0.2084763   0.3191587 -2.085E-01 :  3.192E-01    36
3           1        x      3   0.3191587   0.8467937  3.192E-01 :  8.468E-01    15
4           1        x      4   0.8467937   1.3744287  8.468E-01 :  1.374E+00    28
5           1        x      5   1.3744287   1.9020637  1.374E+00 :  1.902E+00    11
6           1        y      1  -0.7426619  -0.2588969 -7.427E-01 : -2.589E-01     9
7           1        y      2  -0.2588969   0.2248681 -2.589E-01 :  2.249E-01    30
8           1        y      3   0.2248681   0.7086331  2.249E-01 :  7.086E-01    15
9           1        y      4   0.7086331   1.1923980  7.086E-01 :  1.192E+00    32
10          1        y      5   1.1923980   1.6761630  1.192E+00 :  1.676E+00    14
11          2        x      1  -0.7361113  -0.2084763 -7.361E-01 : -2.085E-01     0
12          2        x      2  -0.2084763   0.3191587 -2.085E-01 :  3.192E-01     1
13          2        x      3   0.3191587   0.8467937  3.192E-01 :  8.468E-01    10
14          2        x      4   0.8467937   1.3744287  8.468E-01 :  1.374E+00    28
15          2        x      5   1.3744287   1.9020637  1.374E+00 :  1.902E+00    11
16          2        y      1  -0.7426619  -0.2588969 -7.427E-01 : -2.589E-01     0
17          2        y      2  -0.2588969   0.2248681 -2.589E-01 :  2.249E-01     0
18          2        y      3   0.2248681   0.7086331  2.249E-01 :  7.086E-01     5
19          2        y      4   0.7086331   1.1923980  7.086E-01 :  1.192E+00    31
20          2        y      5   1.1923980   1.6761630  1.192E+00 :  1.676E+00    14
21          3        x      1  -0.7361113  -0.2084763 -7.361E-01 : -2.085E-01    10
22          3        x      2  -0.2084763   0.3191587 -2.085E-01 :  3.192E-01    35
23          3        x      3   0.3191587   0.8467937  3.192E-01 :  8.468E-01     5
24          3        x      4   0.8467937   1.3744287  8.468E-01 :  1.374E+00     0
25          3        x      5   1.3744287   1.9020637  1.374E+00 :  1.902E+00     0
26          3        y      1  -0.7426619  -0.2588969 -7.427E-01 : -2.589E-01     9
27          3        y      2  -0.2588969   0.2248681 -2.589E-01 :  2.249E-01    30
28          3        y      3   0.2248681   0.7086331  2.249E-01 :  7.086E-01    10
29          3        y      4   0.7086331   1.1923980  7.086E-01 :  1.192E+00     1
30          3        y      5   1.1923980   1.6761630  1.192E+00 :  1.676E+00     0
```



Copyright © 2020 Oracle and/or its affiliates.

ore.odmKMeans

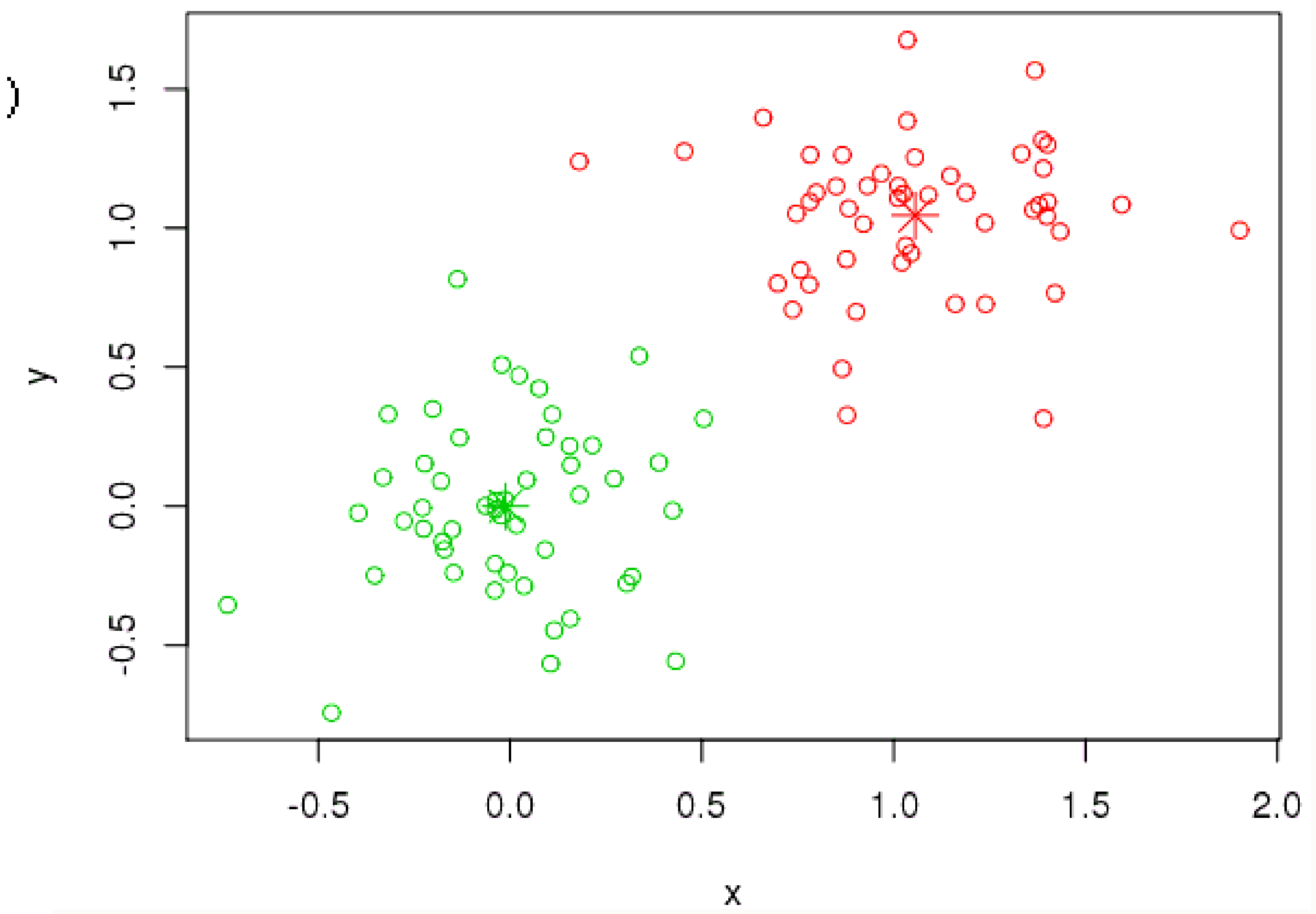
K-Means Clustering

```
km.res1 <- predict(km.mod1,X,type="class",supplemental.cols=c("x","y"))
head(km.res1,3)
km.res1.local <- ore.pull(km.res1)

plot(data.frame(x=km.res1.local$x, y=km.res1.local$y), col=km.res1.local$CLUSTER_ID)
points(km.mod1$centers2, col = rownames(km.mod1$centers2), pch = 8, cex=2)

head(predict(km.mod1,X))
head(predict(km.mod1,X,type=c("class","raw"),supplemental.cols=c("x","y")),3)
head(predict(km.mod1,X,type="raw",supplemental.cols=c("x","y")),3)
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmKMeans – results

```
R> km.res1 <- predict(km.mod1,X,type="class",supplemental.cols=c("x","y"))
R> head(km.res1,3)
            x          y CLUSTER_ID
1 -0.03999935 -0.3029228          3
2  0.50486611  0.3145332          3
3 -0.20133745  0.3497027          3
R> km.res1.local <- ore.pull(km.res1)
R> plot(data.frame(x=km.res1.local$x, y=km.res1.local$y), col=km.res1.local$CLUSTER_ID)
R> points(km.mod1$centers2, col = rownames(km.mod1$centers2), pch = 8, cex=2)
R>
R> head(predict(km.mod1,X))
        '3'          '2' CLUSTER_ID
1 0.9999998 1.844763e-07          3
2 0.9338791 6.612089e-02          3
3 0.9999185 8.154833e-05          3
4 0.9999520 4.798267e-05          3
5 0.9999885 1.153331e-05          3
6 0.9995041 4.959004e-04          3
R> head(predict(km.mod1,X,type=c("class","raw"),supplemental.cols=c("x","y")),3)
        '3'          '2'           x          y CLUSTER_ID
1 0.9999998 1.844763e-07 -0.03999935 -0.3029228          3
2 0.9338791 6.612089e-02  0.50486611  0.3145332          3
3 0.9999185 8.154833e-05 -0.20133745  0.3497027          3
R> head(predict(km.mod1,X,type="raw",supplemental.cols=c("x","y")),3)
            x          y       '3'          '2'
1 -0.03999935 -0.3029228 0.9999998 1.844763e-07
2  0.50486611  0.3145332 0.9338791 6.612089e-02
3 -0.20133745  0.3497027 0.9999185 8.154833e-05
```



Copyright © 2020 Oracle and/or its affiliates.

K-Means – model object

ore.odmKMeans object

- **name** …
- **settings** …
- **attributes** …
- **cluster** contain general per-cluster information
- **leaf.cluster.count** leaf clusters with support
- **taxonomy** is the parent-child cluster relationship
- **centers** are per cluster-attribute center (centroid) information
- **formula** …
- **call** …

Copyright © 2020 Oracle and/or its affiliates.

ore.odmKMeans with text mining

K-Means Clustering

```
dat <- scan("SOTU-2009.txt", what=character(), sep="\n")
df <- data.frame(ID = seq(length(dat)), PARAGRAPH = dat)
SOTU_TEXT <- ore.push(df)
ore.exec("begin ctx_ddl.create_policy('MY_TXTPOL'); end;") # CTXSYS.CTX_DDL privilege required

km.mod <- ore.odmKMeans( ~ PARAGRAPH, data = SOTU_TEXT, num.centers = 10L,
                         odm.settings = list(ODMS_TEXT_POLICY_NAME = "MY_TXTPOL",
                                             ODMS_TEXT_MIN_DOCUMENTS = 2,
                                             ODMS_TEXT_MAX_FEATURES = 20,
                                             kmns_distance ="dbms_data_mining.kmns_cosine",
                                             kmns_details = "kmns_details_all"),
                         ctx.settings = list(PARAGRAPH="TEXT(TOKEN_TYPE:STEM)"))
```

Copyright © 2020 Oracle and/or its affiliates.

Orthogonal Partitioning Clustering

O-Cluster

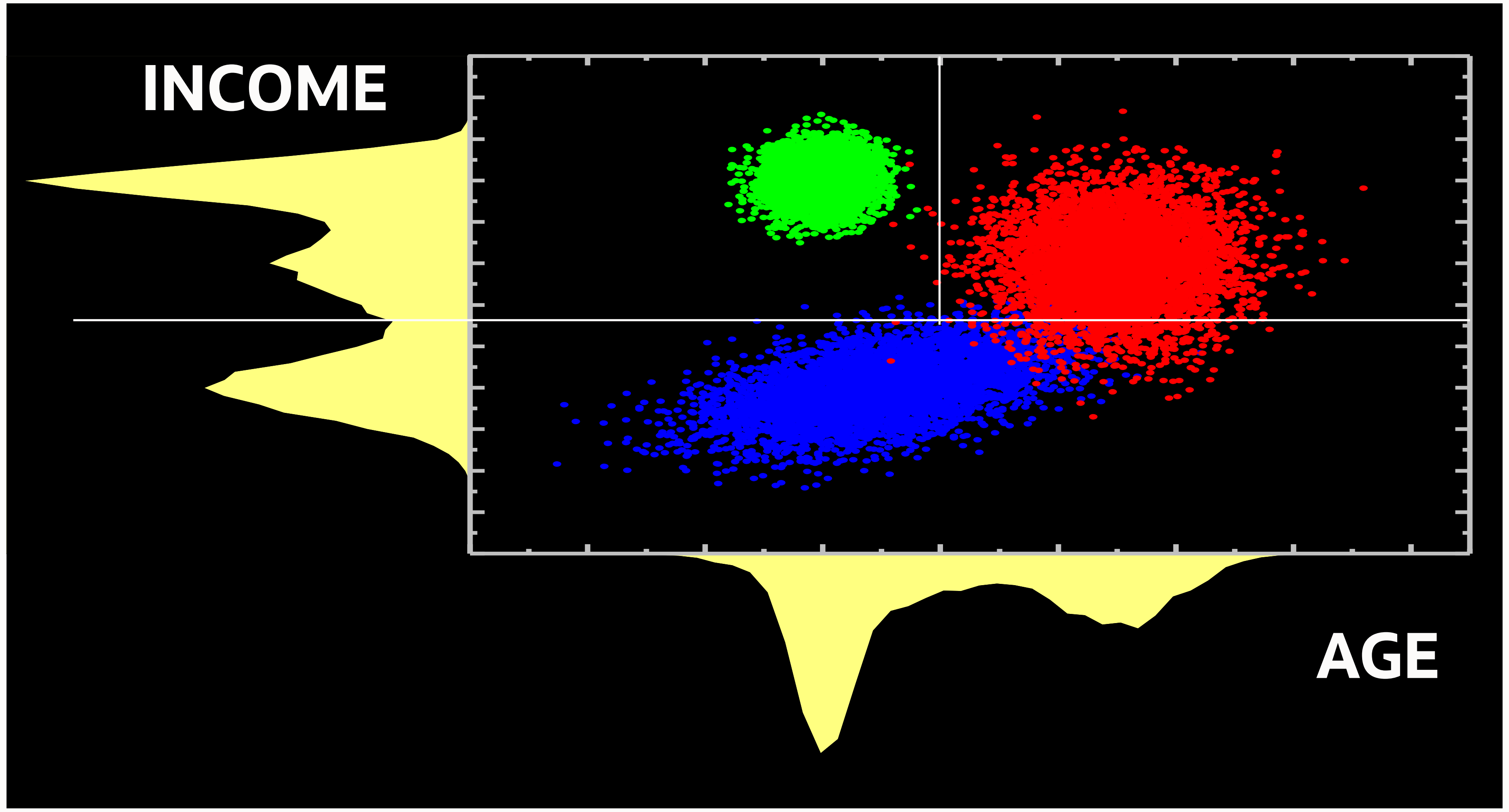
Doc link

Uses grid-based approach

Finds natural data clusters

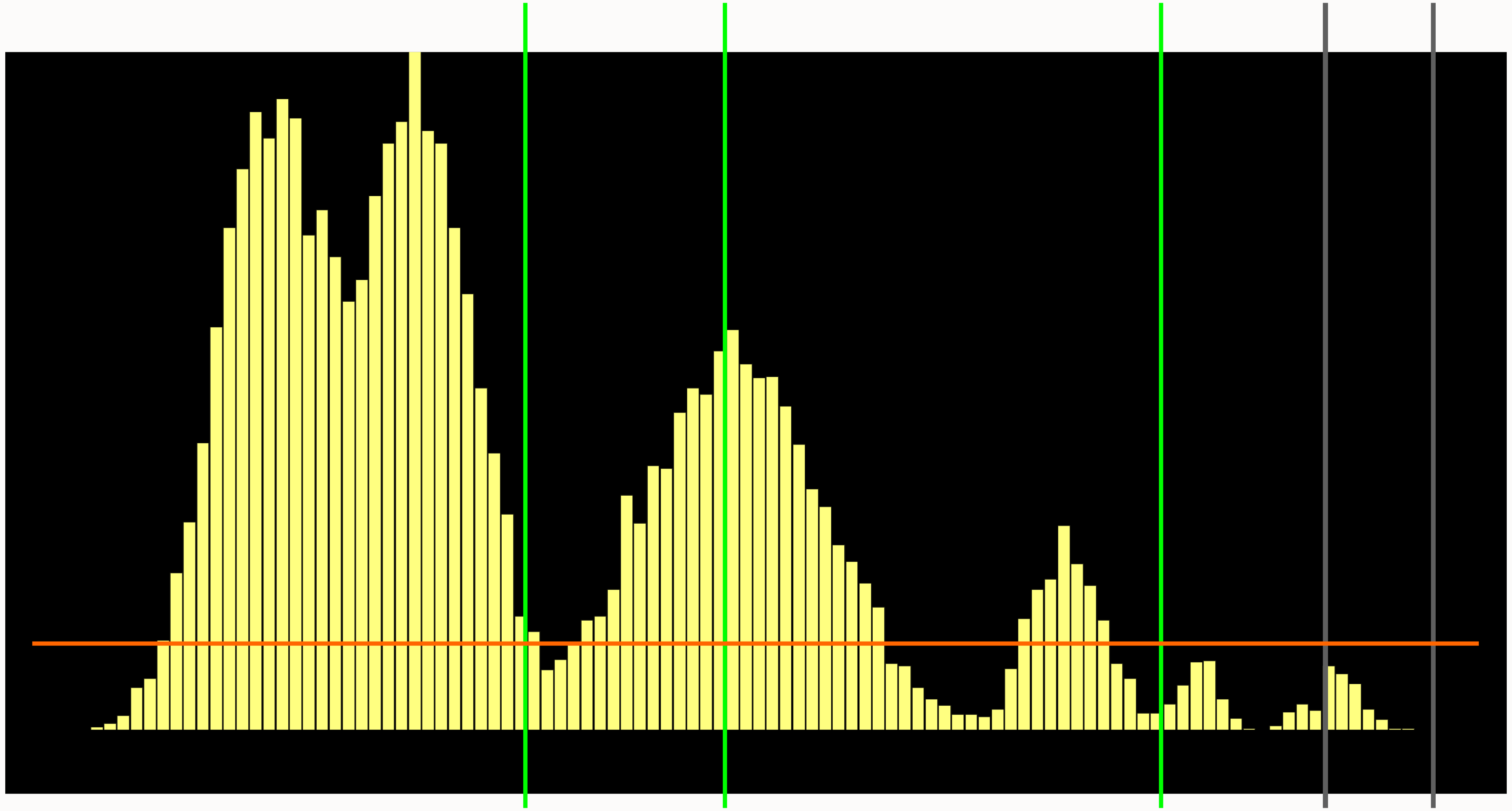
Creates unbalanced hierarchical trees

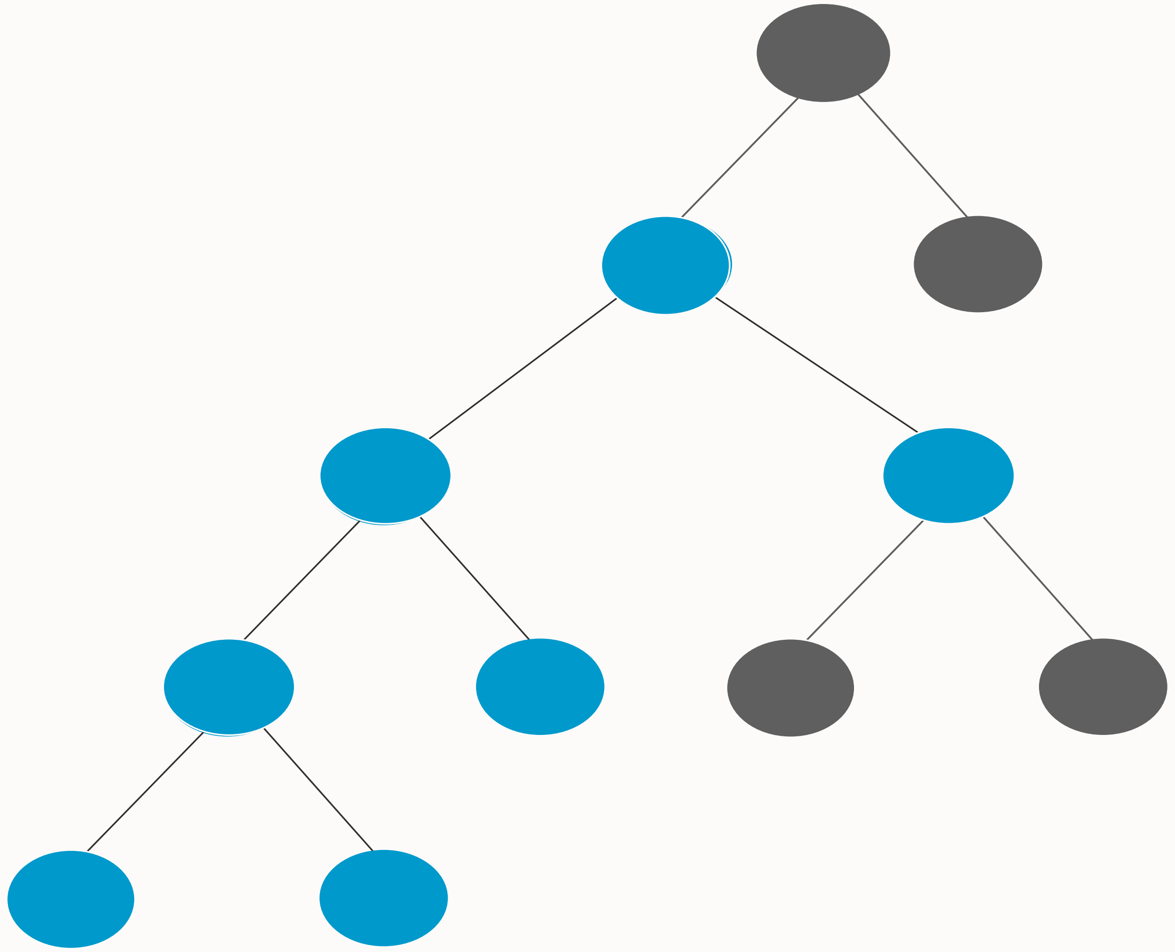
Uses active sampling



Copyright © 2020 Oracle and/or its affiliates.

O-Cluster Grid-Based Partitioning





Copyright © 2020 Oracle and/or its affiliates.

When to Use O-Cluster?

High number of records

- needed for detailed histogram computation

High number of attributes

Presence of noise

Numeric and categorical attributes

Multi-modal density data

- finds “natural” clusters, may not reach max number of clusters
  set by the user

Copyright © 2020 Oracle and/or its affiliates.

Orthogonal Partitioning Clustering

Creates a hierarchical grid-based clustering model

- creates axis-parallel (orthogonal) partitions in the input attribute space
- operates recursively
- resulting hierarchical structure represents irregular grid that tessellates attribute space into clusters
- resulting clusters define dense areas in the attribute space

Clusters described by intervals along the attribute axes and corresponding centroids and histograms

Parameter 'sensitivity' defines a baseline density level

- Only areas with peak density above this baseline level can be identified as clusters

O-Cluster separates areas of high density by placing cutting planes through areas of low density

- O-Cluster needs multi-modal histograms (peaks and valleys)
- If an area has projections with uniform or monotonically changing density, O-Cluster does not partition it

Copyright © 2020 Oracle and/or its affiliates.

Orthogonal Partitioning Clustering

O-Cluster reads the data in batches (the default batch size is 50000)

- Only read another batch if, based on statistical tests, there may still exist clusters that it has not yet uncovered.
- Since O-Cluster may stop the model build before it reads all of the data, it is highly recommended that the data be randomized
- Binary attributes should be declared as categorical
- O-Cluster maps categorical data to numerical values
- Recommend to use ODM’s equi-width binning transformation with automated estimation of the required number of bins
- Outliers can significantly impact clustering algorithms
  - Use a clipping transformation before binning or normalizing
  - Outliers with equi-width binning can prevent O-Cluster from detecting clusters - as a result, the whole population appears to falls within a single cluster.

Copyright © 2020 Oracle and/or its affiliates.

ore.odmOC

Orthogonal Partitioning Clustering

```
ore.odmOC(formula,
           data,
           auto.data.prep = TRUE,
           num.centers = 10,
           max.buffer = 50000,
           sensitivity = 0.5,
           na.action = na.pass,
           odm.setting = NULL

    ## S3 method for class 'ore.odmOC'
    predict(object,
            newdata,
            supplemental.cols = NULL,
            type = c("class","raw"),
            na.action = na.pass,...)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

num.centers – number of clusters to create, > 1, default 10

max.buffer – maximum buffer size, >0, default 50000

sensitivity – A fraction that specifies the peak density required for separating a new cluster. The fraction is related to the global uniform density. Value [0,1]. (default: 0.5)

Copyright © 2020 Oracle and/or its affiliates.

OCluster – model object

ore.odmOC object

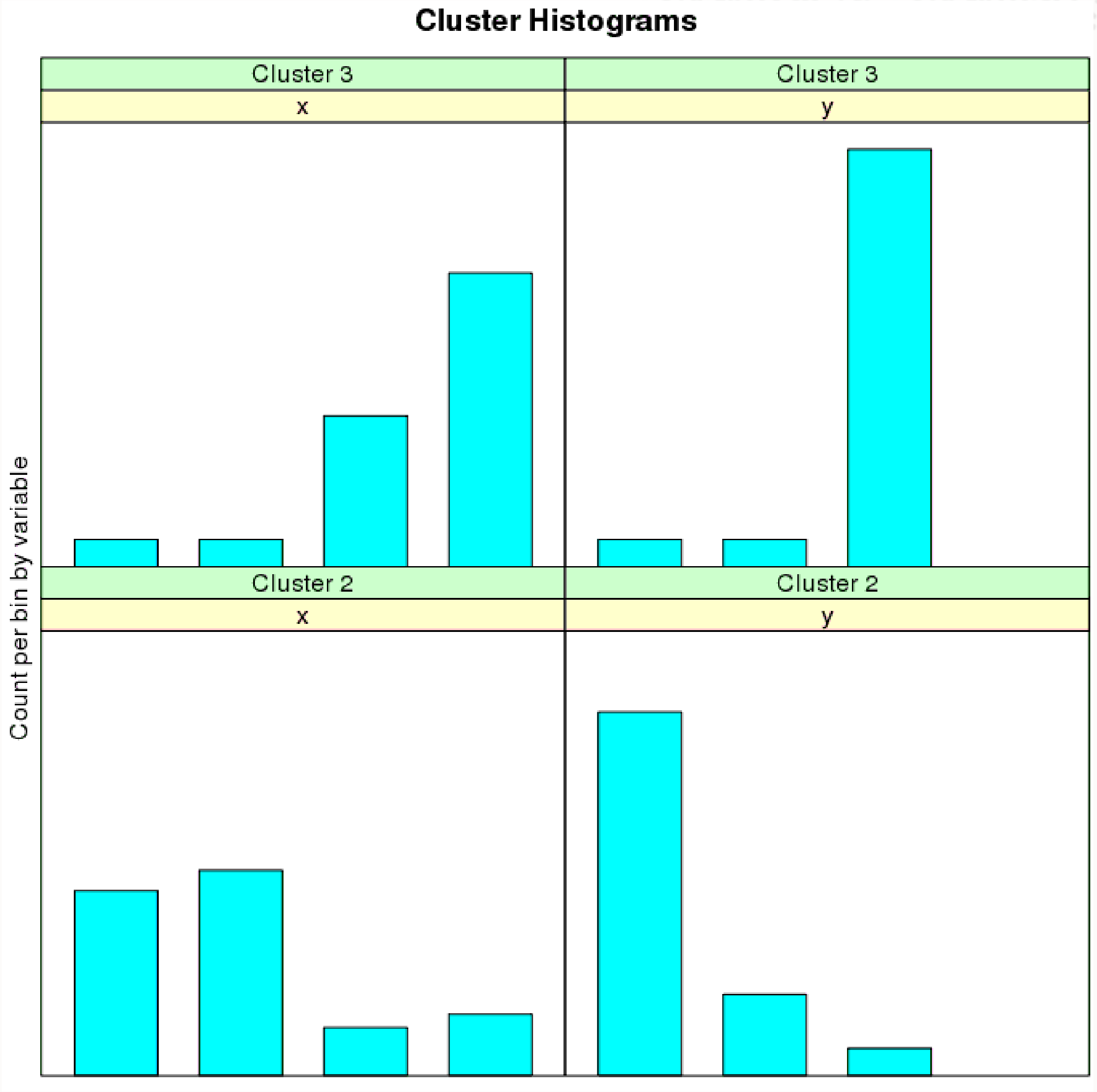
- **name** name of model in database
- **settings** data.frame with settings used to build model
- **attributes** data.frame of variable/columns used to build model
- **clusters** contain general per-cluster information
- **leaf.cluster.count** data.frame of leaf clusters with support
- **taxonomy** parent-child cluster relationship
- **centers** per cluster-attribute center (centroid) information
- **centers2** simplified cluster centroids (means)
- **histogram** per cluster attribute histogram information
- **rules** rules defining clusters
- **formula** formula used to build the model
- **call** specific invocation of the function with arguments

Copyright © 2020 Oracle and/or its affiliates.

ore.odmOC

O-Cluster Clustering

```
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 2, sd = 0.3), ncol = 2))
colnames(x) <- c("x", "y")
X <- ore.push (data.frame(x))

oc.mod1 <- ore.odmOC(~., X, num.centers=2)
summary(oc.mod1)
rules(oc.mod1)
clusterhists(oc.mod1)
histogram(oc.mod1)
```

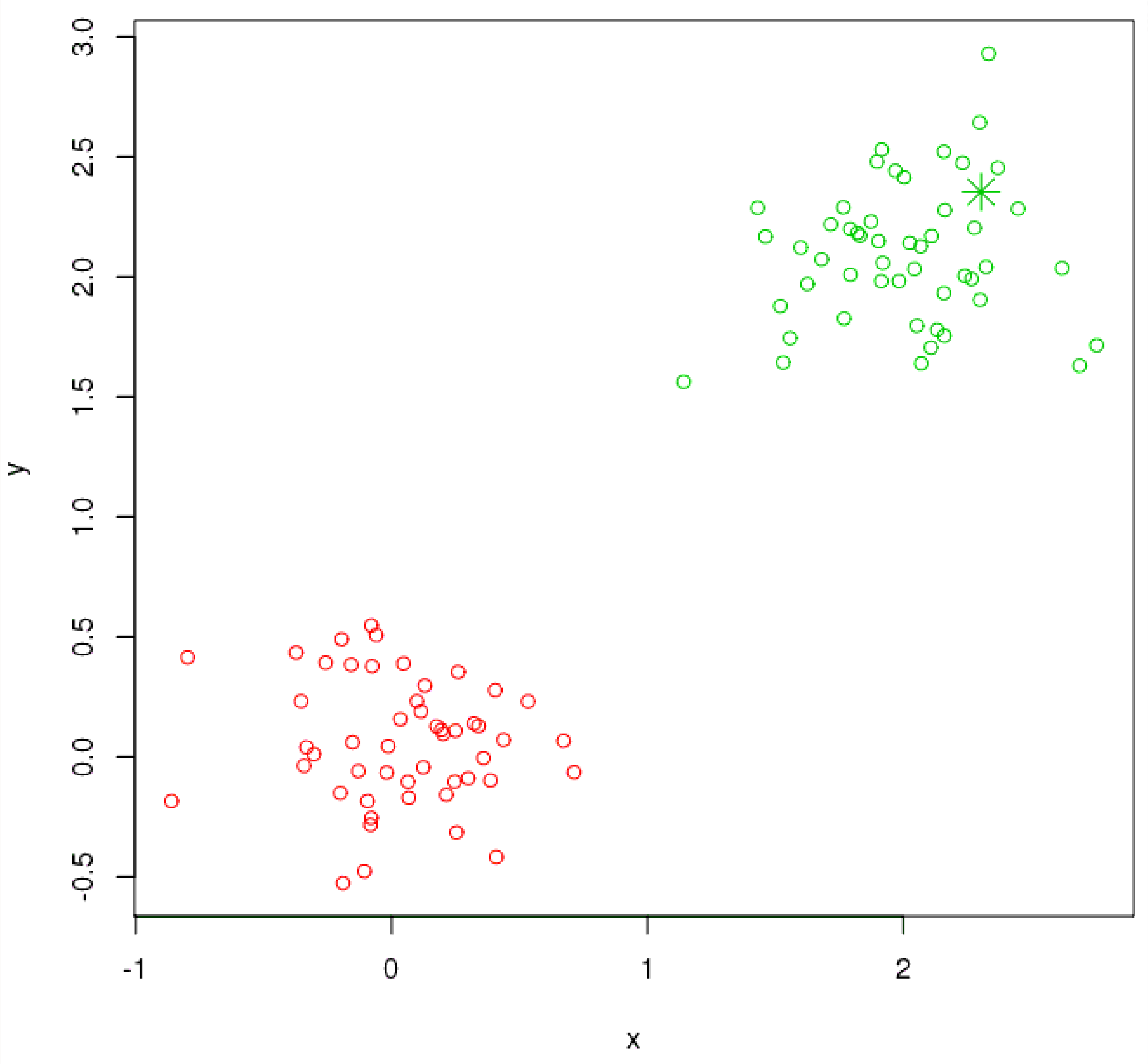


Copyright © 2020 Oracle and/or its affiliates.

ore.odmOC

O-Cluster Clustering

```
oc.res1 <- predict(oc.mod1,X,type="class",
                   supplemental.cols=c("x","y"))
head(oc.res1,3)
oc.res1.local <- ore.pull(oc.res1)

plot(data.frame(x=oc.res1.local$x,
                y=oc.res1.local$y),
                col=oc.res1.local$CLUSTER_ID)
points(oc.mod1$centers2,
       col = rownames(oc.mod1$centers2),
       pch = 8, cex=2)
```



Copyright © 2020 Oracle and/or its affiliates.

Expectation Maximization Clustering

Popular probability density estimation technique

EM used to implement a distribution-based clustering algorithm (EM-clustering)

Copyright © 2020 Oracle and/or its affiliates.

Expectation Maximization Clustering

ore.odmEM

[Doc link]

- Automated model search to find number of clusters / components (enabled via EMCS_MODEL_SEARCH)
- Protection against overfitting
- Supports numeric and multinomial distributions
- High quality probability estimates
- Generates cluster hierarchy, rules, and other statistics
- Supports both Gaussian and multi-value Bernoulli distributions
- Includes heuristics that automatically choose distribution types

Copyright © 2020 Oracle and/or its affiliates.

When to Use EM?

In general, EM is a significantly more expensive algorithm than k-Means. If you have a large dataset, k-Means should be the first choice. However, Oracle's EM is very scalable relative to other EM implementations.

Parallel implementation allows this EM algorithm to scale linearly with the number of rows. High column dimensionality is handled through the feature selection or random projections.

Provides a component clustering capability to group overlapping EM components into higher level clusters, enabling discovery of arbitrarily shaped clusters. This feature is on by default and may result in fewer clusters than the maximum size. It is also important to distinguish between EM components and the concept of clusters which can include multiple components.

Performs automatic feature selection by removing statistically independent columns, which effectively removes irrelevant noisy columns

Nested columns (ODM SQL only) and text use random projections and are modeled in a lower dimensional space

Copyright © 2020 Oracle and/or its affiliates.

ore.odmEM

Expectation Maximization Clustering

```
ore.odmEM(formula,
          data,
          num.centers = NULL,
          auto.data.prep = TRUE,
          na.action = na.pass,
          odm.setting = NULL)

histogram(x,
        data=NULL,
        cluster.id="all",...)

predict (object,
        newdata,
        supplemental.cols = NULL,
        type = c("class","raw"),
        na.action = na.pass,...)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

num.centers – number of clusters to create, > 1, default NULL – system determined

auto.data.prep – default TRUE

odm.setting – A list to specify Oracle Data Mining parameter settings. This argument is applicable to building a model in Database 12.2 or later. Each list element's name and value refer to the parameter setting name and value, respectively. The setting values must be numeric or string. When parameter `ODMS_PARTITION_COLUMNS` is set to the name(s) of the partition column(s), a partition model with a sub-model in each partition is created from the input data. (See ODM documentation for specific settings options.)

Copyright © 2020 Oracle and/or its affiliates.

EM – model object

ore.odmEM object

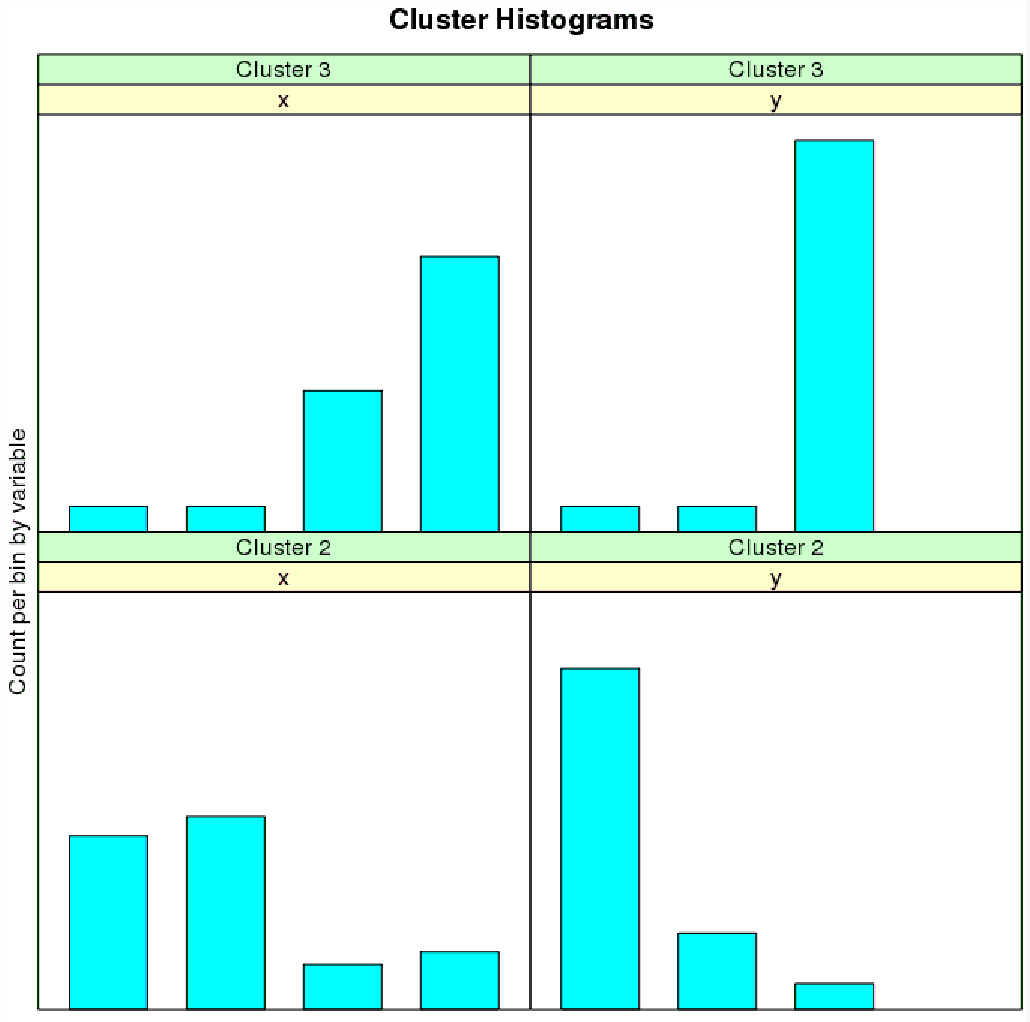
- **name** name of model in database
- **settings** data.frame with settings used to build model
- **attributes** data.frame of variable/columns used to build model
- **clusters** contain general per-cluster information
- **leaf.cluster.count** data.frame of leaf clusters with support
- **taxonomy** parent-child cluster relationship
- **centers** per cluster-attribute center (centroid) information
- **centers2** simplified cluster centroids (means)
- **formula** formula used to build the model
- **call** specific invocation of the function with arguments

Copyright © 2020 Oracle and/or its affiliates.

ore.odmEM

Expectation Maximization Clustering

```
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
colnames(x) <- c("x", "y")

X <- ore.push(cbind(data.frame(x),
              part = as.integer(x[,2] * 100)\%\%2))
em.mod <- ore.odmEM(~. -part, X, num.centers = 3)
em.mod
summary(em.mod)
rules(em.mod)
clusterhists(em.mod)
histogram(em.mod)
```

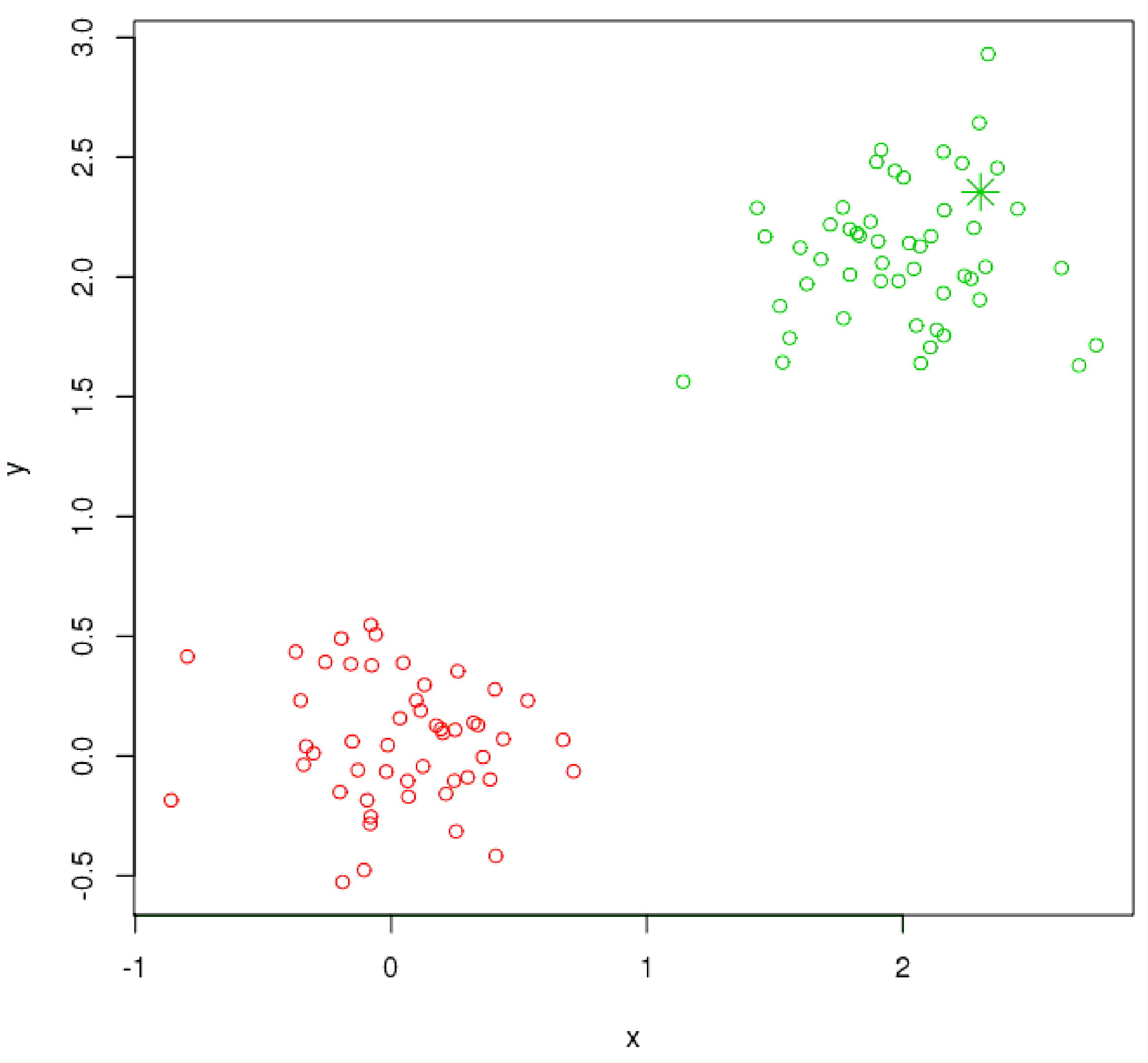


Copyright © 2020 Oracle and/or its affiliates.

ore.odmEM

Expectation Maximization Clustering

```
em.res1 <- predict(em.mod,X,type="class",
                   supplemental.cols=c("x","y"))
head(em.res1,3)
em.res1.local <- ore.pull(em.res1)
plot(data.frame(x=em.res1.local$x,
                y=em.res1.local$y),
                col=em.res1.local$CLUSTER_ID)
points(em.mod$centers2, col =rownames(em.mod$centers2),
       pch = 8, cex=2)

head(predict(em.mod,X))
head(predict(em.mod,X,type=c("class","raw"),
             supplemental.cols=c("x","y")),3)
```

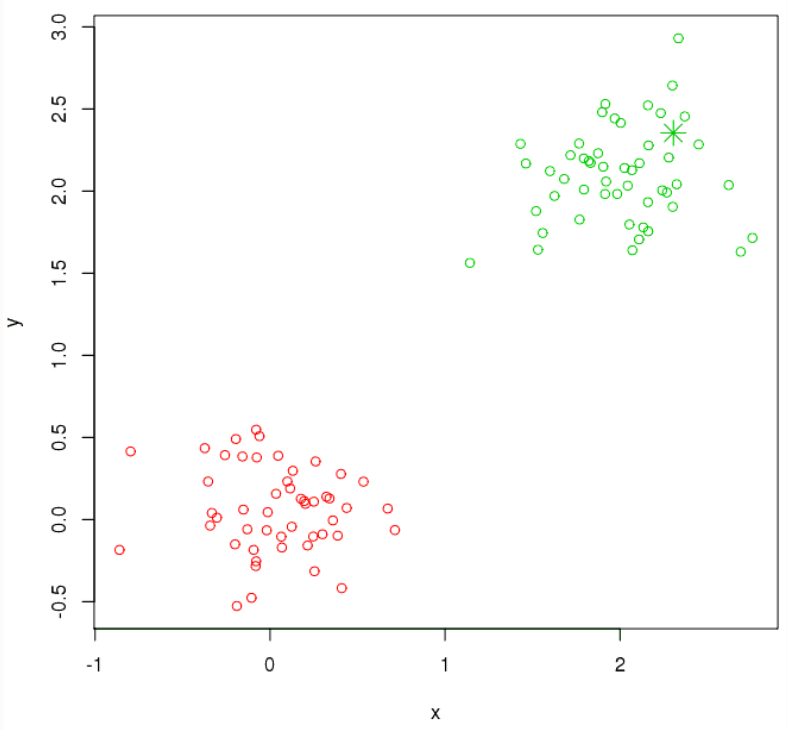


Copyright © 2020 Oracle and/or its affiliates.

ore.odmEM

Expectation Maximization Clustering with partitioned model

```
em.pmod <- ore.odmEM(~. , X, num.centers = 3,
           odm.setting = list(odms_partition_columns = "part"))
partitions(em.pmod)
summary(em.pmod)
rules(em.pmod)
clusterhists(em.pmod)
histogram(em.pmod, part = "DM$$_P1")

head(predict(em.pmod,X))
head(predict(em.pmod,X,type=c("class","raw"),
             supplemental.cols=c("x","y")),3)
head(predict(em.pmod,X,type="raw",
             supplemental.cols=c("x","y")),3)
```



Copyright © 2020 Oracle and/or its affiliates.

Doc link

Decision Tree

Classification algorithm

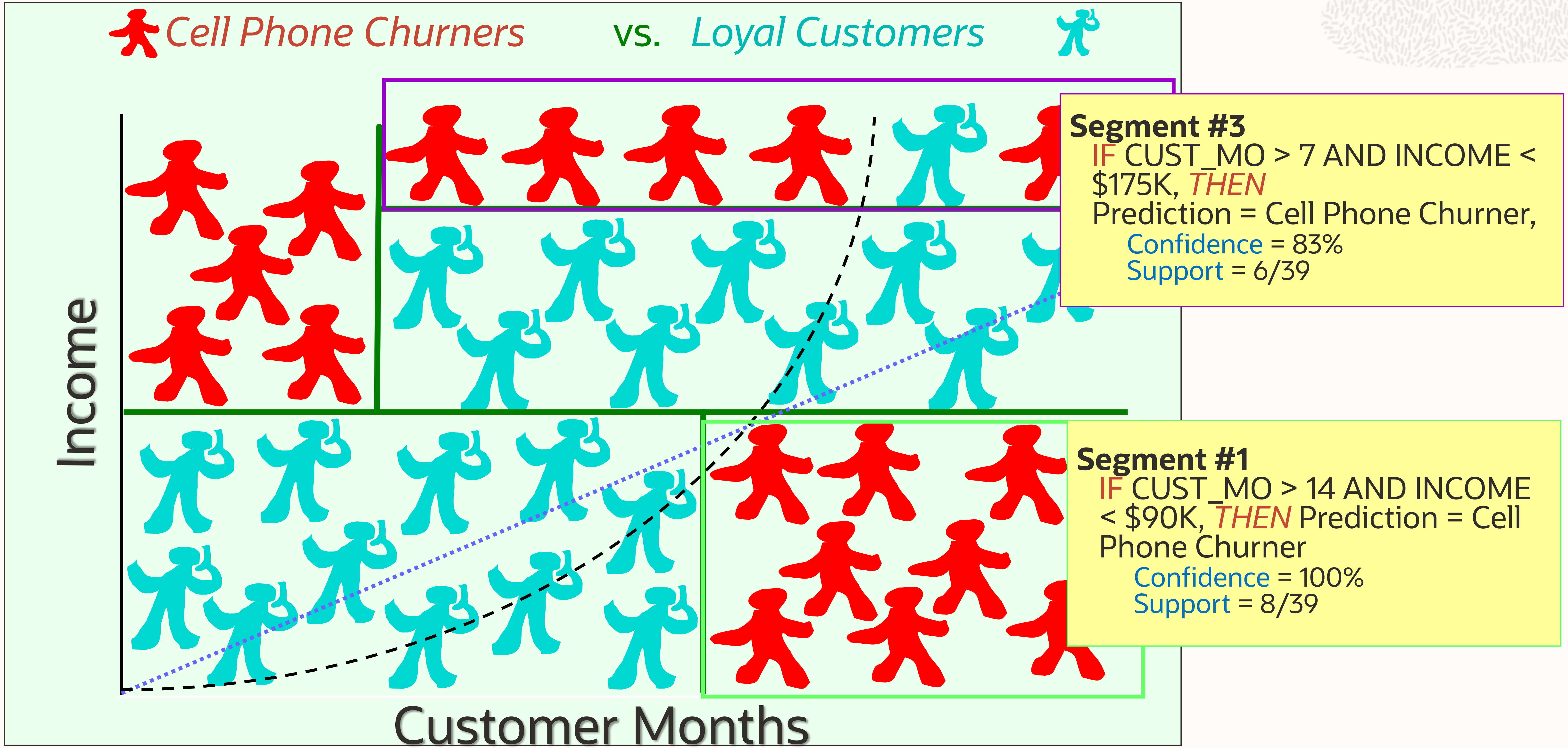
- Predicts a discrete value for each case: 0 or 1, Yes or No, Low Medium or High, with corresponding probability
- Based on classification component of well-known C&RT algorithm
- Enhancement of supplying Surrogate splitting attributes, if possible, at each node

Uses include

- Prediction
- Segmentation
- Understanding predictions

Copyright © 2020 Oracle and/or its affiliates.

Decision Tree Example



Source: Inspired from *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management* by Michael J. A. Berry, Gordon S. Linoff

Copyright © 2020 Oracle and/or its affiliates.

ore.odmDT

Decision Tree

```
ore.odmDT(
  formula,                      # formula specifying attributes for model build
  data,                         # ore.frame of the training dataset
  auto.data.prep = TRUE,        # Setting to perform automatic data preparation
  cost.matrix = NULL,           # numerical sq matrix for costs of incorrect prediction
  impurity.metric = "gini",     # gini or entropy
  max.depth = 7,                # maximum depth of tree from root to leaf inclusive [2..20]
  min.rec.split = 20,           # minimum number of cases required to split a node
  min.pct.split = 0.1,          # minimum percent of cases required to split a node
  min.rec.node = 10,            # minimum number of cases required in a child node
  min.pct.node = 0.05,          # minimum percent of cases required in child node
  na.action = na.pass,          # Allows missing values (na.pass), or removes rows with
                                #    missing values (na.omit)
  odm.setting = NULL)           # A list to specify Oracle Data Mining parameter settings
)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

cost.matrix – default NULL

impurity.metric

- options: gini or entropy, default "gini“
- measure of node purity
- tree algorithms seek the best test question for splitting data at each node. The best splitter and split value are those that result in the largest increase in target value homogeneity (purity) for the entities in the node

max.depth

- default 7
- Criteria for splits: maximum tree depth (the maximum number of nodes between the root and any leaf node, including the leaf node)

min.rec.split – default 20

min.pct.split – default 0.1

min.rec.node – default 10

min.pct.node – default 0.05

Copyright © 2020 Oracle and/or its affiliates.

ore.odmDT

Decision Tree Classification

```
m <- mtcars
m$gear <- as.factor(m$gear)
m$cyl  <- as.factor(m$cyl)
m$vs   <- as.factor(m$vs)
m$ID   <- 1:nrow(m)
MTCARS <- ore.push(m)
row.names(MTCARS) <- MTCARS

dt.mod  <- ore.odmDT(gear ~ ., MTCARS)
summary(dt.mod)

dt.res  <- predict (dt.mod, MTCARS,"gear")
# confusion matrix
with(dt.res, table(gear,PREDICTION))
```

```
> dt.mod  <- ore.odmDT(gear ~ ., MTCARS)
> summary(dt.mod)
Call:
ore.odmDT(formula = gear ~ ., data = MTCARS)
  n =  32
Nodes:
  parent node.id row.count prediction             split
1     NA       0        32          3              <NA>
2      0       1        16          4 (disp <= 196.2999)
3      0       2        16          3  (disp > 196.2999)
           surrogate                 full.splits
1                <NA>                       <NA>
2 (cyl in ("4" "6" )) (disp <= 196.299999999999995)
3     (cyl in ("8" ))  (disp > 196.299999999999995)

Settings:
                          value
prep.auto                    on
impurity.metric   impurity.gini
term.max.depth                7
term.minpct.node           0.05
term.minpct.split           0.1
term.minrec.node             10
term.minrec.split            20

> dt.res  <- predict (dt.mod, MTCARS,"gear")
> with(dt.res, table(gear,PREDICTION))
    PREDICTION
gear  3  4
   3 14  1
   4  0 12
   5  2  3
```

Copyright © 2020 Oracle and/or its affiliates.

Decision Tree – model object

ore.odmDT object

- **name** …
- **settings** …
- **attributes** …
- **costs** a data.frame containing the cost matrix supplied at model build
- **Distributions** target class distributions at each tree node
- **nodes** a data.frame with tree node details, including: parent node id, node id, number of rows assigned to that node, predicted value, split predicate, surrogate variables (if applicable), and full split predicates from current node to root node
- **formula** …
- **call** …

Copyright © 2020 Oracle and/or its affiliates.

Generalized Linear Models

Doc link

Linear Models

Assumes Y is normally distributed with constant variance

Linear models fit

$$\mu_Y = \beta_o + \Sigma^p_{j=1}\beta_j X_j$$

No assumptions about predictors X_j distributions, e.g., need not be normally distributed

Nonlinear functions on predictors allowed

Advantages

- Computational simplicity
- Interpretable model form
- ability to compute certain diagnostic information about the quality of the fit

Generalized Linear Models

Addresses target variables that are non-normal

- Assume Y follows distribution from exponential family
- Specify link function and probability distribution, or variance function

GLM fits models of the form

$$g(\mu_Y) = \beta_o + \Sigma^p_{j=1}\beta_j X_j$$

$g(\mu_Y)$ is a function of the conditional mean, a.k.a. link function

http://en.wikipedia.org/wiki/Exponential_family

Copyright © 2020 Oracle and/or its affiliates.

ore.odmGLM

Generalized Linear Model

```
ore.odmGLM(
  formula,                    # formula specifying attributes for model build
  data,                       # ore.frame of the training dataset
  weights = NULL,
  type = c("normal", "logistic"),
  na.treatment = c("delete.row", "mean.or.mode"),
  reference = NULL,
  ridge = FALSE,
  ridge.value = NULL,
  ridge.vif = FALSE,
  auto.data.prep = TRUE,      # Setting to perform automatic data preparation
  odm.setting = NULL)         # A list to specify Oracle Data Mining parameter settings
)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

weights

- An optional character string representing the column name in the data argument to use as analytical weights in the model fit, Default NULL

type

- the type of generalized linear model, default “normal”
  - "normal" (Gaussian) - identify link function and variance function = 1 (constant over range of response values)
  - "logistic" (binomial) - logit link function and binomial variance function

na.treatment

- The missing value treatment; either "delete.row" (delete entire row) or "mean.or.mode" (replace missing values with the mean in numeric predictors and the mode in categorical predictors) , Default "delete.row"

reference

- An optional response variable category to use as the reference value (non-case/failure code) in a binary logistic regression model
- By default, reference is taken to be the category with the highest prevalence, default NULL

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

ridge

- Compensates for multicollinearity
- TRUE to enable ridge estimation of the coefficients, FALSE otherwise, default FALSE
- Applies both to regression and classification
- When enabled, no prediction bounds can be produced

ridge.value

- The value for the ridge parameter used by the algorithm
- Used when ridge regression explicitly enabled
- If ridge regression is enabled internally by the algorithm, the ridge parameter is determined by the algorithm, default NULL

ridge.vif

- (Linear regression only) Optional logical indicator for whether to produce Variance Inflation Factor (VIF) statistics for the ridge estimates
- VIFs can only be produced if enough Oracle database system resources are available
- Default FALSE

Copyright © 2020 Oracle and/or its affiliates.

ore.odmGLM

Generalized Linear Model Regression

```
# Linear regression using the longley data set
LONGLEY <- ore.push(longley)
longfit1 <- ore.odmGLM(Employed ~ ., data = LONGLEY)
summary(longfit1)
```

```
> longfit1 <- ore.odmGLM(Employed ~ ., data = LONGLEY)
> summary(longfit1)

Call:
ore.odmGLM(formula = Employed ~ ., data = LONGLEY)

Residuals:
     Min       1Q   Median       3Q      Max
-0.41011 -0.15767 -0.02816  0.10155  0.45539

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.482e+03  8.904e+02  -3.911 0.003560 **
GNP.deflator  1.506e-02  8.492e-02   0.177 0.863141
GNP          -3.582e-02  3.349e-02  -1.070 0.312681
Unemployed   -2.020e-02  4.884e-03  -4.136 0.002535 **
Armed.Forces -1.033e-02  2.143e-03  -4.822 0.000944 ***
Population   -5.110e-02  2.261e-01  -0.226 0.826212
Year          1.829e+00  4.555e-01   4.016 0.003037 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3049 on 9 degrees of freedom
Multiple R-squared: 0.9955,    Adjusted R-squared: 0.9925
F-statistic: 330.3 on 6 and 9 DF,  p-value: 4.984e-10
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmGLM

Generalized Linear Model Regression

```
# Ridge regression using the longley data set
longfit2 <- ore.odmGLM(Employed ~ ., data = LONGLEY,
                       ridge = TRUE,
                       ridge.vif = TRUE)
summary(longfit2)
```

```
> # Ridge regression using the longley data set
> longfit2 <- ore.odmGLM(Employed ~ ., data = LONGLEY,
+                        ridge = TRUE,
+                        ridge.vif = TRUE)
> summary(longfit2)

Call:
ore.odmGLM(formula = Employed ~ ., data = LONGLEY,
    ridge = TRUE, ridge.vif = TRUE)

Residuals:
    Min      1Q  Median      3Q     Max
-0.4100 -0.1579 -0.0271  0.1017  0.4575

Coefficients:
               Estimate   VIF
(Intercept)  -3.466e+03 0.000
GNP.deflator  1.479e-02 0.077
GNP          -3.535e-02 0.012
Unemployed   -2.013e-02 0.000
Armed.Forces -1.031e-02 0.000
Population   -5.262e-02 0.548
Year          1.821e+00 2.212

Residual standard error: 0.3049 on 9 degrees of freedom
Multiple R-squared: 0.9955,    Adjusted R-squared: 0.9925
F-statistic: 330.2 on 6 and 9 DF,  p-value: 4.986e-10
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmGLM

Generalized Linear Model Regression

```
# Logistic regresion using the infert data set
INFERT <- ore.push(infert)
infit1 <- ore.odmGLM(
 case ~ age+parity+education+spontaneous+induced,
                 data = INFERT,
                 type = "logistic")
infit1
```

```
R> # Logistic regresion using the infert data set
R> INFERT <- ore.push(infert)
R> infit1 <- ore.odmGLM(case ~ age+parity+education+spontaneous+induced,
+                        data = INFERT, type = "logistic")
R> infit1

Response:
case == "1"

Call:  ore.odmGLM(formula = case ~ age + parity + education + spontaneous +
    induced, data = INFERT, type = "logistic")

Coefficients:
     (Intercept)                 age              parity   education0-5yrs
        -2.19348             0.03958            -0.82828           1.04424
education12+ yrs         spontaneous             induced
        -0.35896             2.04590             1.28876

Degrees of Freedom: 247 Total (i.e. Null);  241 Residual
Null Deviance:       316.2
Residual Deviance: 257.8        AIC: 271.8
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmGLM – other functions

Generalized Linear Model

```
residuals(object,
       type = c("deviance", "pearson", "response"), ...)

fitted(object, ...)

predict(object, newdata, supplemental.cols = NULL,
        confint = FALSE, level = 0.95,
        na.action = na.pass,...)

confint(object, parm, level = 0.95, ...)

deviance(object, ...)

extractAIC(fit, scale = 0, k = 2, ...)

logLik(object, ...)

nobs(object, ...)
```

confint: A logical indicator for whether to produce confidence intervals for the predicted values.

level: A numeric value within [0, 1] to use for the confidence level.

na.action: Function to use for missing value handling; either 'na.pass' (allow missing values) or 'na.omit' (remove rows with missing values).

parm: An optional character vector that specifies which coefficients to include in the set of confidence intervals.

scale: An optional numeric scale parameter.

k: An optional numeric weight of the equivalent degrees of freedom.

Copyright © 2020 Oracle and/or its affiliates.

ore.odmGLM

Generalized Linear Model Prediction

```
res <- predict(infit1, newdata = INFERT, confint=TRUE, level = 0.97)
head(res)

head(residuals(infit1))
extractAIC(infit1)
logLik(infit1)
nobs(infit1)
```

```
R> res <- predict(infit1, newdata = INFERT, confint=TRUE, level = 0.97)
R> head(res)
  PREDICTION LOWER.CONF UPPER.CONF
1  0.5721917  0.1767983  0.8928118
2  0.7258536  0.2887066  0.9452694
3  0.1194461  0.5546963  0.9775927
4  0.3684102  0.2546444  0.8958629
5  0.5104286  0.3632442  0.6558268
6  0.6322268  0.4007028  0.8154924
R>
R> head(residuals(infit1))
[1] 1.0566751 0.8005085 2.0614994 1.4131937 1.1597452 0.9576085
R> extractAIC(infit1)
[1]   7.0000 271.7977
R> logLik(infit1)
'log Lik.' -128.8988 (df=7)
R> nobs(infit1)
[1] 248
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmGLM

Generalized Linear Model Regression

```
# Changing the reference value to 1
infit2 <- ore.odmGLM(
   case ~ age+parity+education+spontaneous+induced,
                  data = INFERT,
                  type = "logistic",
                  reference = 1)
infit2
```

```
R> # Changing the reference value to 1
R> infit2 <- ore.odmGLM(case ~ age+parity+education+spontaneous+induced,
+                       data = INFERT, type = "logistic", reference = 1)
R> infit2

Response:
case == "0"

Call:  ore.odmGLM(formula = case ~ age + parity + education + spontaneous +
    induced, data = INFERT, type = "logistic", reference = 1)

Coefficients:
     (Intercept)                age             parity   education0-5yrs
         2.19348           -0.03958            0.82828          -1.04424
education12+ yrs        spontaneous            induced
         0.35896           -2.04590           -1.28876

Degrees of Freedom: 247 Total (i.e. Null);  241 Residual
Null Deviance:      316.2
Residual Deviance: 257.8        AIC: 271.8
```

Copyright © 2020 Oracle and/or its affiliates.

Generalized Linear Model – model object

ore.odmGLM object

• **name**	…
• **settings**	…
• **attributes**	…
• **coefficients**	a named vector of coefficients
• **residuals**	ore.frame containing 3 types of residuals: "deviance", "pearson", and "response"
• **fitted.values**	an ore.vector containing the fitted values
• **rank**	numeric rank of the fitted model
• **type**	type of model fit
• **deviance**	minus twice the maximized log-likelihood, up to a constant
• **aic**	same version of Akaike's An Information Criterion as used by glm
• **null.deviance**	deviance for the null (intercept only) model
• **prior.weights**	weights initially supplied or 1 if none were
• **df.residual**	residual degrees of freedom

Copyright © 2020 Oracle and/or its affiliates.

Generalized Linear Model – model object (2)

ore.odmGLM object

• **df.null**	residual degrees of freedom for the null model
• **y**	ore.vector containing the response variable
• **converged**	indicator for whether the model converged
• **model**	ore.frame containing the model frame
• **na.treatment**	how missing values were treated
• **na.action**	number of rows with missing values that were removed
• **terms**	terms object used
• **data**	data argument
• **nonreference**	in logistic regression, the response values that represents success
• **ridge**	ridge argument
• **auto.data.prep**	whether or not auto data preparation should be used
• **fit.name**	internal name for the in-database model
• **fit.details**	model details
• **formula**	…
• **call**	…

Copyright © 2020 Oracle and/or its affiliates.

To Ridge or not to Ridge

If the data has a large number of attributes AND accuracy is more important than a compact model ridge is the preferred approach

If having a compact model is important, then feature selection is the preferred approach

If the problem is believed to be non-linear in nature or the user does not know, then it is also a good idea to create a model with feature generation on

- Will generate compact polynomial (quadratic or cubic) models that may fit the data better

User can also easily create these 3 types of models in a single model build node in Oracle Data Miner and then compare models to select best model

Copyright © 2020 Oracle and/or its affiliates.

GLM Analysis of Variance tables (SQL)

Stats from Analysis can be obtained from the global statistics:

```
SELECT *
FROM TABLE(dbms_data_mining.get_model_details_global('<model_name>'))
order by global_detail_name;
```

The stats from the parameter table can be obtained through attribute level details

```
SELECT *
FROM TABLE(dbms_data_mining.get_model_details_glm('<model_name>'));
```

Copyright © 2020 Oracle and/or its affiliates.

Doc link

Association Rules – Market Basket Analysis

Apriori algorithm (Agrawal and Srikant 1994)

Finds frequent itemsets and generates association models

- Finds co-occurrence of items in large volumes of data: both transactional and relational

Produces rules

- Set of items in a transactional record implies the existence of another set of items
- Groups of items form rules if they pass a minimum threshold
- Thresholds include: how frequently they occur (support) and how often the consequent follows the antecedent (confidence)

Apriori algorithm is efficient, and scales well with respect to the number of transactions, number of items, and number of itemsets and rules produced

Copyright © 2020 Oracle and/or its affiliates.

Association (Market Basket Analysis)

Transactional Data and Rule Example

Input Data:

User ID	Movies Viewed
1	{Movie1, Movie2, Movie3}
2	{Movie1, Movie4}
3	{Movie1, Movie3}
4	{Movie2, Movie5, Movie6}
…	…
N	{Movie3, Movie4, Movie6}

Movie1 and Movie2 ➔ Movie3

with support of .12 and confidence .78

Copyright © 2020 Oracle and/or its affiliates.

Association Rules

Support and Confidence

User ID	Movies Viewed
1	{1, 2, 3}
2	{1, 4}
3	{1, 3}
4	{2, 5, 6}

Support ($A \rightarrow B$)
= P(AB)
= count (A & B) / totalCount

Confidence ($A \rightarrow B$)
= P(AB)/P(A)
= count (A & B) / count (A)

$1 \rightarrow 3$:

Support = 2/4 = 50%

Confidence = 2/3 = 66%

$3 \rightarrow 1$:

Support = 2/4 = 50%

Confidence = 2/2 = 100%

Copyright © 2020 Oracle and/or its affiliates.

ore.odmAssocRules

Association Rules

```
ore.odmAssocRules(formula,
                  data,
                  case.id.column,
                  item.id.column = NULL,
                  item.value.column = NULL,
                  min.support = 0.05,
                  min.confidence = 0.05,
                  max.rule.length = 2,
                  na.action = na.pass,
                  odm.setting = NULL)

## S3 method for class 'ore.odmAssocRules'
rules(object, ...)

## S3 method for class 'ore.odmAssocRules'
itemsets(object, ...)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

case.id.column

- Column name in 'data' that contains unique case identifiers

item.id.column

- Column in 'data' that contains item IDs. If NULL (default), 'data' treated as single-record case relational table, where each row considered a transaction and column values of that row are converted to items for that transaction; if specified, treated as transactional or multi-record case table where each row corresponds to an item in transaction, and model ignores any columns in 'data' other than item ID and item value.

item.value.column

- Column name in 'data' that contains the value of the item. (default: NULL)

min.support

- Numeric value that specifies the minimum support for rules in the model

min.confidence

- Numeric value that specifies the minimum confidence for rules in the model

max.rule.length

- Numeric value that specifies the maximum number of items in rule

Copyright © 2020 Oracle and/or its affiliates.

Association Rules – model object

ore.odmAssocRules object

- **name** – name of in-database model
- **settings** - data.frame of settings used to build model
- **attributes** - named 'vector' of the types of input item values
- **inputType:** The type of input data table. It is "trans","tranWithValue", or "relational" for a multi-record case table, a multi-record case table with the values specified, or a single-record case table, respectively
- **formula:** A formula specified by users

Copyright © 2020 Oracle and/or its affiliates.

Association Rules – model object

ore.itemsets object - returned by itemsets() that describes the property of each itemset

- **ITEMSET_ID:** numerical identifier associated with each itemset
- **NUMBER_OF_ITEMS:** number of items in the itemset
- **ITEMS:** names of items in the itemset
- **SUPPORT:** number of transactions containing this itemset

ore.rules object - returned by rules() that describes the property of each rule

- RULE_ID
- NUMBER_OF_ITEMS
- **LHS:** left hand side of rule (antecedent)
- **RHS:** right hand side of rule (consequent)
- SUPPORT
- CONFIDENCE
- LIFT

Copyright © 2020 Oracle and/or its affiliates.

ore.odmAssocRules

Association Rules

```
id <- c(1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3)
item <- c("b", "d", "e", "a", "b", "c", "e", "b", "c", "d", "e")
data.ore <- ore.push(data.frame(ID = id, ITEM = item))

ar.mod1 <- ore.odmAssocRules(~., data.ore, case.id.column =
"ID",item.id.column = "ITEM", min.support = 0.6, min.confidence =
0.6,max.rule.length = 3)

# Generate itemsets and rules of the model
itemsets <- itemsets(ar.mod1)
rules <- rules(ar.mod1)

# subsetting
sub.itemsets <- subset(itemsets, min.support=0.7, items=list("b"))
sub.rules <- subset(rules, min.confidence=0.7,
                    lhs=list("b", "c"))
library(arules)
# Convert the rules to the rules object in arules package
rules.arules <- ore.pull(rules)
inspect(rules.arules)
```

```
R>      inspect(rules.arules)
   lhs      rhs   support  confidence lift
1  {b} => {e} 1.0000000  1.0000000    1
2  {e} => {b} 1.0000000  1.0000000    1
3  {c} => {e} 0.6666667  1.0000000    1
4  {d,
    e} => {b} 0.6666667  1.0000000    1
5  {c,
    e} => {b} 0.6666667  1.0000000    1
6  {b,
    d} => {e} 0.6666667  1.0000000    1
7  {b,
    c} => {e} 0.6666667  1.0000000    1
8  {d} => {b} 0.6666667  1.0000000    1
9  {d} => {e} 0.6666667  1.0000000    1
10 {c} => {b} 0.6666667  1.0000000    1
11 {b} => {d} 0.6666667  0.6666667    1
12 {b} => {c} 0.6666667  0.6666667    1
13 {e} => {d} 0.6666667  0.6666667    1
14 {e} => {c} 0.6666667  0.6666667    1
15 {b,
    e} => {d} 0.6666667  0.6666667    1
16 {b,
    e} => {c} 0.6666667  0.6666667    1
```

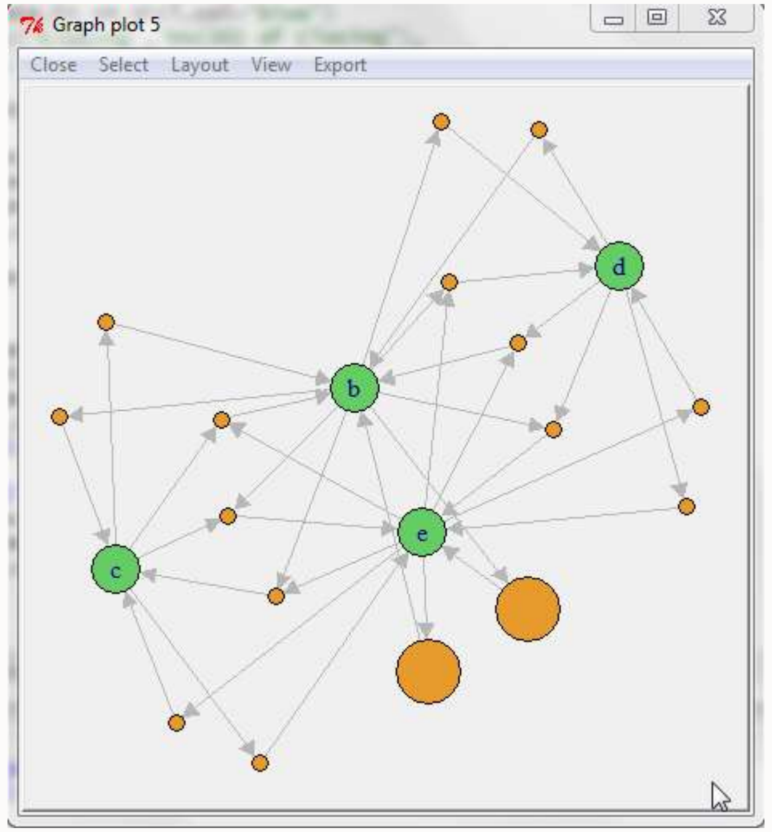
Copyright © 2020 Oracle and/or its affiliates.

ore.odmAssocRules

Association Rules

```
# Convert itemsets to the itemsets object in arules package
itemsets.arules <- ore.pull(itemsets)
inspect(itemsets.arules)

library(arulesViz)
plot(rules.arules, method = "graph",interactive=TRUE)
```

```
R>      inspect(itemsets.arules)
   items   support
1  {b}    1.0000000
2  {e}    1.0000000
3  {b,
    e}    1.0000000
4  {c}    0.6666667
5  {d}    0.6666667
6  {b,
    c}    0.6666667
7  {b,
    d}    0.6666667
8  {c,
    e}    0.6666667
9  {d,
    e}    0.6666667
10 {b,
    c,
    e}    0.6666667
11 {b,
    d,
    e}    0.6666667
```

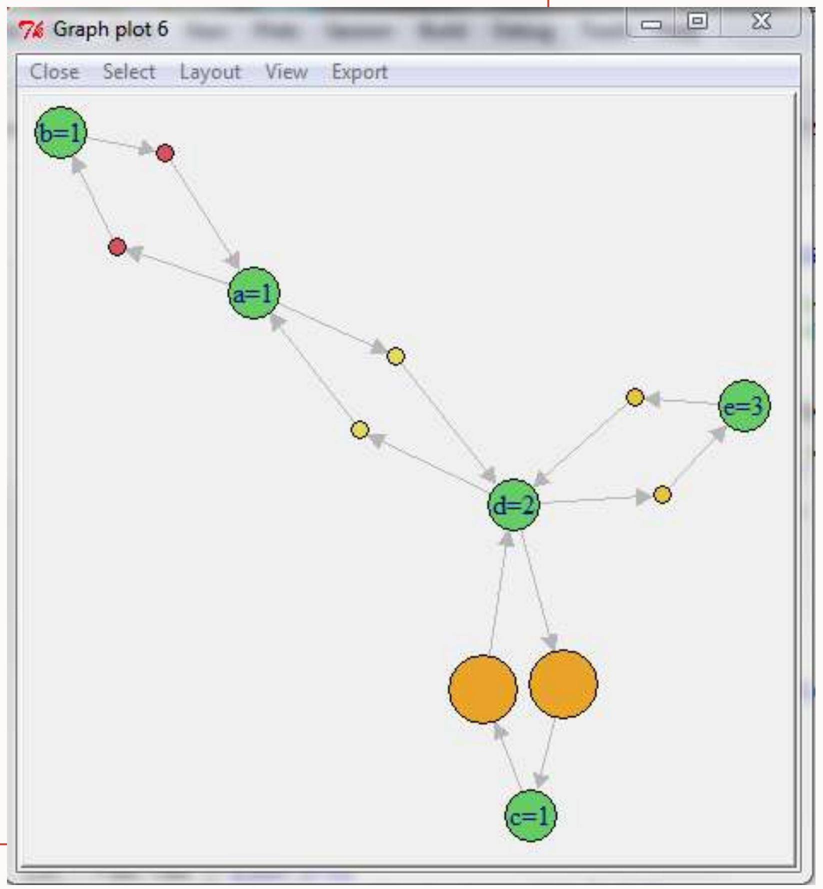


Copyright © 2020 Oracle and/or its affiliates.

ore.odmAssocRules – multi-record case with value

Association Rules

```
id <- c(1, 1, 2, 2, 3, 3, 3, 4, 4, 5, 5, 6, 6, 7, 8, 8, 9, 9, 10, 11, 12, 12, 13, 14, 15, 16, 17)
item <- c("a","b","a","b","a","c","d","c","d","c","d","c","d","d","d","a","d","e","a",
           "a","d","e","d","e","d","d","d")
value <- c(1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 3, 2, 1, 2, 3, 1, 2, 2, 3, 2, 3, 2, 3, 3)
data2.ore <- ore.push(data.frame("ID" = id, "ITEM" = item, "VALUE" = value))

ar.mod2 <- ore.odmAssocRules(~., data2.ore, case.id.column = "ID",
item.id.column = "ITEM", item.value.column = "VALUE", max.rule.length = 3)

rules <- rules(ar.mod2)
itemsets <- itemsets(ar.mod2)

itemsets.arules <- ore.pull(itemsets)
inspect(itemsets.arules)

rules.arules <- ore.pull(rules)

plot(rules.arules, method = "graph",interactive=TRUE)
```



Copyright © 2020 Oracle and/or its affiliates.

ore.odmAssocRules

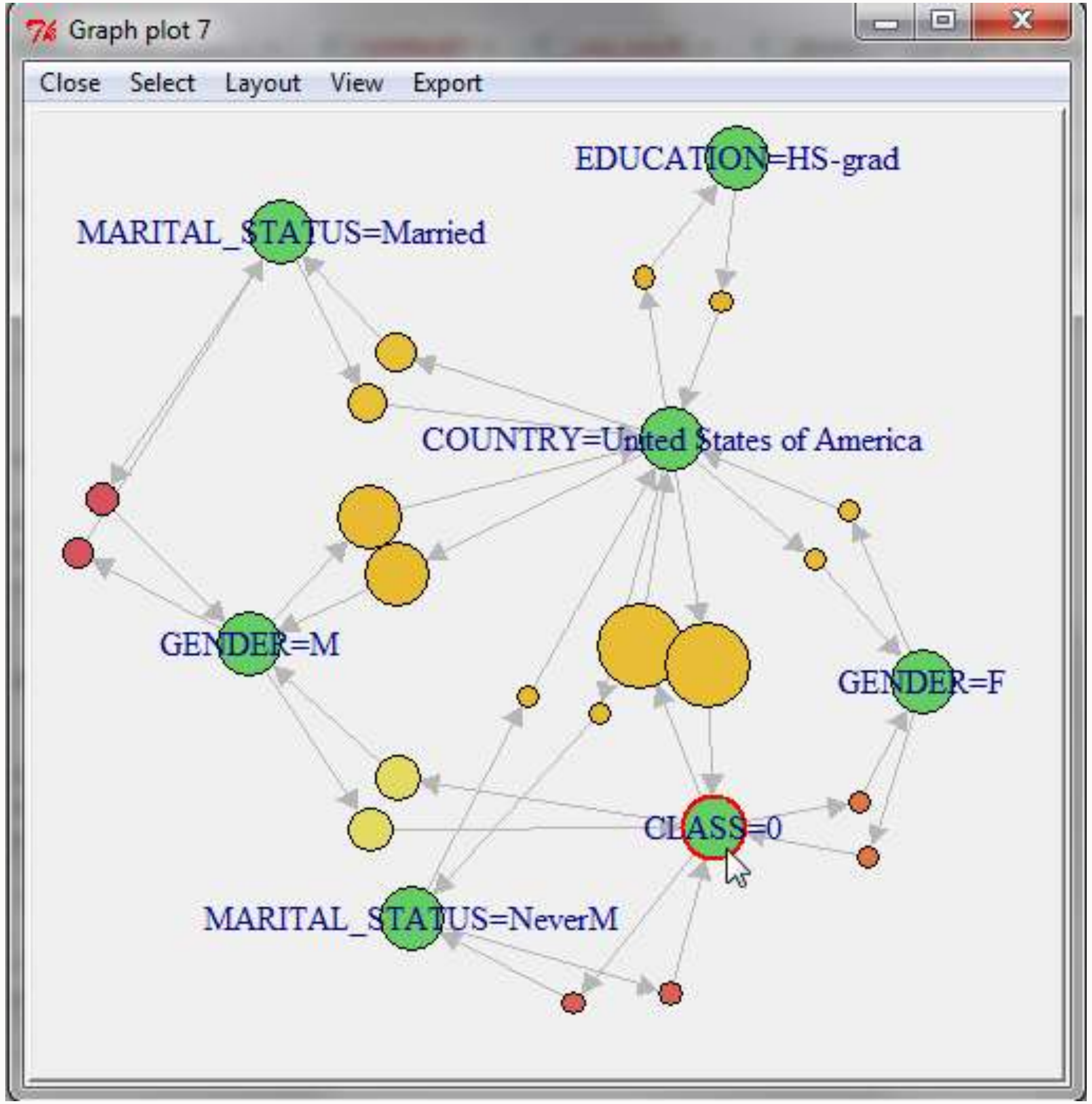
Association Rules

```
# Relational data in a single-record case table.
ar.mod3 <- ore.odmAssocRules(~., NARROW,
case.id.column = "ID",
   min.support=0.25, min.confidence=0.15,
max.rule.length = 2)

rules = rules(ar.mod3)
itemsets = itemsets(ar.mod3)

itemsets.arules <- ore.pull(itemsets)
inspect(itemsets.arules)

rules.arules <- ore.pull(rules)

plot(rules.arules, method =
"graph",interactive=TRUE)
```

```
R> inspect(itemsets.arules)
   items                                    support
1  {COUNTRY=United States of America} 0.8960000
2  {CLASS=0}                          0.7466667
3  {CLASS=0,
    COUNTRY=United States of America} 0.6646667
4  {GENDER=M}                         0.5866667
5  {COUNTRY=United States of America,
    GENDER=M}                         0.5273333
6  {MARITAL_STATUS=Married}           0.4133333
7  {CLASS=0,
    GENDER=M}                         0.3986667
8  {COUNTRY=United States of America,
    MARITAL_STATUS=Married}           0.3646667
9  {GENDER=M,
    MARITAL_STATUS=Married}           0.3140000
10 {GENDER=F}                         0.2806667
11 {EDUCATION=HS-grad}                0.2806667
12 {MARITAL_STATUS=NeverM}            0.2793333
13 {CLASS=0,
    MARITAL_STATUS=NeverM}            0.2633333
14 {COUNTRY=United States of America,
    EDUCATION=HS-grad}                0.2586667
15 {CLASS=1}                          0.2533333
16 {COUNTRY=United States of America,
    MARITAL_STATUS=NeverM}            0.2533333
17 {CLASS=0,
    GENDER=F}                         0.2520000
18 {COUNTRY=United States of America,
    GENDER=F}                         0.2520000
```

Copyright © 2020 Oracle and/or its affiliates.

```
plot(rules.arules, method = "graph",
     interactive = TRUE)
```



Copyright © 2020 Oracle and/or its affiliates.

Feature Extraction

Copyright © 2020 Oracle and/or its affiliates.

Singular Value Decomposition

Feature extraction algorithm

Orthogonal linear transformations capture the underlying variance of data by decomposing a rectangular matrix into three matrixes: U, D and V

Matrix D is a diagonal matrix and its singular values reflect the amount of data variance captured by the bases

Copyright © 2020 Oracle and/or its affiliates.

Singular Value Decomposition

ore.odmSVD

[Doc link]

Supports narrow data via Tall and Skinny solvers

Supports wide data via stochastic solvers

Provides eigensolvers for faster analysis with sparse data

Provides traditional SVD for more stable results

Copyright © 2020 Oracle and/or its affiliates.

ore.odmSVD

Singular Value Decomposition

```
ore.odmSVD(formula,
           data,
           auto.data.prep = TRUE,
           na.action = na.pass,
           odm.setting = NULL,
           ctx.setting = NULL)

features(object,...)

feature_compare(object,
                newdata,
                compare.cols = NULL,
                supplemental.cols = NULL)
```

```
predict(object,
        newdata,
        supplemental.cols = NULL,
        type = c("class","raw"),
        na.action = na.pass,...)

u(object)

v(object)

d(object)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

num.centers – number of clusters to create, > 1, default NULL – system determined

auto.data.prep – default TRUE

odm.setting – A list to specify Oracle Data Mining parameter settings. This argument is applicable to building a model in Database 12.2 or later. Each list element's name and value refer to the parameter setting name and value, respectively. The setting values must be numeric or string. To perform text mining, parameter `ODMS_TEXT_POLICY_NAME` must be set to a text policy name. When parameter `ODMS_PARTITION_COLUMNS` is set to the name(s) of the partition column(s), a partition model with a sub-model in each partition is created from the input data

ctx.setting – A list to specify Oracle Text attribute-specific settings. This argument is applicable to building model in Database 12.2 or later. The name of each list element refers to the text column while the list value specifies the text transformation.

See ODM documentation for specific settings options.

Copyright © 2020 Oracle and/or its affiliates.

SVD – model object

ore.odmSVD object

- **name** name of model in database
- **settings** data.frame with settings used to build model
- **attributes** data.frame of variable/columns used to build model
- **formula** formula used to build the model
- **call** specific invocation of the function with arguments

Copyright © 2020 Oracle and/or its affiliates.

ore.odmSVD

Singular Value Decomposition

```
IRIS <- ore.push(cbind(ID = seq_along(iris[[1L]]), iris))

svd.mod <- ore.odmSVD(~. -ID, IRIS)
summary(svd.mod)
d(svd.mod)
v(svd.mod)
head(predict(svd.mod, IRIS, supplemental.cols = "ID"))

svd.pmod <- ore.odmSVD(~. -ID, IRIS,
              odm.setting = list(odms_partition_columns = "Species"))
summary(svd.pmod)
d(svd.pmod)
v(svd.pmod)
head(predict(svd.pmod, IRIS, supplemental.cols = "ID"))
```

Copyright © 2020 Oracle and/or its affiliates.

Non-negative Matrix Factorization

State-of-the-art algorithm for Feature Extraction

Dimensionality reduction technique

- Creates new features of existing attributes
- Compare to AI which reduces attributes by taking a subset
- NMF derives fewer new “features” taking into account interactions among original attributes

Supports text mining, life sciences, marketing applications

Copyright © 2020 Oracle and/or its affiliates.

NMF, intuitively…

Useful where there are many attributes

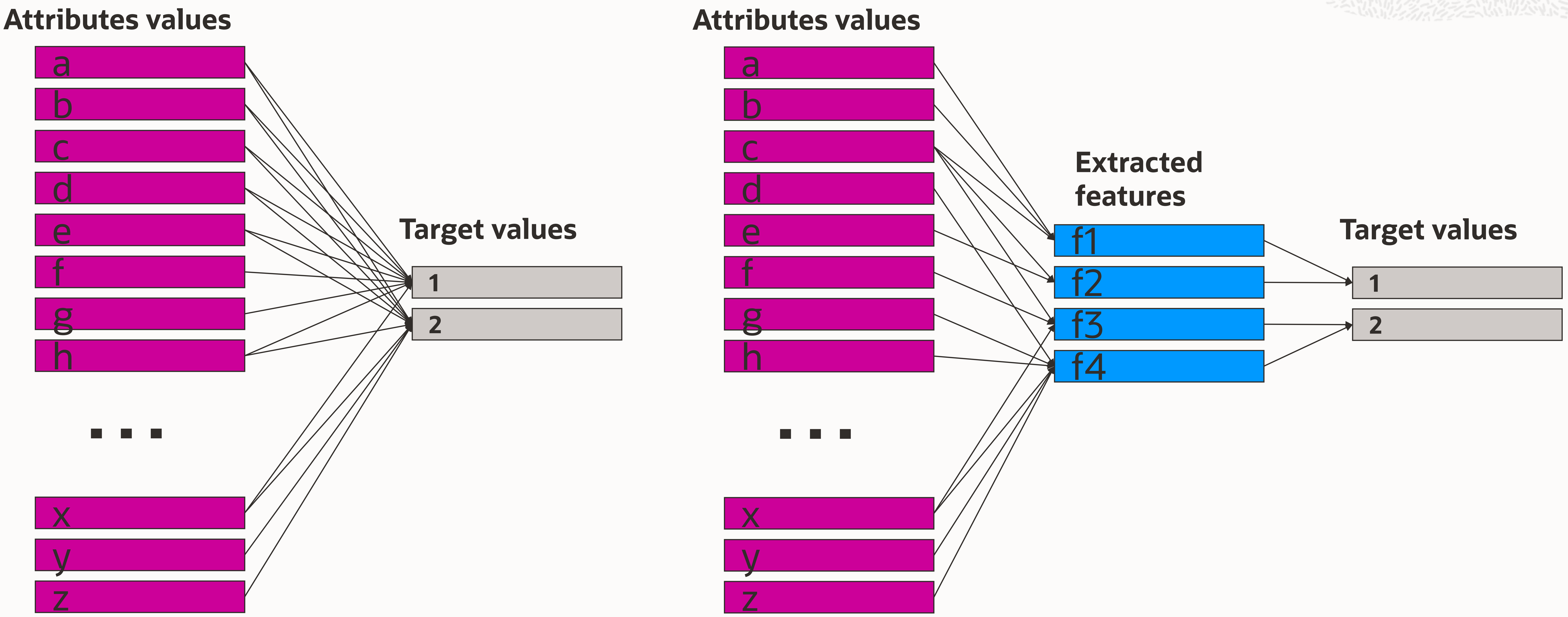
- Each has weak predictability, even ambiguous
- But when taken in combination, produce meaningful patterns, topics, or themes

Example: Text

- Same word can predict different documents
  e.g., “hike” can be applied to the outdoors or interest rates
- NMF introduces context which is essential for predictive power
  e.g., “hike” + “mountain” -> “outdoors sports”
  “hike” + “interest” -> “interest rates”

Copyright © 2020 Oracle and/or its affiliates.

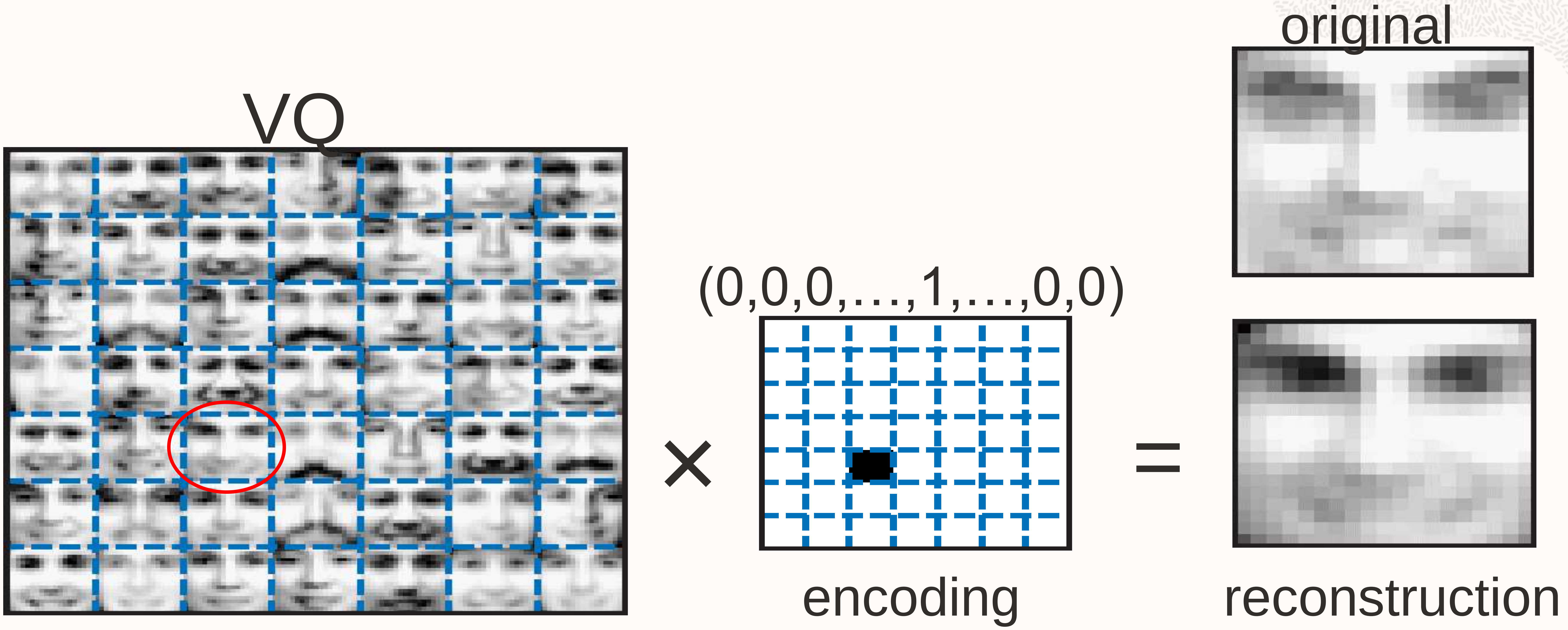
Conceptual view…



Copyright © 2020 Oracle and/or its affiliates.

Feature Extraction

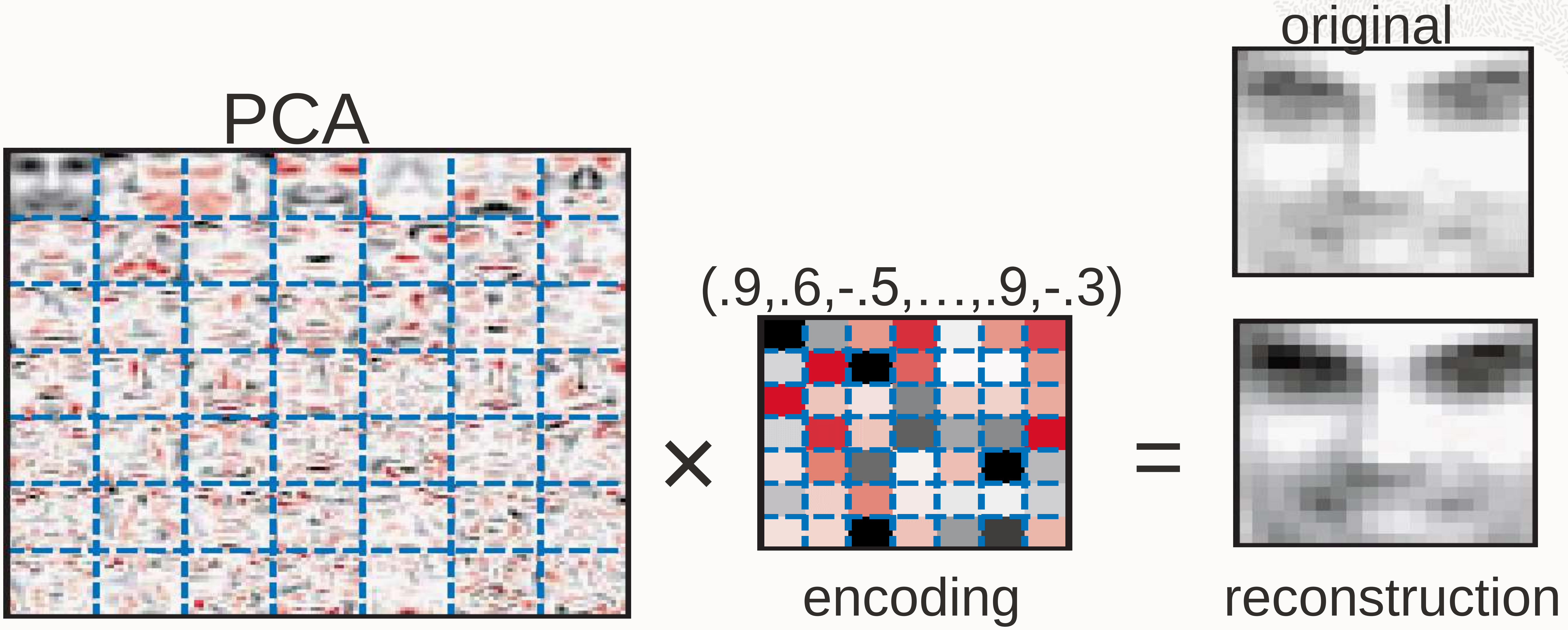
Face representation with Vector Quantization



Copyright © 2020 Oracle and/or its affiliates.

Feature Extraction

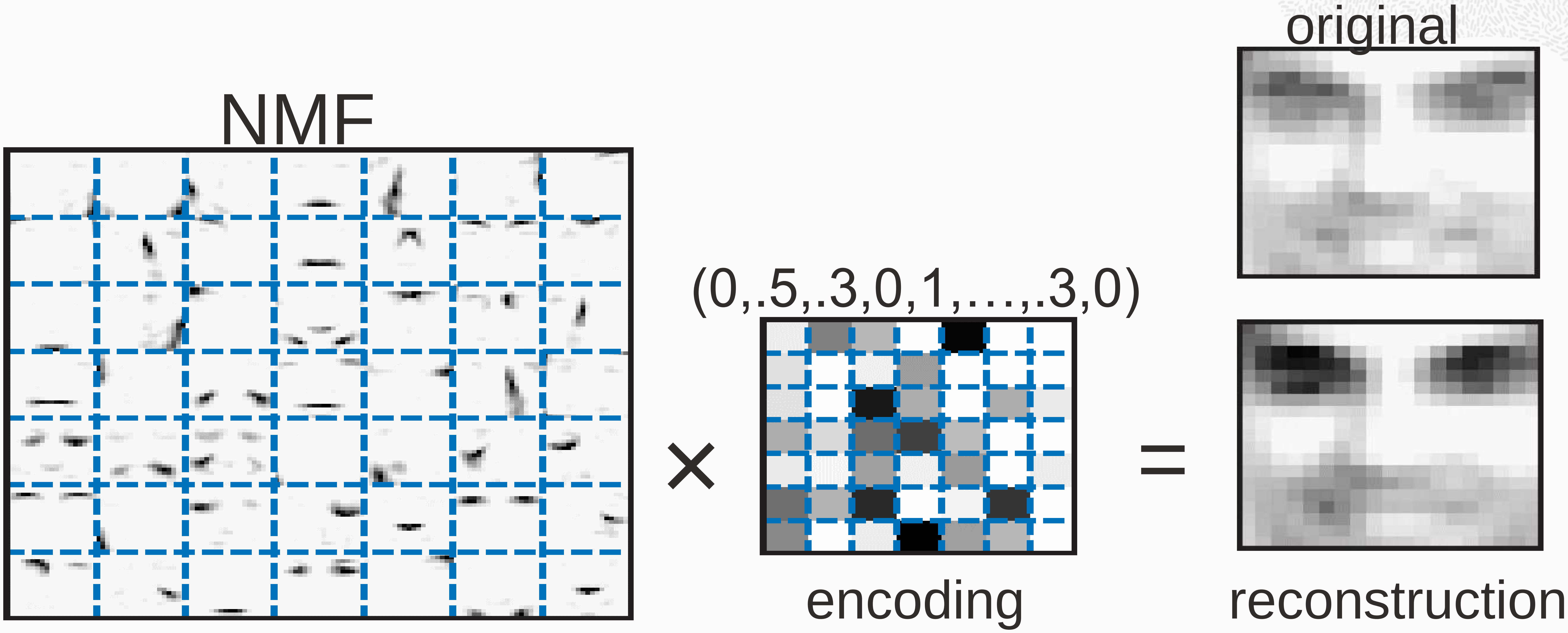
Face representation with Principal Component Analysis



Copyright © 2020 Oracle and/or its affiliates.

Feature Extraction

Face representation with NMF



Copyright © 2020 Oracle and/or its affiliates.

ore.odmNMF

Non-negative Matrix Factorization

Doc link

```
ore.odmNMF(formula,
           data,
           auto.data.prep = TRUE,
           num.features = NULL,
           conv.tolerance = NULL,
           num.iter = NULL,
           rand.seed = NULL,
           nonnegative.scoring = TRUE,
           na.action = na.pass,
           odm.setting = NULL,
           ctx.setting = NULL)

predict(object,
        newdata,
        supplemental.cols = NULL,
        type = c("class","raw"),
        na.action = na.pass,...)
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

num.features – number of features to be extracted

conv.tolerance – convergence tolerance

num.iter – maximum number of iterations

rand.seed – random seed

nonnegative.scoring – non-negative values allowed in scoring

Copyright © 2020 Oracle and/or its affiliates.

ore.odmNMF

Non-negative Matrix Factorization

```
training.set <- ore.push(npk[1:18, c("N","P","K")])
scoring.set <- ore.push(npk[19:24, c("N","P","K")])

nmf.mod <- ore.odmNMF(~., training.set, num.features = 3)
features(nmf.mod)
summary(nmf.mod)
predict(nmf.mod, scoring.set)
```

```
R> features(nmf.mod)
   FEATURE_ID ATTRIBUTE_NAME ATTRIBUTE_VALUE  COEFFICIENT
1           1              K               0 3.723468e-01
2           1              K               1 1.761670e-01
3           1              N               0 7.469067e-01
4           1              N               1 1.085058e-02
5           1              P               0 5.730082e-01
6           1              P               1 2.797865e-02
7           2              K               0 4.107375e-01
8           2              K               1 2.193757e-01
9           2              N               0 8.065393e-03
10          2              N               1 8.569538e-01
11          2              P               0 4.005661e-01
12          2              P               1 4.124996e-02
13          3              K               0 1.918852e-01
14          3              K               1 3.311137e-01
15          3              N               0 1.547561e-01
16          3              N               1 1.283887e-01
17          3              P               0 9.791965e-06
18          3              P               1 9.113922e-01
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmNMF

Non-negative Matrix Factorization

```
R> predict(nmf.mod, scoring.set)
         '1'       '2'        '3' FEATURE_ID
19 0.1972489 1.2400782 0.03280919          2
20 0.7298919 0.0000000 1.29438165          3
21 0.1972489 1.2400782 0.03280919          2
22 0.0000000 1.0231268 0.98567623          2
23 0.7298919 0.0000000 1.29438165          3
24 1.5703239 0.1523159 0.00000000          1
```

```
R> summary(nmf.mod)

Call:
ore.odmNMF(formula = ~., data = training.set, num.features = 3)

Settings:
                                                value
feat.num.features                                   3
nmfs.conv.tolerance                               .05
nmfs.nonnegative.scoring nmfs.nonneg.scoring.enable
nmfs.num.iterations                                50
nmfs.random.seed                                   -1
prep.auto                                          on

Features:
   FEATURE_ID ATTRIBUTE_NAME ATTRIBUTE_VALUE  COEFFICIENT
1           1              K               0 3.723468e-01
2           1              K               1 1.761670e-01
3           1              N               0 7.469067e-01
4           1              N               1 1.085058e-02
5           1              P               0 5.730082e-01
6           1              P               1 2.797865e-02
7           2              K               0 4.107375e-01
8           2              K               1 2.193757e-01
9           2              N               0 8.065393e-03
10          2              N               1 8.569538e-01
11          2              P               0 4.005661e-01
12          2              P               1 4.124996e-02
13          3              K               0 1.918852e-01
14          3              K               1 3.311137e-01
15          3              N               0 1.547561e-01
16          3              N               1 1.283887e-01
17          3              P               0 9.791965e-06
18          3              P               1 9.113922e-01
```

Copyright © 2020 Oracle and/or its affiliates.

Explicit Semantic Analysis (ESA)

Oracle Advanced Analytics 12.2+

Copyright © 2020 Oracle and/or its affiliates.

Explicit Semantic Analysis (ESA)

In NLP and information retrieval, ESA is a **vectorial representation** of text (individual words or entire documents) that uses a document corpus as a knowledge base

- A word is represented as a column vector in the TF-IDF matrix of the text corpus
- A document (string of words) is represented as the centroid of the vectors representing its words

Text corpus often is English Wikipedia, though other corpora can be used

Designed to **improve text categorization**

- Computes "semantic relatedness" using cosine similarity between aforementioned vectors, collectively interpreted as **a space of "concepts explicitly defined and described by humans“**
- Wikipedia articles are equated with concepts

The name "explicit semantic analysis" contrasts with latent semantic analysis (LSA), because use of a knowledge base makes possible to **assign human-readable labels** to concepts comprising the vector space

Copyright © 2020 Oracle and/or its affiliates.

Explicit Semantic Analysis (ESA)

Data

- Text documents
- Data with mixed set of columns, i.e., text + categorical + numerical

Examples

- Calculate semantic similarity between text documents or between mixed data
- Explicit topic modeling for text

Copyright © 2020 Oracle and/or its affiliates.

Case 1: Calculate semantic similarity between text documents or between mixed data

Requires Wikipedia or another encyclopedic source to create a model

Model source data should cover all aspects of language usage

- E.g., number of articles in the source data should be comparable to dictionary size
- A dictionary of size 200K is often sufficient
- Ideally source data is ~orthogonal, i.e. without overlapping articles

Copyright © 2020 Oracle and/or its affiliates.

Case 1: Example

The following two paragraphs score a high similarity at 0.695 according to a Wikipedia-based ESA model:

- The Securities and Exchange Commission sued Tesla's CEO on Thursday for making 'false and misleading' statements to investors. It's asking a federal judge to prevent Musk from serving as an officer or a director of a public company, among other penalties. The complaint hinges on a tweet Musk sent on August 7 about taking Tesla private. 'Am considering taking Tesla private at $420,' Musk said. 'Funding secured.' The SEC said he had not actually secured the funding. 'In truth and in fact, Musk had not even discussed, much less confirmed, key deal terms, including price, with any potential funding source,' the SEC said in its complaint. That tweet, and subsequent tweets from Musk over the next three hours, caused 'significant confusion and disruption in the market for Tesla's stock,' as well as harm to investors, the SEC said. On the day of Musk's tweet, Tesla's stock shot up nearly 9%. It has declined substantially since then.
- The Securities and Exchange Commission filed a lawsuit Thursday against Elon Musk, the chief executive of Tesla, accusing him of making false public statements with the potential to hurt investors. The lawsuit, filed in federal court in New York, seeks to bar Mr. Musk from serving as an executive or director of publicly traded companies. Tesla, the electric-car maker of which Mr. Musk was a co-founder, is publicly traded. The suit relates to an Aug. 7 Twitter post by Mr. Musk, in which he said he had 'funding secured' to convert Tesla into a private company. The S.E.C. said Mr. Musk 'knew or was reckless in not knowing' that his statements were false or misleading. 'In truth and in fact, Musk had not even discussed, much less confirmed, key deal terms, including price, with any potential funding source,' the S.E.C. said in its lawsuit..‘

In contrast, similarity between the first paragraph and this paragraph is only 0.051:

- If humans had lived 200 million years ago, they would have marveled at the largest dinosaur of its time. It's name means 'a giant thunderclap at dawn.' The recently discovered fossil of a new dinosaur species in South Africa revealed a relative of the brontosaurus that weighed 26,000 pounds, about double the size of a large African elephant. The researchers have named it Ledumahadi mafube, which is Sesotho for 'a giant thunderclap at dawn.' Sesotho is an official South African language indigenous to the part of the country where the dinosaur was found. 'The name reflects the great size of the animal as well as the fact that its lineage appeared at the origins of sauropod dinosaurs,' said Jonah Choiniere, study author and paleontology professor at the University of the Witwatersrand in Johannesburg, South Africa. 'It honors both the recent and ancient heritage of southern Africa.'

Copyright © 2020 Oracle and/or its affiliates.

Case 2: Explicit topic modeling for text

Discover the most relevant topics for a given text document

- Not really applicable to mixed data

Using Wikipedia as the model source data is typical

Explicit topic modeling benefits from domain-specific data

- E.g., medicine, biology, physics and all other science branches

Requires that data source is encyclopedic for the selected domain

- If domain topic coverage is insufficient, results will be poor

Copyright © 2020 Oracle and/or its affiliates.

Case 2: Example

"The more things change… Yes, I'm inclined to agree, especially with regards to the historical relationship between stock prices and bond yields. The two have generally traded together, rising during periods of economic growth and falling during periods of contraction. Consider the period from 1998 through 2010, during which the U.S. economy experienced two expansions as well as two recessions: Then central banks came to the rescue. Fed Chairman Ben Bernanke led from Washington with the help of the bank's current $3.6T balance sheet. He's accompanied by Mario Draghi at the European Central Bank and an equally forthright Shinzo Abe in Japan. Their coordinated monetary expansion has provided all the sugar needed for an equities moonshot, while they vowed to hold global borrowing costs at record lows"

Top topics (**concepts, people, organizations, events**) discovered by ESA using Wikipedia as model source data

- *Recession, Ben Bernanke , Lost Decade Japan, Mario Draghi, Quantitative easing, Long Depression, Great Recession, Federal Open Market Committee, Bank of Canada, Monetary policy, Japanese asset price bubble, Money supply, Great Depression, Central bank, Federal Reserve System*

If instead of using the entire Wikipedia, we limit ourselves to the source dataset comprised of concepts only, this result would translate to:

- *Recession, Quantitative easing, Monetary policy, Money supply, Central bank, Federal Reserve System*

Copyright © 2020 Oracle and/or its affiliates.

ESA vs. LDA (Latent Dirichlet Allocation)

ESA is more interpretable than LDA

Topics discovered by LDA are *latent*, meaning difficult to interpret

- Topics are defined by their keywords, i.e., they have no names, no abstract descriptions
- To give meaning to topics, keywords can be extracted by LDA
- Definitions solely based on keywords are fuzzy, and keywords for different topics usually overlap
- Extracted keywords can be just generic words
- Set of automatically extracted keywords for a topic does not map to a convenient English topic name

Biggest problem with LDA: set of topics is fluid

- Topic set changes with any changes to the training data
- Any modification of training data changes topic boundaries
- → topics cannot be mapped to existing knowledge base or topics understood by humans if training data not static
- Training data is almost never static

ESA discovers topics from a given set of topics in a knowledge base

- Topics are defined by humans → topics are well understood.
- Topic set of interest can be selected and augmented if necessary → full control of the selection of topics
- Set of topics can be geared toward a specific task, .e.g., knowledge base for topic modeling of online messages possibly related to terrorist activities, which is different than one for topic modeling of technical reports from academia
- Can combine multiple knowledge bases, each with its own topic set, which may or may not overlap
- Topic overlapping does not affect ESA's capability to detect relevant topics

Copyright © 2020 Oracle and/or its affiliates.

ore.odmESA

Explicit Semantic Analysis

Doc link

```
ore.odmESA(formula,
           data,
           auto.data.prep = TRUE,
           na.action = na.pass,
           odm.setting = NULL,
           ctx.setting = NULL)

features(object,...)

feature_compare(object, newdata, compare.cols = NULL, supplemental.cols = NULL)

predict(object,
        newdata,
        supplemental.cols = NULL,
        type = c("class","raw"),
        na.action = na.pass,...)
}
```

Copyright © 2020 Oracle and/or its affiliates.

Basic Argument Concepts

odm.setting – A list to specify Oracle Data Mining parameter settings. This argument is applicable to building a model in Database 12.2 or later. Each list element's name and value refer to the parameter setting name and value, respectively. The setting values must be numeric or string. Parameter CASE_ID_COLUMN_NAME must specify the name of the column containing unique identifier. Parameter ODMS_TEXT_POLICY_NAME specifies the name of a valid Oracle text policy used for text mining. When parameter ODMS_PARTITION_COLUMNS is set to the names of the partition columns, then a partition model with sub-model in each partition is created from the input data.

ctx.setting – A list to specify Oracle Text attribute-specific settings. This argument is applicable to building model in Database 12.2 or later. The name of each list element refers to the text column while the list value specifies the text transformation.

(See ODM documentation for specific settings options.)

Copyright © 2020 Oracle and/or its affiliates.

ESA – model object

ore.odmESA object

- **name** name of model in database
- **settings** data.frame with settings used to build model
- **attributes** data.frame of variable/columns used to build model
- **formula** formula used to build the model
- **call** specific invocation of the function with arguments

Copyright © 2020 Oracle and/or its affiliates.

ore.odmESA

Explicit Semantic Analysis

```
title <- c('Aids in Africa: Planning for a long war',
           'Mars rover maneuvers for rim shot',
           'Mars express confirms presence of water at Mars south pole',
           'NASA announces major Mars rover finding',
           'Drug access, Asia threat in focus at AIDS summit',
           'NASA Mars Odyssey THEMIS image: typical crater',
           'Road blocks for Aids')
ESA_TEXT <- ore.push(data.frame(CUST_ID = seq(length(title)),
                                TITLE = title))

# create text policy (CTXSYS.CTX_DDL privilege is required)
ore.exec("begin ctx_ddl.create_policy('ESA_TXTPOL'); end;")
```

Copyright © 2020 Oracle and/or its affiliates.

ore.odmESA

Explicit Semantic Analysis

```
esa.mod <- ore.odmESA(~., data = ESA_TEXT,
    odm.setting = list(case_id_column_name = "CUST_ID",
                       ODMS_TEXT_POLICY_NAME = "ESA_TXTPOL",
                       ESAS_MIN_ITEMS = 1),
    ctx.setting = list(TITLE = c("MIN_DOCUMENTS:1", "MAX_FEATURES:3")))
esa.mod
class(esa.mod)
summary(esa.mod)
settings(esa.mod)
predict(esa.mod, ESA_TEXT, type = "class", supplemental.cols = "CUST_ID")

ore.exec("begin ctx_ddl.drop_policy('ESA_TXTPOL'); end;")
```

Copyright © 2020 Oracle and/or its affiliates.

OREmodels Package

Copyright © 2020 Oracle and/or its affiliates.

OREmodels Algorithms

Algorithm	Main R Function
Linear Regression	ore.lm
Stepwise Linear Regression	ore.stepwise
Generalized Linear Models	ore.glm
Feedforward Neural Networks	ore.neural
Random Forest	ore.randomForest
Singular Value Decomposition	svd *overloaded*
Principal Component Analysis	prcomp *overloaded* princomp *overloaded*

Copyright © 2020 Oracle and/or its affiliates.

ore.lm and ore.stepwise

Overview

ore.lm performs least squares regression

ore.stepwise performs stepwise least squares regression with marginal t-tests for variable selection – similar to SAS

Uses database data represented by *ore.frame* objects

In-database algorithm

- Estimates model using block update QR decomposition with column pivoting
- Once coefficients have been estimated, a second pass of the data estimates model-level statistics
- If collinear terms in data, ore.lm and ore.stepwise will not estimate coefficient values for the collinear set of terms
- For ore.stepwise, this collinear set of terms will be excluded throughout the procedure

Copyright © 2020 Oracle and/or its affiliates.

lm

For comparison with ore.lm

```
# Fit full model
fit1 <- lm(Employed ~ ., data = longley)
summary(fit1)
```

Coefficient *Armed.Forces* significant at p < .001 indicates for a 1 unit increase in *Armed.Forces*, *Employed* decreases by 0.01 units when all other predictors held constant

Multiple R-squared of 0.9955 indicates the model accounts for 99.55% of the variance in the target

Adjusted R-squared takes into account the number of predictors to
account for chance improvement of R-squared simply be increasing
number of predictors

Residual standard error is the average error in predicting the target

F-statistic indicates if predictors predict target beyond chance

```
R> fit1 <- lm(Employed ~ ., data = longley)
R> summary(fit1)

Call:
lm(formula = Employed ~ ., data = longley)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.41011 -0.15767 -0.02816  0.10155  0.45539 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -3.482e+03  8.904e+02  -3.911 0.003560 ** 
GNP.deflator  1.506e-02  8.492e-02   0.177 0.863141    
GNP          -3.582e-02  3.349e-02  -1.070 0.312681    
Unemployed   -2.020e-02  4.884e-03  -4.136 0.002535 ** 
Armed.Forces -1.033e-02  2.143e-03  -4.822 0.000944 ***
Population   -5.110e-02  2.261e-01  -0.226 0.826212    
Year          1.829e+00  4.555e-01   4.016 0.003037 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3049 on 9 degrees of freedom
Multiple R-squared: 0.9955,     Adjusted R-squared: 0.9925 
F-statistic: 330.3 on 6 and 9 DF,  p-value: 4.984e-10
```

Copyright © 2020 Oracle and/or its affiliates.

ore.lm

```
# To limit overhead caused by parallelism
options(ore.parallel=1)

LONGLEY <- ore.push(longley)

# Fit full model
oreFit1 <- ore.lm(Employed ~ ., data = LONGLEY)
summary(oreFit1)
```

Since data is small, turn off parallelism by setting ore.parallel to 1

```
R> LONGLEY <- ore.push(longley)
R>
R> # Fit full model
R> oreFit1 <- ore.lm(Employed ~ ., data = LONGLEY)
R> summary(oreFit1)

Call:
ore.lm(formula = Employed ~ ., data = LONGLEY)

Residuals:
     Min       1Q   Median       3Q      Max
-0.41011 -0.15980 -0.02816  0.15681  0.45539

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.482e+03  8.904e+02  -3.911 0.003560 **
GNP.deflator  1.506e-02  8.492e-02   0.177 0.863141
GNP          -3.582e-02  3.349e-02  -1.070 0.312681
Unemployed   -2.020e-02  4.884e-03  -4.136 0.002535 **
Armed.Forces -1.033e-02  2.143e-03  -4.822 0.000944 ***
Population   -5.110e-02  2.261e-01  -0.226 0.826212
Year          1.829e+00  4.555e-01   4.016 0.003037 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3049 on 9 degrees of freedom
Multiple R-squared: 0.9955,     Adjusted R-squared: 0.9925
F-statistic: 330.3 on 6 and 9 DF,  p-value: 4.984e-10
```

Copyright © 2020 Oracle and/or its affiliates.

lm and ore.lm results side-by-side

They're identical

```
R> fit1 <- lm(Employed ~ ., data = longley)
R> summary(fit1)

Call:
lm(formula = Employed ~ ., data = longley)

Residuals:
     Min       1Q   Median       3Q      Max
-0.41011 -0.15767 -0.02816  0.10155  0.45539

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.482e+03  8.904e+02  -3.911 0.003560 **
GNP.deflator  1.506e-02  8.492e-02   0.177 0.863141
GNP          -3.582e-02  3.349e-02  -1.070 0.312681
Unemployed   -2.020e-02  4.884e-03  -4.136 0.002535 **
Armed.Forces -1.033e-02  2.143e-03  -4.822 0.000944 ***
Population   -5.110e-02  2.261e-01  -0.226 0.826212
Year          1.829e+00  4.555e-01   4.016 0.003037 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3049 on 9 degrees of freedom
Multiple R-squared: 0.9955,     Adjusted R-squared: 0.9925
F-statistic: 330.3 on 6 and 9 DF,  p-value: 4.984e-10
```

```
R> LONGLEY <- ore.push(longley)
R>
R> # Fit full model
R> oreFit1 <- ore.lm(Employed ~ ., data = LONGLEY)
R> summary(oreFit1)

Call:
ore.lm(formula = Employed ~ ., data = LONGLEY)

Residuals:
     Min       1Q   Median       3Q      Max
-0.41011 -0.15980 -0.02816  0.15681  0.45539

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.482e+03  8.904e+02  -3.911 0.003560 **
GNP.deflator  1.506e-02  8.492e-02   0.177 0.863141
GNP          -3.582e-02  3.349e-02  -1.070 0.312681
Unemployed   -2.020e-02  4.884e-03  -4.136 0.002535 **
Armed.Forces -1.033e-02  2.143e-03  -4.822 0.000944 ***
Population   -5.110e-02  2.261e-01  -0.226 0.826212
Year          1.829e+00  4.555e-01   4.016 0.003037 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3049 on 9 degrees of freedom
Multiple R-squared: 0.9955,     Adjusted R-squared: 0.9925
F-statistic: 330.3 on 6 and 9 DF,  p-value: 4.984e-10
```

Copyright © 2020 Oracle and/or its affiliates.

Other functions on ore.lm model

summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)

- Return the call, residuals, coefficients, and various statistics

**predict(object, newdata, se.fit = FALSE, scale = NULL, df = Inf,
interval = c("none", "confidence", "prediction"), level = 0.95, type = c("response", "terms"),
terms = NULL, na.action = na.pass, pred.var = NULL, weights = NULL, ...)**

vcov(object, ...)

- Returns the variance-covariance matrix of the main parameters of a fitted model object

logLik(object, ...)

- Returns object of class **logLik** – a number with at least one attribute, "df"(**d**egrees of **f**reedom), giving number of (estimated) parameters in model

hatvalues(model, ...)

- Returns measure of high leverage for observations. Observations with value > 2x or 3x average hat value should be examined for possible removal. Average hat value = p / n, where p is number of parameters (including intercept) and n is number of observations

add1(object, scope, scale = 0, test = c("none", "Chisq", "F"), x = NULL, k = 2, ...)

drop1(object, scope, scale = 0, all.cols = TRUE, test = c("none", "Chisq", "F"), k = 2, ...)

- Compute all single terms in scope argument that can be added to / dropped from model, fit those models and compute table of changes in fit

Copyright © 2020 Oracle and/or its affiliates.

Other functions on ore.lm model

anova(object,…)

coef(object, ...) & coefficients(object, ...)

- Return the coefficients for the model

confint(object, parm, level = 0.95, ...)

- Return the confidence interval of the coefficients

deviance(object, ...)

- Returns the deviance of the model

extractAIC(fit, scale, k = 2, ...)

- Returns the (generalized) **A**kaike **I**nformation **C**riterion for a fitted parametric model

fitted(object, ...) & fitted.values(object, ...)

- Returns the predicted values on each of the training data observations/rows

formula(x, ...)

- Returns the formula used to specify the model

model.frame(formula, ...)

- Returns the training data used to build the model

nobs(object, ...)

- Returns the number of observations in the training data

resid(object, ...) & residuals(object, ...)

- Returns the residual values from the predictions (predicted – actual) for each observation

plot(object,…)

- Produces diagnostic plots to assess model fit
- Plot of residuals against fitted values, a Scale-Location plot of *sqrt(| residuals |)* against fitted values, a Normal Q-Q plot, a plot of Cook's distances versus row labels, a plot of residuals against leverages, and a plot of Cook's distances against leverage/(1-leverage)

Copyright © 2020 Oracle and/or its affiliates.

Stepwise Regression: ore.stepwise

Motivation

Automatically selects predictive variables

Produces models with fewer terms

Enable handling data with complex patterns

- Even for relatively small data sets (e.g., < 1M rows) R may not yield satisfactory results

Increases performance

- Side benefit of handling complex patterns is to dramatically boost performance
- No need to pull data into memory from database
- Leverage more powerful database machine

Provide a stepwise regression that maps to SAS PROC REG

- Uses marginal t-tests for variable selection as opposed to AIC, which is used for R's step() function. Note R's step function can be used with ore.lm.

Copyright © 2020 Oracle and/or its affiliates.

ore.stepwise – parameters

```
ore.stepwise(formula, data, scope,
             direction = c("both", "backward", "forward", "alternate", "none"),
             add.p = 0.50, drop.p = 0.10, nbest = 1, steps = 1000,
             contrasts = NULL, xlev = NULL, ...)
```

scope – range of models to examine, either single formula object, or list containing lower and upper formula object elements

direction – The stepwise search mode; one of '"both"' (first try to add a term using the 'add.p' argument value and then try repeatedly to drop terms using the 'drop.p' argument value), '"backward"', '"forward"', '"alternate"' (similar to '"both"' but only one drop is attempted per add attempt) or '"none"' with a default of '"both"'

add.p – F-test p-value threshold for adding term to model

drop.p – F-test p-value threshold for dropping term from model

nbest – number of best models to report at each step

steps – maximum number of steps

contrasts – named list to be supplied to the contrasts.arg argument of model.matrix

xlev – a named list of character vectors specifying the levels for each ore.factor variable

Copyright © 2020 Oracle and/or its affiliates.

ore.stepwise – example

```
LONGLEY <- ore.push(longley)

# Using ore.stepwise
oreStep1 <-
  ore.stepwise(Employed ~ .^2, data = LONGLEY,
                add.p = 0.1, drop.p = 0.1)
oreStep1
```

```
# Using R step with ore.lm
oreStep2 <-
  step(ore.lm(Employed ~ 1, data = LONGLEY),
        scope = terms(Employed ~ .^2, data = LONGLEY))
oreStep2
```

Build model with interaction terms

Copyright © 2020 Oracle and/or its affiliates.

ore.stepwise – results

```
R> oreStep1 <-
+   ore.stepwise(Employed ~ .^2, data = LONGLEY,
+                add.p = 0.1, drop.p = 0.1)
R> oreStep1

Aliased:
[1] "Unemployed:Armed.Forces" "Unemployed:Population"   "Unemployed:Year"         "Armed.Forces:Population
"
[5] "Armed.Forces:Year"       "Population:Year"

Steps:
                      Add Drop     RSS Rank
1 GNP.deflator:Unemployed <NA> 384.426    2
2                GNP:Year <NA> 218.957    3
3        GNP.deflator:GNP <NA> 130.525    4
4 GNP.deflator:Population <NA>  81.211    5
5        GNP:Armed.Forces <NA>  18.244    6
6                    Year <NA>  14.492    7

Call:
ore.stepwise(formula = Employed ~ .^2, data = LONGLEY, add.p = 0.1,
    drop.p = 0.1)

Coefficients:
            (Intercept)                    Year        GNP.deflator:GNP  GNP.deflator:Unemployed  GNP.de
flator:Population
             -3.539e-01               3.589e-05              -2.978e-03               2.326e-04
        2.303e-05
       GNP:Armed.Forces                GNP:Year
              6.875e-06               2.007e-04
```

Copyright © 2020 Oracle and/or its affiliates.

step with ore.lm – results

Akaike information criterion (**AIC**)
- Measure of quality of a model
- Used for model selection

```
R> oreStep2 <-
+   step(ore.lm(Employed ~ 1, data = LONGLEY),
+        scope = terms(Employed ~ .^2, data = LONGLEY))
Start:  AIC=41.17
Employed ~ 1

               Df Sum of Sq     RSS     AIC
+ GNP           1   178.973   6.036 -11.597
+ Year          1   174.552  10.457  -2.806
+ GNP.deflator  1   174.397  10.611  -2.571
+ Population    1   170.643  14.366   2.276
+ Unemployed    1    46.716 138.293  38.509
+ Armed.Forces  1    38.691 146.318  39.411
<none>                      185.009  41.165

Step:  AIC=-11.6
Employed ~ GNP

               Df Sum of Sq     RSS     AIC
+ Unemployed    1     2.457   3.579 -17.960
+ Population    1     2.162   3.874 -16.691
+ Year          1     1.125   4.911 -12.898
<none>                        6.036 -11.597
+ GNP.deflator  1     0.212   5.824 -10.169
+ Armed.Forces  1     0.077   5.959  -9.802
- GNP           1   178.973 185.009  41.165

Step:  AIC=-17.96
Employed ~ GNP + Unemployed

                 Df Sum of Sq     RSS     AIC
+ Armed.Forces    1     0.822   2.757 -20.137
<none>                          3.579 -17.960
+ Year            1     0.340   3.239 -17.556
+ GNP:Unemployed  1     0.182   3.397 -16.795
+ Population      1     0.097   3.482 -16.399
+ GNP.deflator    1     0.019   3.560 -16.044
- Unemployed      1     2.457   6.036 -11.597
- GNP             1   134.714 138.293  38.509
```

```
Step:  AIC=-20.14
Employed ~ GNP + Unemployed + Armed.Forces

                          Df Sum of Sq    RSS     AIC
+ Year                     1     1.898  0.859 -36.799
+ GNP:Unemployed           1     0.614  2.143 -22.168
+ Population               1     0.390  2.367 -20.578
<none>                                  2.757 -20.137
+ Unemployed:Armed.Forces  1     0.083  2.673 -18.629
+ GNP.deflator             1     0.073  2.684 -18.566
+ GNP:Armed.Forces         1     0.060  2.697 -18.489
- Armed.Forces             1     0.822  3.579 -17.960
- Unemployed               1     3.203  5.959  -9.802
- GNP                      1    78.494 81.250  31.999

Step:  AIC=-36.8
Employed ~ GNP + Unemployed + Armed.Forces + Year

                          Df Sum of Sq    RSS     AIC
<none>                                 0.8587 -36.799
+ Unemployed:Year          1    0.0749 0.7838 -36.259
+ GNP:Unemployed           1    0.0678 0.7909 -36.115
+ Unemployed:Armed.Forces  1    0.0515 0.8072 -35.788
+ GNP:Armed.Forces         1    0.0367 0.8220 -35.498
+ Population               1    0.0193 0.8393 -35.163
+ GNP.deflator             1    0.0175 0.8412 -35.129
+ Armed.Forces:Year        1    0.0136 0.8451 -35.054
+ GNP:Year                 1    0.0084 0.8502 -34.957
- GNP                      1    0.4647 1.3234 -31.879
- Year                     1    1.8980 2.7567 -20.137
- Armed.Forces             1    2.3806 3.2393 -17.556
- Unemployed               1    4.0491 4.9077 -10.908
R> oreStep2

Call:
ore.lm(formula = Employed ~ GNP + Unemployed + Armed.Forces +
    Year, data = LONGLEY)

Coefficients:
 (Intercept)          GNP    Unemployed  Armed.Forces          Year
  -3.599e+03   -4.019e-02    -2.088e-02    -1.015e-02     1.887e+00
```

Copyright © 2020 Oracle and/or its affiliates.

How to use Akaike's Information Criterion (AIC) as a selection criterion for stepwise regression

AIC cannot be used with ore.stepwise, since the ore.stepwise function uses marginal t-tests for variable selection

ore.lm integrates with R's step function, which does use AIC

It is not as fast as ore.stepwise, but will get the job done

```
R> LONGLEY <- ore.push(longley)
R> mod <- ore.lm(Employed ~ ., data = LONGLEY)
R> step(mod)
Start:  AIC=-33.22
Employed ~ GNP.deflator + GNP + Unemployed + Armed.Forces +
Population + Year

               Df Sum of Sq     RSS     AIC
- GNP.deflator  1   0.00292 0.83935 -35.163
- Population    1   0.00475 0.84117 -35.129
- GNP           1   0.10631 0.94273 -33.305
<none>                      0.83642 -33.219
- Year          1   1.49881 2.33524 -18.792
- Unemployed    1   1.59014 2.42656 -18.178
- Armed.Forces  1   2.16091 2.99733 -14.798

Step:  AIC=-35.16
Employed ~ GNP + Unemployed + Armed.Forces + Population + Year
```

```
               Df Sum of Sq    RSS     AIC
- Population    1   0.01933 0.8587 -36.799
<none>                      0.8393 -35.163
- GNP           1   0.14637 0.9857 -34.592
- Year          1   1.52725 2.3666 -20.578
- Unemployed    1   2.18989 3.0292 -16.628
- Armed.Forces  1   2.39752 3.2369 -15.568

Step:  AIC=-36.8
Employed ~ GNP + Unemployed + Armed.Forces + Year

               Df Sum of Sq    RSS     AIC
<none>                      0.8587 -36.799
- GNP           1    0.4647 1.3234 -31.879
- Year          1    1.8980 2.7567 -20.137
- Armed.Forces  1    2.3806 3.2393 -17.556
- Unemployed    1    4.0491 4.9077 -10.908

Call:
ore.lm(formula = Employed ~ GNP + Unemployed + Armed.Forces +
    Year, data = LONGLEY)

Coefficients:
 (Intercept)           GNP     Unemployed  Armed.Forces   Year
  -3.599e+03    -4.019e-02    -2.088e-02     -1.015e-2  1.887e+00
```

Copyright © 2020 Oracle and/or its affiliates.

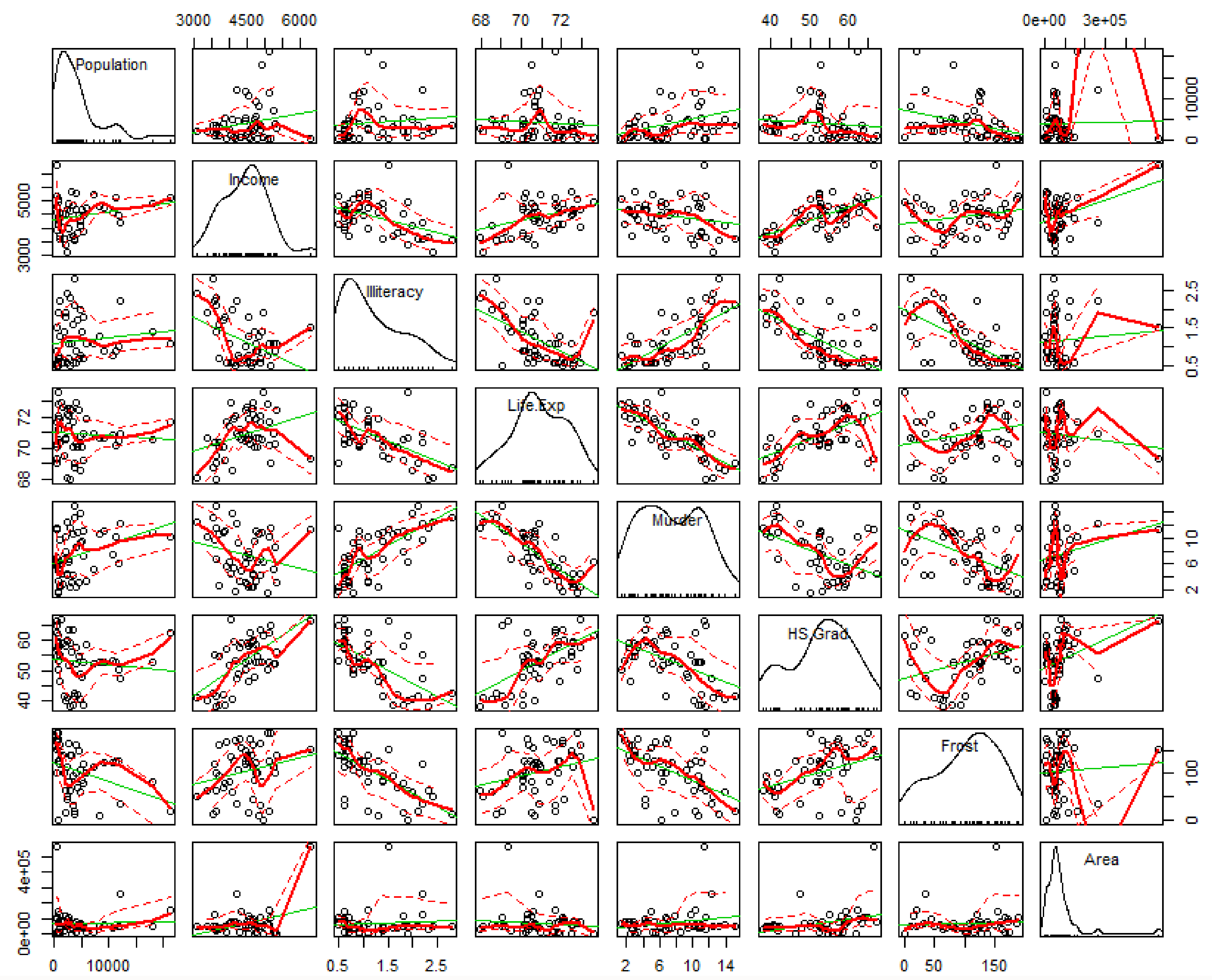
Linear Model Example

Copyright © 2020 Oracle and/or its affiliates.

An example using the R state.x77 data set

```
library(car)
?state.x77
state.df <-
as.data.frame(state.x77)
scatterplotMatrix(state.df)
```

Scatterplot of pairs of variables (bivariate analysis)
Fitted loess curve (red)
Fitted linear model (green)
Diagonals show density and run plot per variable



Copyright © 2020 Oracle and/or its affiliates.

An example using the R state.x77 data set

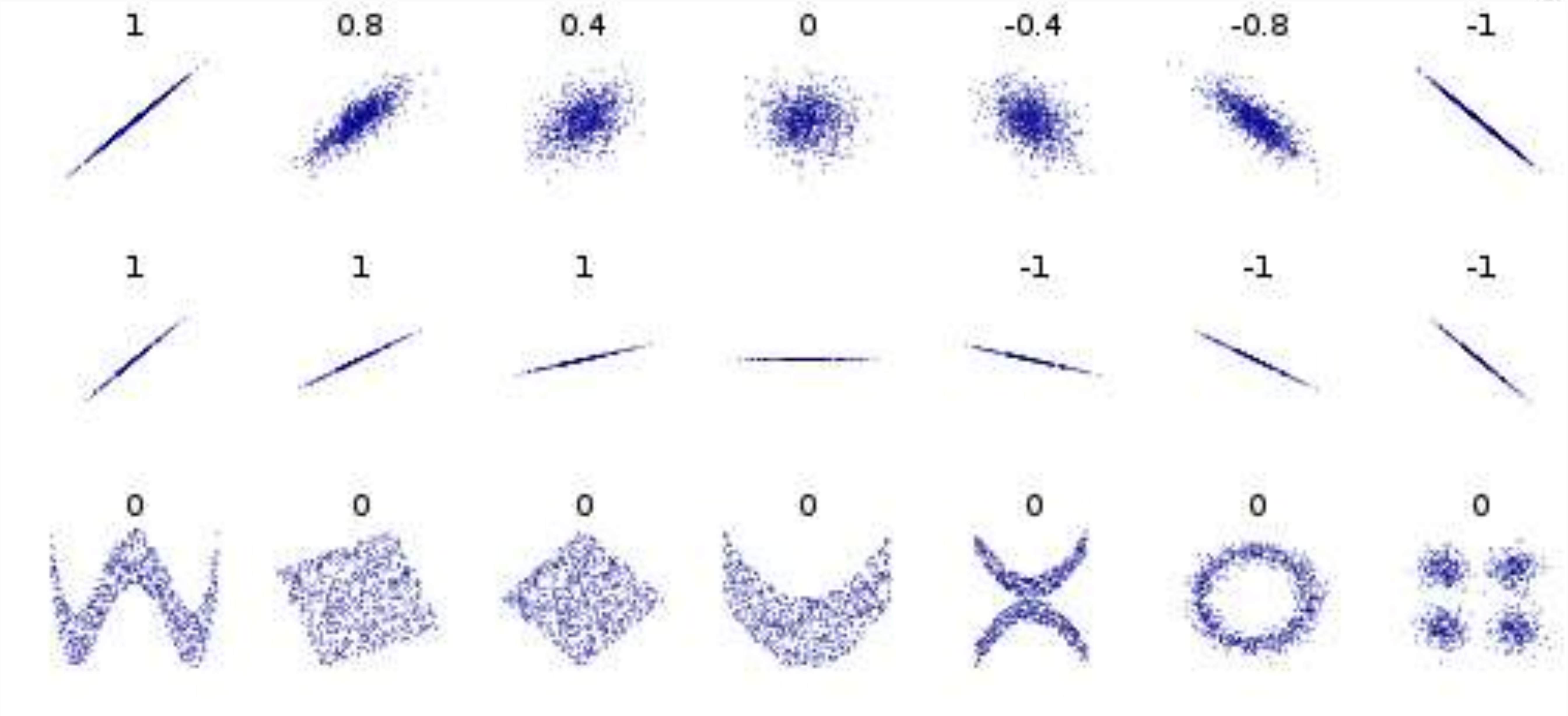
```
options(digits=3)
cor(state.df)
# also with in-database execution
STATE <- ore.push(state.df)
cor(STATE)
```

```
> options(digits=3)
> cor(state.df)
```

	Population	Income	Illiteracy	Life Exp	Murder	HS Grad	Frost	Area
Population	1.0000	0.208	0.1076	-0.0681	0.344	-0.0985	-0.3322	0.0225
Income	0.2082	1.000	-0.4371	0.3403	-0.230	0.6199	0.2263	0.3633
Illiteracy	0.1076	-0.437	1.0000	-0.5885	0.703	-0.6572	-0.6719	0.0773
Life Exp	-0.0681	0.340	-0.5885	1.0000	-0.781	0.5822	0.2621	-0.1073
Murder	0.3436	-0.230	0.7030	-0.7808	1.000	-0.4880	-0.5389	0.2284
HS Grad	-0.0985	0.620	-0.6572	0.5822	-0.488	1.0000	0.3668	0.3335
Frost	-0.3322	0.226	-0.6719	0.2621	-0.539	0.3668	1.0000	0.0592
Area	0.0225	0.363	0.0773	-0.1073	0.228	0.3335	0.0592	1.0000

Copyright © 2020 Oracle and/or its affiliates.

Interpreting correlation



http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

Copyright © 2020 Oracle and/or its affiliates.

An example using the R state.x77 data set

```
fit <- lm(Murder ~ .,
state.df)

summary(fit)
```

Life Exp is significant, but we would expect this

Remove it from the model, then HS Grad and Income

```
> fit <- lm(Murder ~ ., state.df)
> summary(fit)

Call:
lm(formula = Murder ~ ., data = state.df)

Residuals:
   Min     1Q Median     3Q    Max 
 -3.44  -1.10  -0.06   1.18   3.24 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.22e+02   1.79e+01    6.83  2.5e-08 ***
Population   1.88e-04   6.47e-05    2.90   0.0058 ** 
Income      -1.59e-04   5.73e-04   -0.28   0.7823    
Illiteracy   1.37e+00   8.32e-01    1.65   0.1064    
`Life Exp`  -1.65e+00   2.56e-01   -6.46  8.7e-08 ***
`HS Grad`    3.23e-02   5.73e-02    0.56   0.5752    
Frost       -1.29e-02   7.39e-03   -1.74   0.0887 .  
Area         5.97e-06   3.80e-06    1.57   0.1239    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.75 on 42 degrees of freedom
Multiple R-squared: 0.808, Adjusted R-squared: 0.776 
F-statistic: 25.3 on 7 and 42 DF,  p-value: 3.87e-13
```

Copyright © 2020 Oracle and/or its affiliates.

An example using the R state.x77 data set

```
# Rename vars to remove space
names(state.df)[4] <- "LifeExp"
names(state.df)[6] <- "HSGrad"

fit <- lm(Murder ~ .-LifeExp,
state.df)
summary(fit)

fit <- lm(Murder ~ .-LifeExp -
Income - HSGrad, state.df)
summary(fit)
```

```
> fit <- lm(Murder ~ .-LifeExp-Income-HSGrad, state.df)
> summary(fit)

Call:
lm(formula = Murder ~ . - LifeExp - Income - HSGrad, data = state.df)

Residuals:
   Min     1Q Median     3Q    Max 
-4.187 -1.928 -0.191  1.588  7.403 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.65e+00   1.94e+00    0.85    0.398    
Population   2.15e-04   8.44e-05    2.54    0.014 *  
Illiteracy   3.87e+00   7.97e-01    4.86  1.5e-05 ***
Frost       -2.40e-03   9.84e-03   -0.24    0.809    
Area         7.58e-06   4.17e-06    1.82    0.076 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.45 on 45 degrees of freedom
Multiple R-squared: 0.597, Adjusted R-squared: 0.561
F-statistic: 16.6 on 4 and 45 DF,  p-value: 1.95e-08
```

Copyright © 2020 Oracle and/or its affiliates.

An example using the R state.x77 data set

```
fit2 <- lm(Murder ~ .^2,
state.df)
summary(fit2)
```

```
> fit <- lm(Murder ~ .^2, state.df)
> summary(fit)

Call:
lm(formula = Murder ~ .^2, data = state.df)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.29472 -0.59402  0.01303  0.42184  2.69677 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)  
(Intercept)           5.740e+01  4.143e+02   0.139   0.8911  
Population            1.279e-02  1.477e-02   0.866   0.3963  
Income                1.018e-01  5.613e-02   1.814   0.0840 .
Illiteracy           -5.482e+01  9.628e+01  -0.569   0.5751  
LifeExp              -2.466e-01  6.498e+00  -0.038   0.9701  
HSGrad               -6.762e+00  6.317e+00  -1.070   0.2966  
Frost                -7.496e-01  8.487e-01  -0.883   0.3871  
Area                  4.334e-04  7.694e-04   0.563   0.5792  
Population:Income     2.248e-07  3.312e-07   0.679   0.5047  
Population:Illiteracy -6.580e-04 4.989e-04  -1.319   0.2014  
Population:LifeExp   -1.429e-04  2.196e-04  -0.651   0.5223  
Population:HSGrad    -4.850e-05  5.473e-05  -0.886   0.3856  
Population:Frost     -4.495e-06  3.727e-06  -1.206   0.2412  
Population:Area       2.985e-09  3.357e-09   0.889   0.3840  
Income:Illiteracy    -2.587e-03  4.106e-03  -0.630   0.5355  
Income:LifeExp       -1.500e-03  7.634e-04  -1.965   0.0628 .
Income:HSGrad         9.512e-05  1.983e-04   0.480   0.6364  
Income:Frost          1.165e-05  3.225e-05   0.361   0.7215  
Income:Area           1.476e-08  2.130e-08   0.693   0.4958  
Illiteracy:LifeExp    7.934e-01  1.560e+00   0.509   0.6163  
Illiteracy:HSGrad     2.540e-01  2.098e-01   1.211   0.2395  
Illiteracy:Frost      1.915e-02  1.706e-02   1.122   0.2743  
Illiteracy:Area      -3.624e-05  3.121e-05  -1.161   0.2586  
LifeExp:HSGrad        8.504e-02  9.378e-02   0.907   0.3748  
LifeExp:Frost         6.938e-03  1.299e-02   0.534   0.5989  
LifeExp:Area         -4.073e-06  1.043e-05  -0.391   0.7001  
HSGrad:Frost          3.345e-03  2.252e-03   1.485   0.1524  
HSGrad:Area          -2.973e-06  2.212e-06  -1.344   0.1931  
Frost:Area           -3.404e-08  2.920e-07  -0.117   0.9083  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.431 on 21 degrees of freedom
Multiple R-squared: 0.9356, Adjusted R-squared: 0.8497 
F-statistic: 10.89 on 28 and 21 DF,  p-value: 2.42e-07
```

Copyright © 2020 Oracle and/or its affiliates.

An example using the R state.x77 data set

```
STATE <- ore.push(state.df)
fit3 <- ore.stepwise(Murder ~
.^2, STATE)
summary(fit3)
```

```
> state.df <- as.data.frame(state.x77)
> scatterplotMatrix(state.df)
> names(state.df)[4] <- "LifeExp"
> names(state.df)[6] <- "HSGrad"
> STATE.DF <- ore.push(state.df)
> fit <- ore.stepwise(Murder ~ .^2, STATE.DF)
> summary(fit)

Call:
ore.stepwise(formula = Murder ~ .^2, data = STATE.DF)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5399 -1.1127 -0.0823  1.2496  3.5140

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              1.203e+02  1.631e+01   7.380 3.19e-09 ***
LifeExp                 -1.621e+00  2.236e-01  -7.248 4.95e-09 ***
Population:Illiteracy    1.669e-04  4.479e-05   3.726 0.000551 ***
Illiteracy:LifeExp       2.590e-02  7.559e-03   3.426 0.001337 **
Illiteracy:Frost        -1.500e-02  5.625e-03  -2.666 0.010701 *
Frost:Area               5.508e-08  2.138e-08   2.576 0.013433 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.682 on 44 degrees of freedom
Multiple R-squared: 0.8137, Adjusted R-squared: 0.7925
F-statistic: 38.43 on 5 and 44 DF,  p-value: 5.541e-15
```

Copyright © 2020 Oracle and/or its affiliates.

Formula specification option

Class `formula` accepts the following options

Symbol	Description	Example
~	separates response/target variables from explanatory/predictor variables	y ~ x
+	separates predictors	y ~ a + b + c
:	specify interaction terms between predictors	y ~ a + c + a:c
*	Specify all possible interactions between specific predictors	y ~ a * b * c
^	Specify interactions up to a specific degree	y ~ (a + b + c)^2 ➔ y ~ a + b + c + a:b + a:c + b:c
.	Represents all other variables beside target variable	y ~ .
-	Remove the specified predictor(s)	y ~ (a + b + c)^3 – a:b – b:c ➔ y ~ a + b + c + a:c + a:b:c
-1	Suppresses the intercept from the model, forcing the regression line through the origin at a = 0	y ~ a + b - 1
I()	Interpret contents arithmetically	y ~ a + I(b-c)^3) ➔ y ~ a + v, where v = (b-c)^3
function	Mathematical function	sqrt(y) ~ a + log(b)

Copyright © 2020 Oracle and/or its affiliates.

Fitting Linear Models

Copyright © 2020 Oracle and/or its affiliates.

Ordinary Least Squares Regression Assumptions

Normality –
for fixed values of predictor variables, target variable is normally distributed

Independence –
target values are independent of each other – one does not influence others

Linearity –
target is linearly related to the predictor variables

Homoscedasticity –
target variance doesn't change with different ranges of predictor variables

Violating assumptions may mean statistical significance tests and confidence intervals may be inaccurate

Copyright © 2020 Oracle and/or its affiliates.

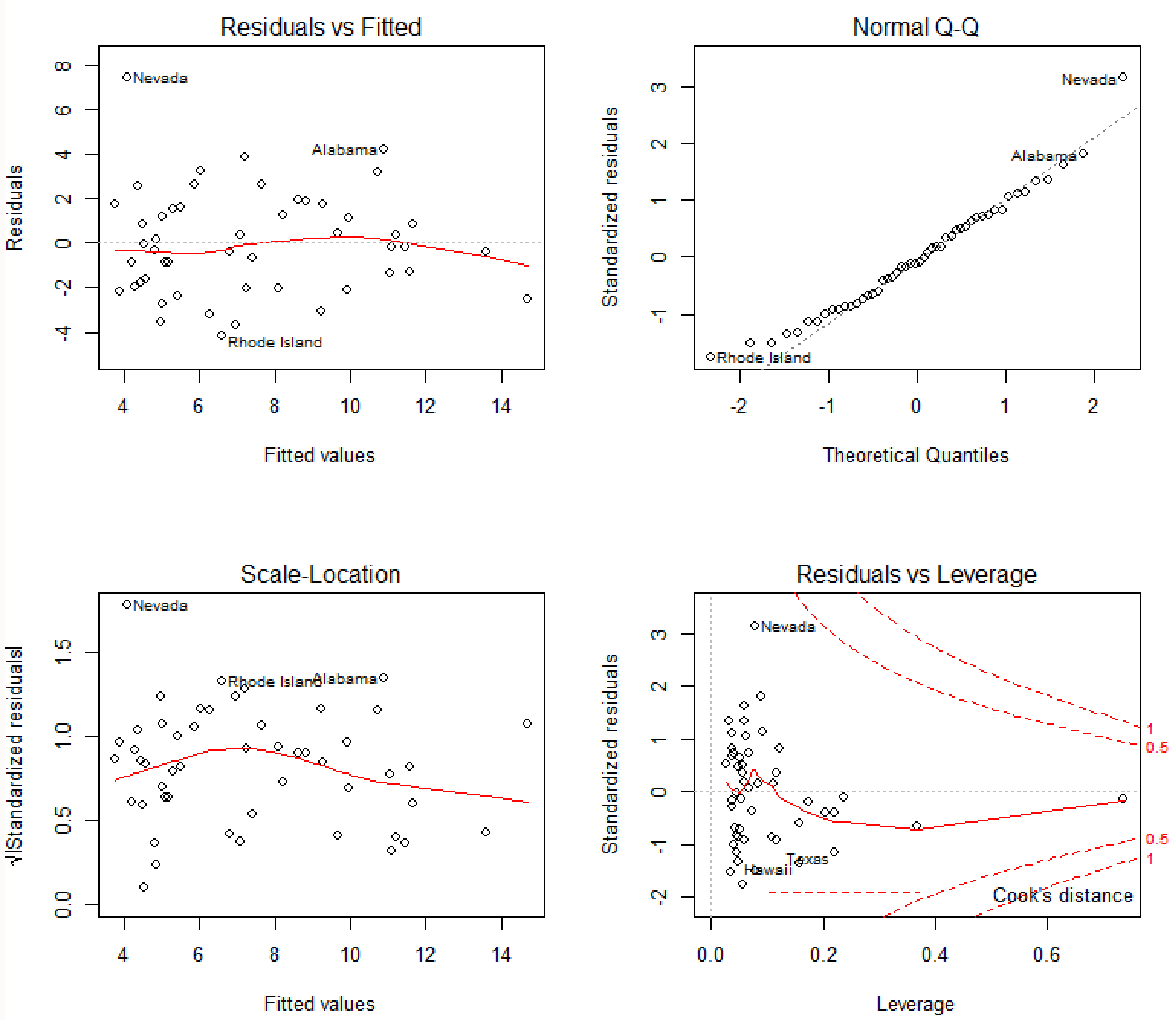
Assessing the quality of a linear model

```
confint(fit)
par(mfrow=c(2,2))
plot(fit)
```

Interpreting results:

The interval 2.26 to 5.47 is 95% likely to contain the true Murder rate change given a 1% change in illiteracy

```
> confint(fit)
                    2.5 %    97.5 %
(Intercept) -2.25e+00 5.554052
Population   4.48e-05 0.000385
Illiteracy   2.26e+00 5.473176
Frost       -2.22e-02 0.017426
Area        -8.15e-07 0.000016
```

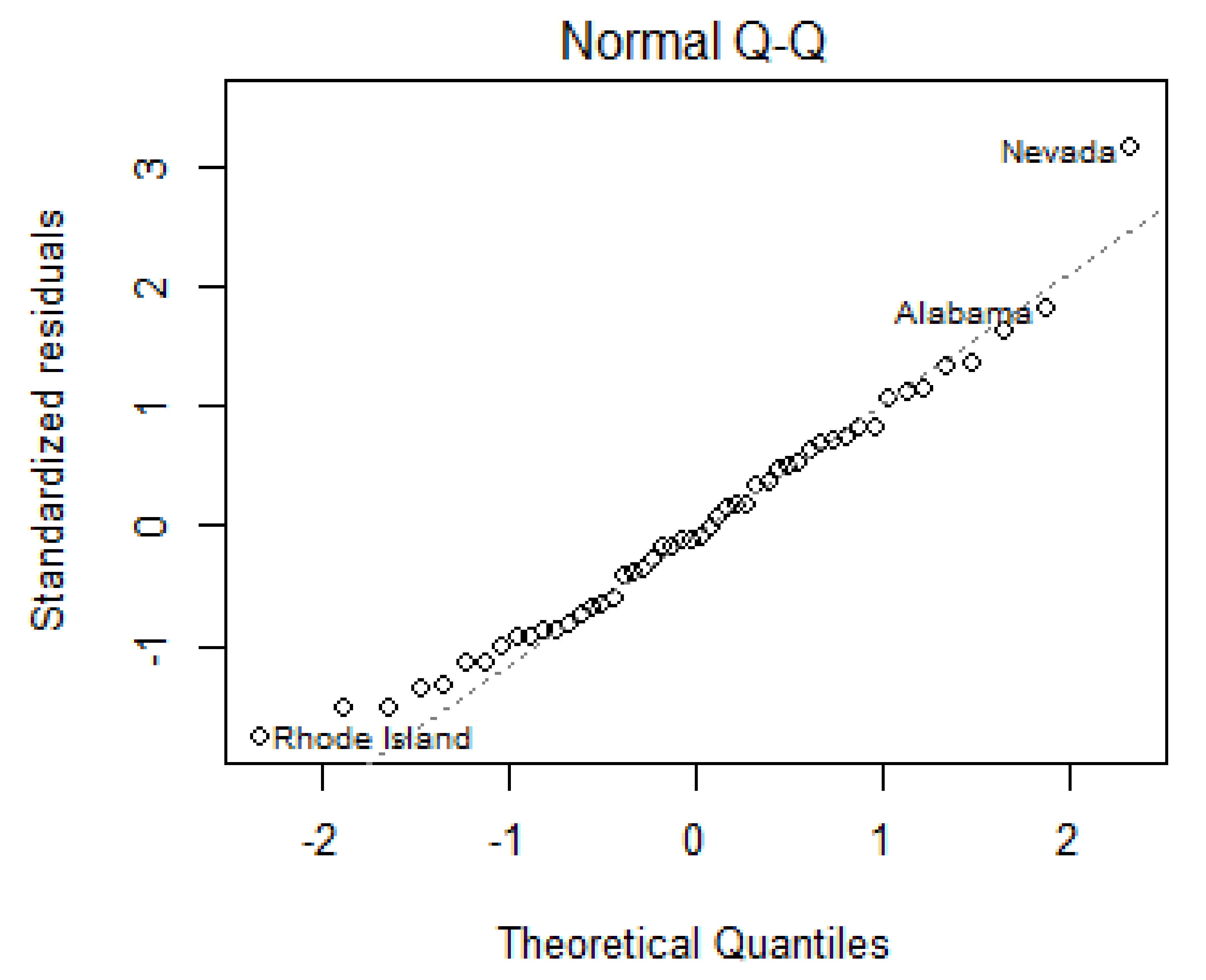


Copyright © 2020 Oracle and/or its affiliates.

Assessing Normality

Normal Q-Q plot should be a straight line if the data meets the normality assumption

Data do not fall on this line, so normality assumption violated



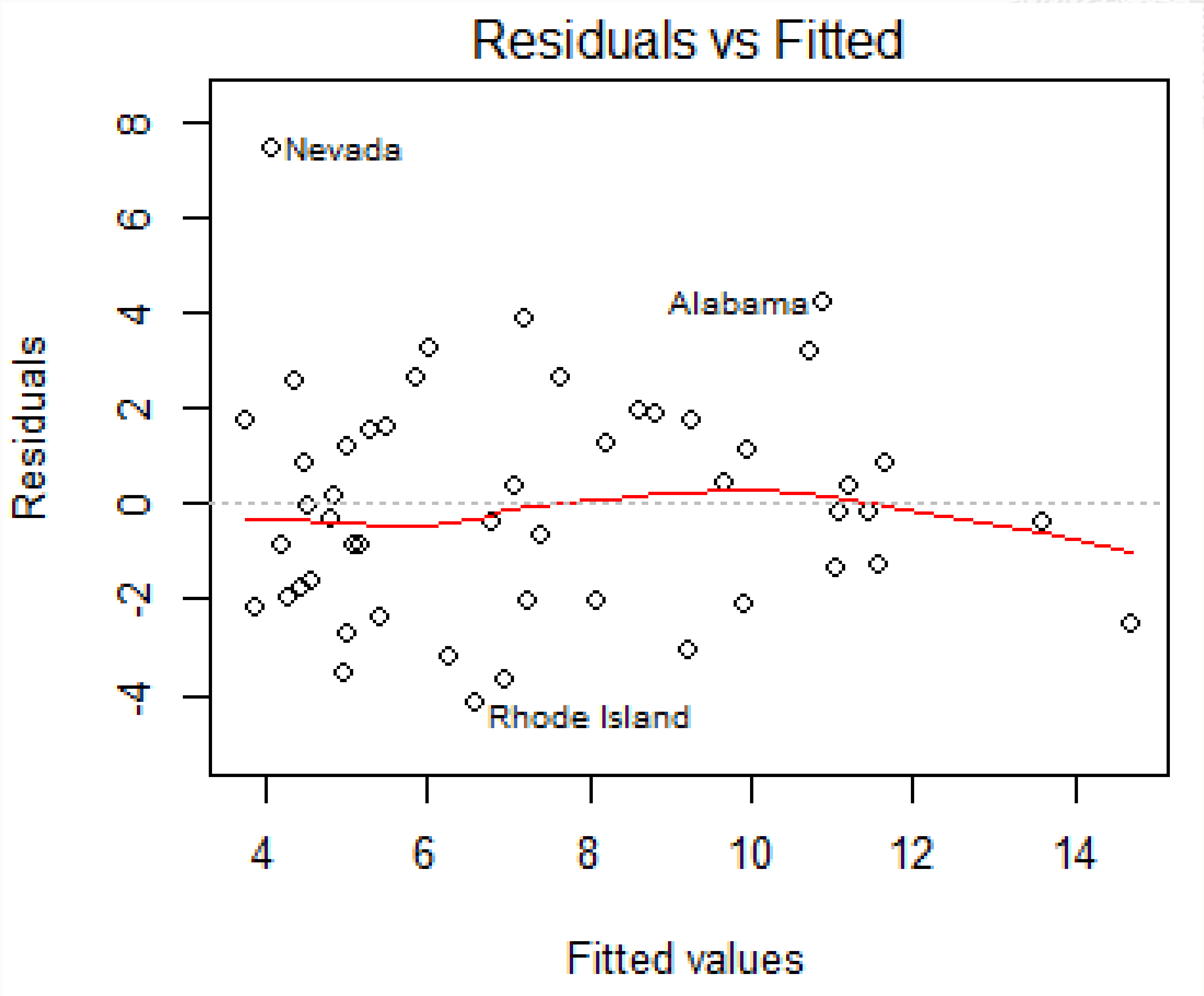
Copyright © 2020 Oracle and/or its affiliates.

Assessing Linearity

Residuals should be randomly distributed if target is linearly related to predictors

Only “random noise” should remain

There appears to be a curved relationship, indicating the need for quadratic terms

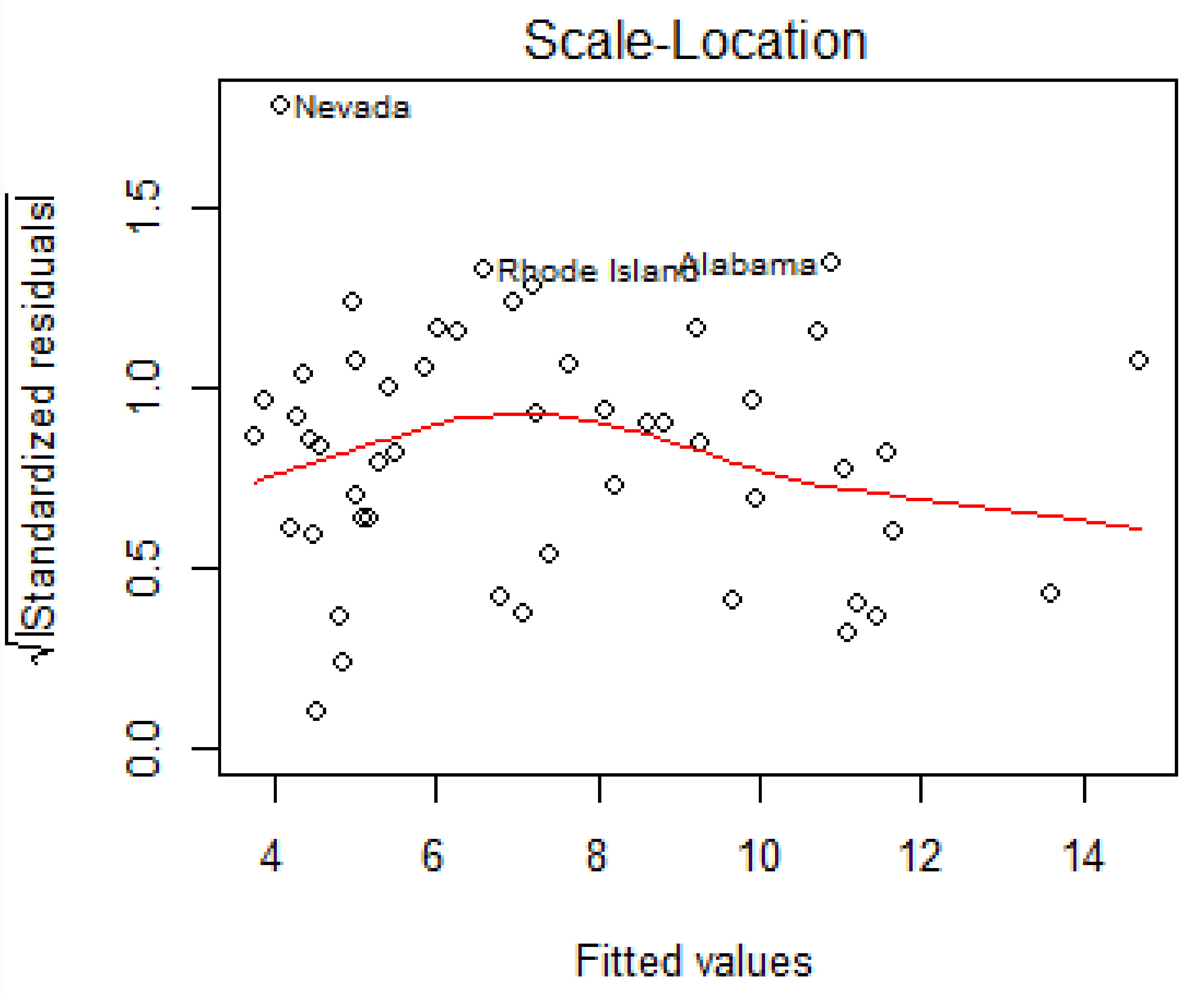


Copyright © 2020 Oracle and/or its affiliates.

Assessing Homoscedasticity

Sqrt of standardized residuals should be randomly distributed about a horizontal line

There appears to be a curved relationship, indicating the need for quadratic terms



Copyright © 2020 Oracle and/or its affiliates.

Should some data be removed from the data set?

Outliers

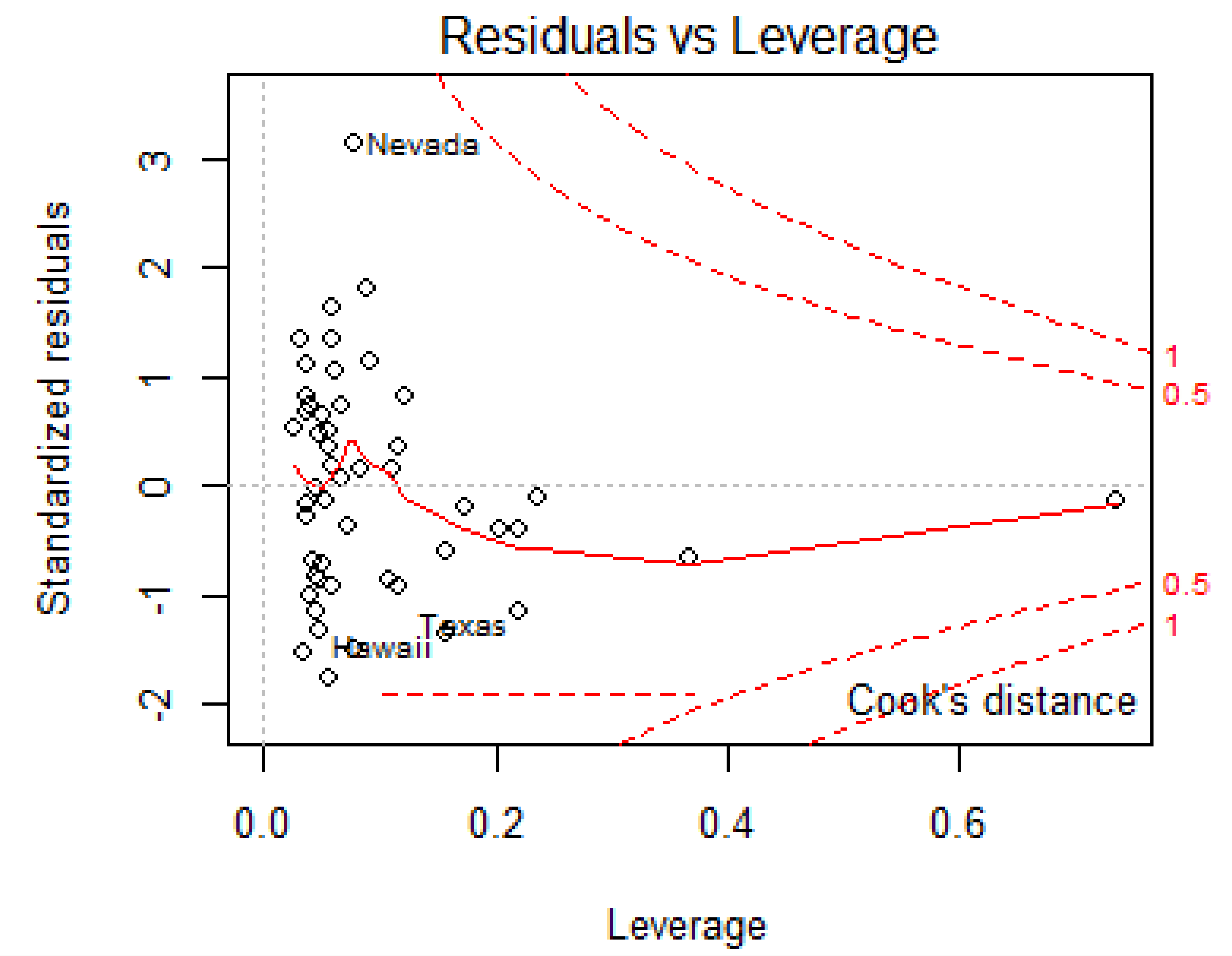
- Have large residual value
- Model doesn't predict well

High Leverage Values

- Predictor values are unusual relative to other observations

Influential Observations

- row has unusually high impact on model parameters
- Indicated by Cook's distance



Copyright © 2020 Oracle and/or its affiliates.

Package **car** and **qqPlot** and **outlierTest**

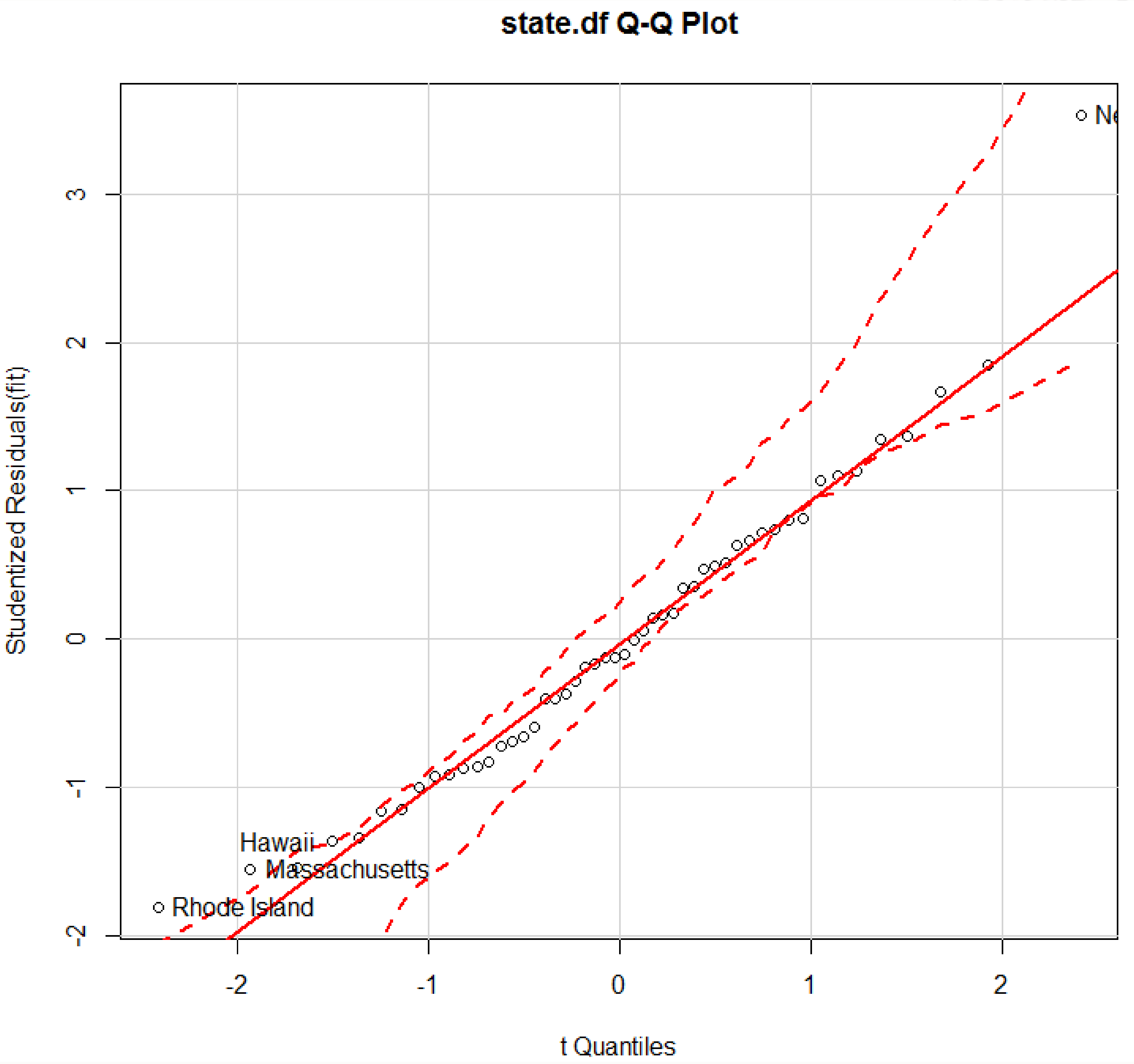
More accurate assessment of normality assumption

95% confidence bound drawn around 45º line

Outliers can be manually selected

Iteratively remove outliers and reinvoke `outlierTest`

```
library(car)
qqPlot(fit,
  labels=row.names(state.df),
  id.method="identify",
  simulate=TRUE,
  main="state.df Q-Q Plot")
outlierTest(fit)
# Select points and hit ESC
```



Copyright © 2020 Oracle and/or its affiliates.

Other options...

See package `gvlma` and function `gvlma` for overall assessment of model assumptions

Copyright © 2020 Oracle and/or its affiliates.

More on Multicollinearity

One variable is closely correlated (or determined by) another

- E.g., age & data of birth
  year & population

Use *variance inflation factor* (VIF)

- Use `car` package function `vif`
- Use sqrt(vif) > 2, then there is a multicollinearity problem

Problematic for interpreting individual predictor variables, but not for prediction

Copyright © 2020 Oracle and/or its affiliates.

Package `car` and `vif`

```
library(car)
vif(fit)
sqrt(vif(fit)) > 2

fit2 <- lm(Murder~., state.df)
vif(fit2)
sqrt(vif(fit2)) > 2
```

```
> vif(fit)
Population Illiteracy      Frost       Area
      1.16       1.93       2.14       1.03
> sqrt(vif(fit)) > 2
Population Illiteracy      Frost       Area
     FALSE      FALSE      FALSE      FALSE
```

```
> fit2 <- lm(Murder~., state.df)
> vif(fit2)
Population     Income Illiteracy    LifeExp     HSGrad
      1.34       1.99       4.14       1.90       3.44
     Frost       Area
      2.37       1.69
> sqrt(vif(fit2)) > 2
Population     Income Illiteracy    LifeExp     HSGrad
     FALSE      FALSE       TRUE      FALSE      FALSE
     Frost       Area
     FALSE      FALSE
```

Copyright © 2020 Oracle and/or its affiliates.

Remedies for common problems

Multicollinearity

- Deleting one of the predictors involved or where sqrt(vif) > 2
- Use ridge regression

Transform target and/or predictor variables

- Y^{λ}, where, for example, λ in {-2, -1, -0.5, 0==log, .5, none, 2}
- log(Y)
- Remember – a transformation should "make sense" for interpretation of result

Normality assumption

- Use a non-parametric algorithm; Use GLM
- In car package, use `powerTransform` on target to get estimate of power λ

Linearity assumption

- Use a non-linear regression algorithm
- In car package, use boxTidwell on predictors to get estimate of power λ

Homoscedasticity assumption – homogeneity of variance

- In car package, use spreadLevelPlot on model to get estimate of power λ on for Y^{λ}

Copyright © 2020 Oracle and/or its affiliates.

Parametric vs. non-parametric algorithms

Parametric

Fixed family of functions

Fixed number of parameters that are independent of the number of observations

E.g., linear regression

Non-parametric

Learned function based on observation data

E.g. KNN classifier, SVM

http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/NonParametric

Copyright © 2020 Oracle and/or its affiliates.

Is one model better than another?

```
anova(fit, fit2)

AIC(fit, fit2)
```

```
> anova(fit, fit2)
Analysis of Variance Table

Model 1: Murder ~ Population + Illiteracy + Area + Frost
Model 2: Murder ~ Population + Income + Illiteracy + LifeExp + HSGrad +
    Frost + Area
  Res.Df RSS Df Sum of Sq    F  Pr(>F)
1     45 269
2     42 128  3       141 15.5 6.4e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> AIC(fit, fit2)
     df AIC
fit   6 238
fit2  9 207
```

ANOVA requires nested models
Look for significant p-value

AIC does not require nested models
Models with smaller AIC values are better

http://en.wikipedia.org/wiki/Akaike_information_criterion

Copyright © 2020 Oracle and/or its affiliates.

RandomForest

Copyright © 2020 Oracle and/or its affiliates.

Random Forest Algorithm

Ensemble learning technique for classification and regression
Known for high accuracy models
Constructs many “small” decision trees
For classification, predicts mode of classes predicted by individual trees
For regression, predicts mean prediction of individual trees
Avoids overfitting, which is common for decision trees

Developed by Leo Breiman and Adele Cutler combining the ideas of “bagging” and random selection of variables resulting in a collection of decision trees with controlled variance

Copyright © 2020 Oracle and/or its affiliates.

ore.randomForest supports classification

Enables performance and scalability for larger data sets

Executes in parallel for model building and scoring

- ore.parallel global option used for preferred DOP

Oracle R Distribution new randomForest function

- Reduces memory requirements over standard R (~7X)
- As a result, reduces memory requirements for ore.randomForest
- ORD randomForest supports *classification* only

Can use Oracle R Distribution's or R's randomForest package

Copyright © 2020 Oracle and/or its affiliates.

Random Forest

```
ore.randomForest(formula, data, ntree=500, mtry = NULL,
              replace = TRUE, classwt = NULL, cutoff = NULL,
              sampsize = if(replace) nrow(data) else ceiling(0.632*nrow(data)),
              nodesize = 1L, maxnodes = NULL, confusion.matrix = FALSE,
              na.action = na.fail, ...)

grabTree(object, k = 1L, labelVar = FALSE, ...)
predict(object, newdata,
        type = c("response", "prob", "vote", "all"),
        norm.votes = TRUE,
        supplemental.cols = NULL,
        cache.model = TRUE, ...)
```

Copyright © 2020 Oracle and/or its affiliates.

Random Forest

ntree – total number of trees to grow

mtry – number of variables randomly sampled as candidates at each tree node split

replace – a logical value indicating whether to execute sampling with replacement

classwt – a vector of priors of the classes. If specified, the length of the vector should be equal to the number of classes in the target column. The vector does not need to add up to 1.

cutoff – a vector of cutoff values. If specified, the length of the vector should be equal to the number of classes in the target column. When determining the prediction class for an observation, the one with the maximum ratio of proportion of votes to cutoff is selected. If not specified, the default is '1/k' where 'k' is the number of classes.

sampsize – size of the sample to draw for growing trees

nodesize – minimum size of terminal nodes

maxnodes – maximum number of terminal nodes of each tree to be grown. If not specified, trees can be grown to the maximum size subject to the limits of 'nodesize'.

confusion.matrix – a logical value indicating whether to calculate the confusion matrix. Note that this confusion matrix is not based on OOB (out-of-bag), it is the result of applying the built random forest model to the entire training data.

Copyright © 2020 Oracle and/or its affiliates.

Random Forest

na.action – the manner in which 'NA' values are handled. With the default 'na.fail', it fails if the training data contains 'NA'.

k – an integer indicating which tree's information to extract

labelVar – a logical value indicating whether the 'split var' and 'prediction' columns in the returned frame use meaningful labels.

newdata – an 'ore.frame' object, the test data

type – specifies the type of the output: 'response', 'prob', 'votes', or 'all' returning predicted values, matrix of class probabilities, matrix of vote counts, or both the vote matrix and predicted values, respectively.

norm.votes – a logical value indicating whether the vote counts in the output vote matrix should be normalized. The argument is ignored if 'type' is 'response' or 'prob'.

supplemental.cols – additional columns to include in the prediction result from the 'newdata' data set

cache.model – a logical value indicating whether the entire random forest model is cached in memory during prediction

Copyright © 2020 Oracle and/or its affiliates.

ore.randomForest

ore.randomForest() builds a random forest model by growing trees in parallel

Scoring method 'predict' runs in parallel

```
options(ore.parallel=4)

IRIS <- ore.push(iris)
mod <- ore.randomForest(Species~., IRIS)

tree10 <- grabTree(mod, k = 10, labelVar = TRUE)

ans <- predict(mod,IRIS,type="all",supplemental.cols="Species", cache.model=FALSE)
table(ans$Species, ans$prediction)
```

Copyright © 2020 Oracle and/or its affiliates.

ore.randomForest Results

```
R> options(ore.parallel=4)
R> IRIS <- ore.push(iris)
R> mod <- ore.randomForest(Species~., IRIS)
R> tree10 <- grabTree(mod, k = 10, labelVar = TRUE)
R> ans <- predict(mod, IRIS, type="all", supplemental.cols="Species")
R> table(ans$Species, ans$prediction)  # learns perfectly

             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         50         0
  virginica       0          0        50
R>
R> tree10
   node.id left.daughter right.daughter    split.var split.point status prediction
1        1             2              3 Sepal.Length        5.55      1       <NA>
2        2             4              5  Petal.Width        0.70      1       <NA>
3        3             6              7 Petal.Length        4.75      1       <NA>
4        4             0              0         <NA>        0.00     -1     setosa
5        5             8              9 Sepal.Length        4.95      1       <NA>
6        6            10             11  Sepal.Width        3.70      1       <NA>
7        7            12             13 Petal.Length        5.05      1       <NA>
8        8             0              0         <NA>        0.00     -1  virginica
9        9             0              0         <NA>        0.00     -1 versicolor
10      10             0              0         <NA>        0.00     -1 versicolor
11      11             0              0         <NA>        0.00     -1     setosa
12      12            14             15  Petal.Width        1.75      1       <NA>
13      13             0              0         <NA>        0.00     -1  virginica
14      14            16             17 Sepal.Length        6.15      1       <NA>
15      15             0              0         <NA>        0.00     -1  virginica
16      16             0              0         <NA>        0.00     -1  virginica
17      17             0              0         <NA>        0.00     -1 versicolor
```

Copyright © 2020 Oracle and/or its affiliates.

Memory vs. Speed

ore.randomForest for 1.5 is architected for speed

- Relying on OML4R embedded R, parallelism of ore.randomForest achieves many times speedup, but at the cost of memory
- ore.randomForest loads a copy of the training data for each extproc

For example, building 100M rows with DOP=72

- Needs at least 72 x C x datasetSize, where C is a small constant (3-5) required by the algorithm
- Hitting memory limitations with a 10M or 100M dataset with DOP=72 is expected for most machines

ORD's randomForest improves memory usage over R's randomForest (~7X less)

Recommendations

- Reduce ore.parallel for large datasets to complete
- Set memory limit to prevent system memory overrun
- Recommendation for all embedded R-based OML4R algorithms, though particularly critical for current version of ore.randomForest

Copyright © 2020 Oracle and/or its affiliates.

ore.randomForest – how it works

ore.randomForest() builds a random forest model by growing trees in parallel

Returns an 'ore.randomForest' object

Requires Oracle R Distribution (ORD) or 'randomForest' package be installed

- Oracle R Distribution is preferred to the package 'randomForest’ for better performance and compatibility.
- A warning is issued if the package 'randomForest' is used

Scoring method 'predict' runs in parallel

- The default value of cache.model 'TRUE' is recommended when sufficient memory is available
- Otherwise, 'cache.model' should be set to 'FALSE' to prevent memory overuse

ore.parallel global option is used by 'ore.randomForest’ to determine preferred DOP

Copyright © 2020 Oracle and/or its affiliates.

Neural Network

Copyright © 2020 Oracle and/or its affiliates.

Artificial Neural Networks

Neural network (NN) is a mathematical model inspired by biological neural networks and in some sense mimics 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 analysis, including time series prediction, fitness approximation, and modeling
- Classification: including pattern and sequence recognition, novelty detection and sequential decision making
- Data processing: including filtering, clustering, blind source separation and compression
- Robotics: including directing manipulators, computer numerical control

Copyright © 2020 Oracle and/or its affiliates.

Artificial Neural Networks

Well-suited to data with noisy and complex sensor data

Problem characteristics

- Potentially many (numeric) predictors, e.g., pixel values
- Target may be discrete-valued, real-valued, or a vector of such
- Training data may contain errors – robust to noise
- Fast scoring
- Model transparency not required – models difficult to interpret

Universal approximator

- Adding more neurons can lower error to be as small as desired
- Not always the desired behavior

Copyright © 2020 Oracle and/or its affiliates.

Steps to Neural Network modeling

Architecture specification

Data preparation

Building the model

- Stopping criteria: iterations, error on validation set within tolerance

Viewing statistical results from model

Improving the model

Copyright © 2020 Oracle and/or its affiliates.

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

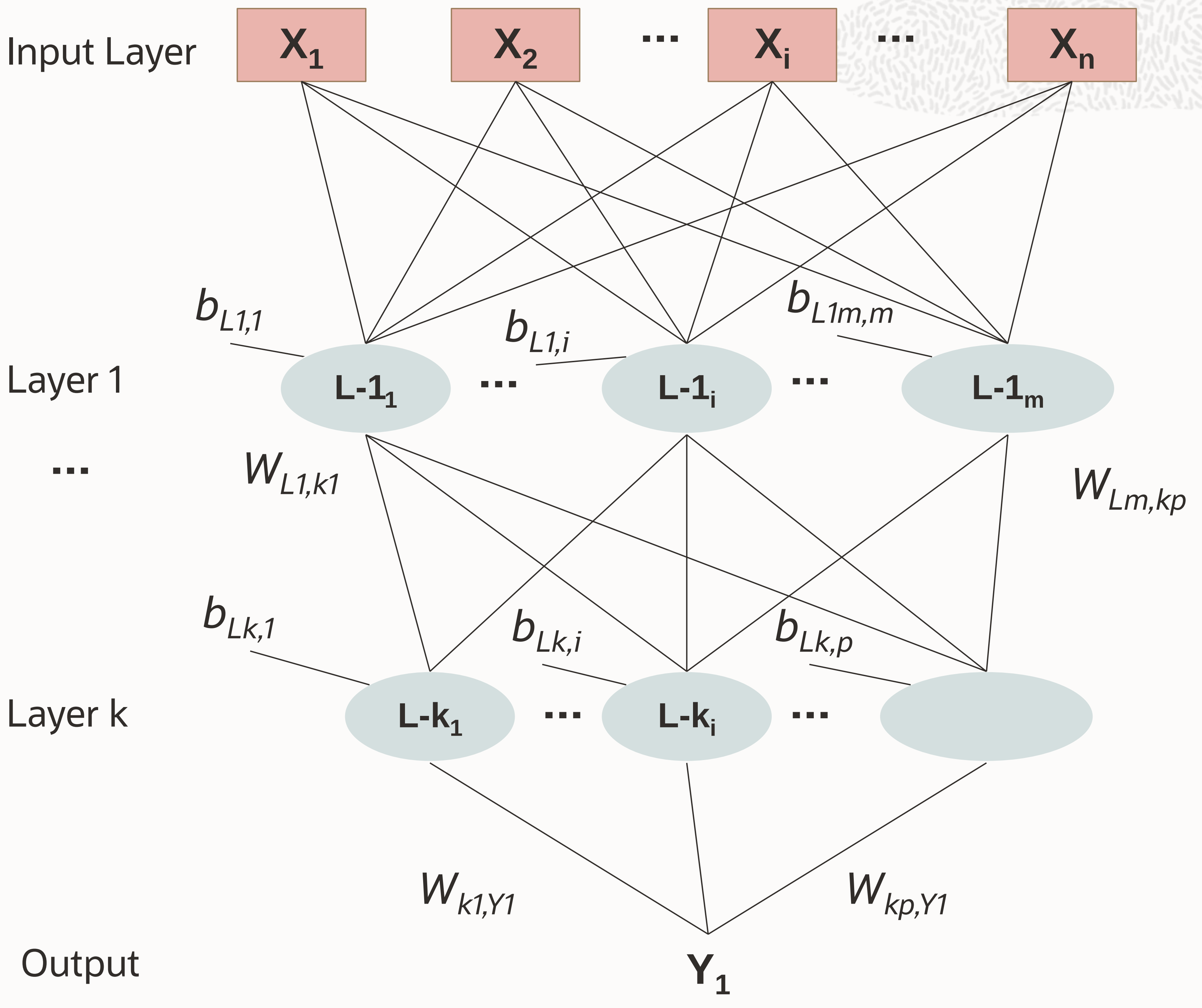
- Currently single numeric target or binary categorical
- Linear activation function default, all others also supported

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



Copyright © 2020 Oracle and/or its affiliates.

Unique aspects of ore.neural

Hidden layer structure complexity

Wide range of 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

Extraordinary scalability on several dimensions including HYPER SPARSE data sets

- Scale-up and Scale-out

Compared to SAS's HPNeural, ore.neural can work with data sets that do not fit in memory

- SAS requires complete data set to fit into distributed memory before it can solve any HP* models

Copyright © 2020 Oracle and/or its affiliates.

Architecture Guidelines

Start

- one hidden layer with one neuron/node
- number of nodes less than sqrt(#observations x #variables)

Test different number of hidden nodes and number of layers

Test different activation functions

Restart (rebuild) model multiple times with different weight initializations to escape local minima, keep model with lowest objective function value, e.g., fit$objValue

Perfecting neural networks is an *art*

Copyright © 2020 Oracle and/or its affiliates.

Data Preparation

Data preparation may be unnecessary if appropriate activation functions are used - especially for targets (outputs)

- Bipolar sigmoid can model values from -1 ... 1 range
- Hyperbolic tangent can model values from -1 ... 1 range
- Logistic sigmoid can model values from 0 ... 1 range

Output preparation

- If target (output) is not scaled (normalized) into ranges above, then linear activation function is appropriate
- If output activation function is non-linear, targets must be scaled

Scaling is recommended for faster convergence, however experimentation is key

For predictors (input data), choose standard R facilities, for instance

- data <- iris
  data$Petal.Length <- scale(data$Petal.Length)
- To normalize Petal.Length around 0

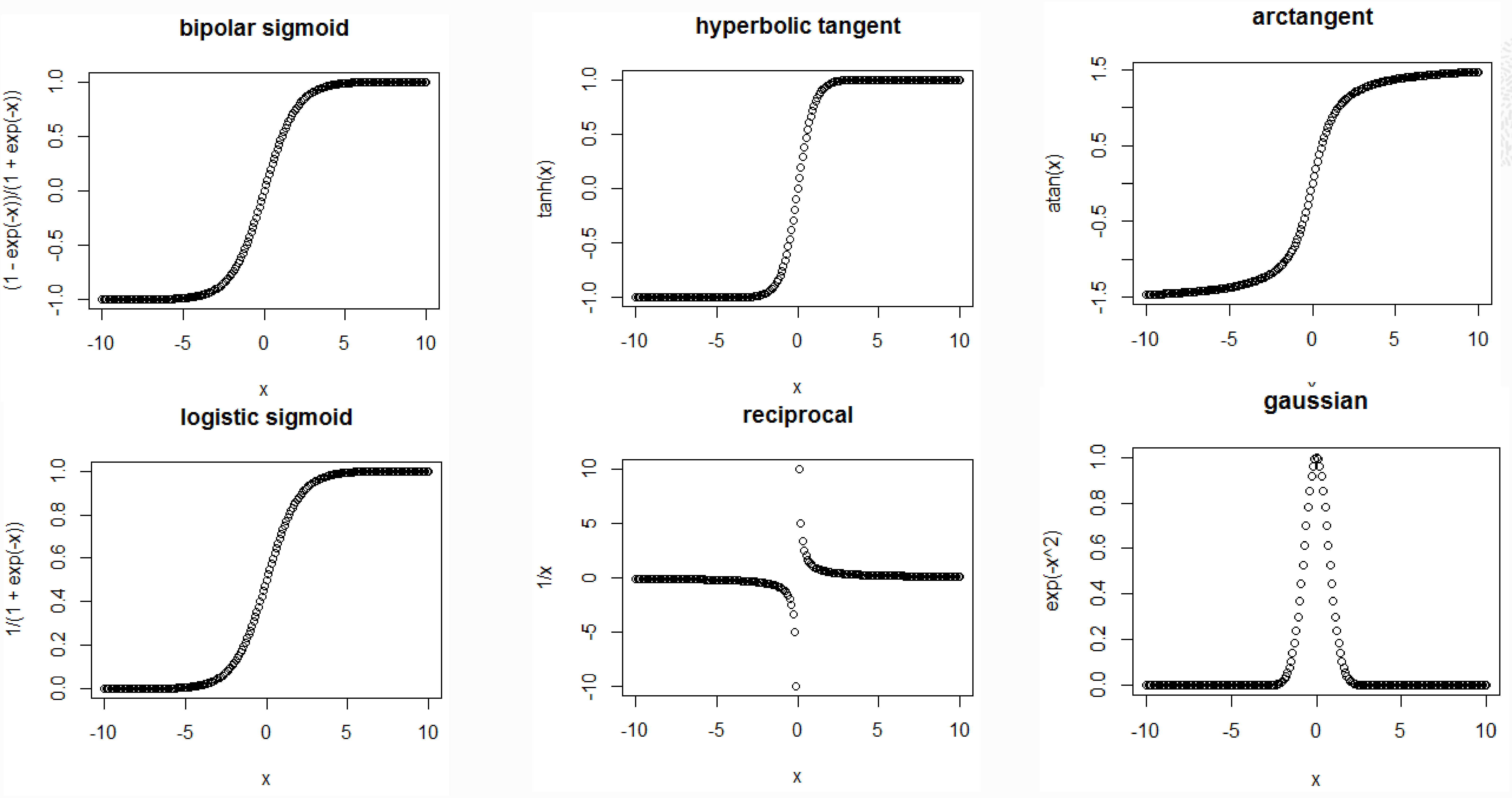
```
> sd(data$Petal.Length)
[1] 1
> mean(data$Petal.Length)
[1] -2.895326e-17
```

Copyright © 2020 Oracle and/or its affiliates.

If specified, must include one for each hidden layer and output layer

Activation Function	Activation Setting	Definition	Notes
Arctangent	atan	f(x) = arctan x	
Bipolar Sigmoid	bSigmoid	f(x) = (1 - e^{-x})/(1 + e^{-x})	Use in input data with different signs or unscaled Use on output layer when values [-1, 1]
Cosine	cos	f(x) = cos x	
Gaussian	gaussian	f(x) = e^{-x^2}	
Gauss error	gaussError	f(x) = 2/sqrt(pi) integral e^{-t^2}dt	
Gompertz	gompertz	f(x) = e^{-e^{-x}}	
Linear	linear	f(x) = x	Applicable across all data ranges
Logistic Sigmoid	sigmoid	f(x) = 1 / (1 + e^{-x})	Use on output layer when values [0..1]
Reciprocal	reciprocal	f(x) = 1 / x	Value should not include 0 value
Sigmoid Modulus	sigmoidModulus	f(x) = x / (1 + |x|)	
Sigmoid Square Root	sigmoidSqrt	f(x) = x / (1 + sqrt{1+x^2)	
Sine	sin	f(x) = sin x	
Square	square	f(x) = x^2	
Hyperbolic Tangent	tanh	f(x) = tanh x	
Wave	wave	f(x) = x / (1 + x^2)	
Entropy (output only)	entropy	f(x) = log(1 + exp(x)) – yx	Use with logistic regression

Copyright © 2020 Oracle and/or its affiliates.



Copyright © 2020 Oracle and/or its affiliates.

ore.neural

Artificial Neural Network

```
ore.neural(
   formula,
   data,
   weight            = NULL,      # initial vector of weights
   xlev              = NULL,      # named list of character vectors specifying levels for each ore.factor var
   hiddenSizes       = NULL,      # vector of nodes per layer, or none, e.g., 2 layers c(20,5)
   activations       = NULL,      # vector activation functions, including one for output
   gradTolerance     = 1E-1,      # numerical optimization stopping crit.
   maxIterations     = 200L,      # select value >= 5
   objMinProgress    = 1E-6,      # Stopping criterion: | f_current - f_previous | / ( 1 + |f| )
   lowerBound        = -0.7,      # weight initialization range
   upperBound        = 0.7,       # weight initialization range
   nUpdates          = 20L,       # number of L-BFGS update pairs
   scaleHessian      = TRUE,      # logical whether to scale inverse of Hessian matrix in L-BFGS updates
   trace             = FALSE)     # repot iteration log for big data solver
```

Copyright © 2020 Oracle and/or its affiliates.

Stopping Criteria

gradTolerance

- Affects how quickly model can converge
- Valid values: > 10^{-9}
- If > 1M observations, set to 1
- If # observations < 1000, set to between .01 and .001

objMinProgress

- Valid values [10^{-1}, 10^{-6}]
- Indicates required change from one iteration to next
- Computed as
  `| f_current - f_previous | / ( 1 + |f| )`

maxIterations

- Valid values >= 5
- Upper limit on the number of iterations

Copyright © 2020 Oracle and/or its affiliates.

Local Minima

Local Minima are non-optimal states that can improve no further with current settings and weights

To determine if a neural network is possibly in a local minima, rebuild model with different weights

- Change random seen to different value
- Change upper/lower bound of weight initialization values
- Select model with best objective function value, e.g, fit$objValue

Copyright © 2020 Oracle and/or its affiliates.

Local Minima

```
d <- data.frame(A=c(0,1,0,1),
                B=c(1,0,0,1),
                T=c(1,1,0,0))

# Run the model below 5 ~ 10 times and observe the resulting objective
# function value - the smaller, the better
library(nnet)
fit.nn <- nnet(formula = T ~ A + B, data = d, size=2)
predict(fit.nn,d)

fit.ore <- ore.neural(formula = T ~ A + B, data = ore.push(d),
       hiddenSizes = c(5000, 10, 10),
       lowerBound=-1, upperBound=1)
predict(fit.ore,ore.push(d))
```

Copyright © 2020 Oracle and/or its affiliates.

Local Minima - results

```
R> fit.nn <- nnet(formula = T ~ A + B, data = d,
size=2)
# weights:  9
initial  value 1.046487
iter  10 value 0.997966
iter  20 value 0.569304
iter  30 value 0.502784
iter  40 value 0.500426
iter  50 value 0.500050
final  value 0.500041
converged
R>
R> predict(fit.nn,d)
         [,1]
1 0.499970251
2 0.999986345
3 0.002586049
4 0.500004855
```

```
R> fit.nn <- nnet(formula = T ~ A + B, data = d, size=2)
# weights:  9
initial  value 1.042918
iter  10 value 0.992396
iter  20 value 0.682306
iter  30 value 0.666802
iter  40 value 0.666718
final  value 0.666709
converged
R> predict(fit.nn,d)
      [,1]
1 0.666687
2 0.666650
3 0.000337
4 0.666700
```

```
R> fit.ore <- ore.neural(formula = T ~ A + B, data =
ore.push(d),
+         hiddenSizes = c(5000, 10, 10),
+         lowerBound=-1, upperBound=1)
R> predict(fit.ore,ore.push(d))
        pred_T
1  0.913355525
2  1.035549253
3 -0.020444140
4 -0.001179834
```

```
R> fit.ore <- ore.neural(formula = T ~ A + B, data = ore.push(d),
+         hiddenSizes = c(5000, 10, 10),
+         lowerBound=-1, upperBound=1)
R> predict(fit.ore,ore.push(d))
    pred_T
1  1.00488
2  0.99595
3 -0.00718
4 -0.00165
```

Copyright © 2020 Oracle and/or its affiliates.

Optimization argument: nUpdates

Optimization parameter for *L-BFGS* solver

Indicates number of matrix adjustments to occur before updating Hessian matrix

Recommended ranges

- Usual models: 7..25
- If # weights > 1M: 3..25
- If highly non-linear behavior, use > 10

If you're unfamiliar with underlying techniques, don't touch

Copyright © 2020 Oracle and/or its affiliates.

Example

```
IRIS <- ore.push(iris)

fit <- ore.neural(Petal.Length ~ Petal.Width + Sepal.Length,
        data = IRIS, hiddenSizes = c(20, 5),
        activations = c('bSigmoid', 'tanh', 'linear'))

print(fit)
```

```
R> print(fit)
Number of input units        2
Number of output units       1
Number of hidden layers      2
Objective value              6.431877E+00
Solution status              Optimal
Hidden layer [1]             number of neurons 20, activation
   'bSigmoid'
Hidden layer [2]             number of neurons 5, activation 'tanh'
Output layer                 number of neurons 1, activation 'linear'
Optimization solver          L-BFGS
Scale Hessian inverse        1
Number of L-BFGS updates     20
```

```
ans <- predict(fit, newdata = IRIS,
               supplemental.cols = 'Petal.Length')
localPredictions <- ore.pull(ans)

# Inspect some predictions
head(localPredictions)

# Compute RMSE
ore.rmse <- function (pred, obs) {
  sqrt(mean((pred-obs)^2,na.rm=TRUE))
}
```

```
R> ore.rmse(localPredictions$pred_Petal.Length,
            localPredictions$Petal.Length)
[1] 0.00148768
```

Copyright © 2020 Oracle and/or its affiliates.

Example – linear regression

No hidden structure in network

```
fit <- ore.neural(Petal.Length ~ Petal.Width + Sepal.Length, data = IRIS)

print(fit)
```

```
# Print fit object
R> print(fit)
Number of input units        2
Number of output units       1
Number of hidden layers      0
Objective value              1.311757E+01
Solution status              Optimal
Output layer                 number of neurons 1, activation 'linear'
Optimization solver          L-BFGS
Scale Hessian inverse        1
Number of L-BFGS updates     20
```

Copyright © 2020 Oracle and/or its affiliates.

Model Details: Solution Status

optimal
- meets all stopping criteria

numerical difficulties encountered
- TBD

maximum iterations reached
- more iterations may be needed to improve model

insufficient memory
- could not build model with current settings due to memory

no progress
- change in objective function insufficient to make process

unbounded
- one of model parameters (weights) is greater than 1E+24 (check input data, unlikely to happen)

Copyright © 2020 Oracle and/or its affiliates.

Model Details: Objective Value

Error statistic on the model

ore.neural tries to minimize this value

Calculated as `sum((predicted - actual)^2)`

Copyright © 2020 Oracle and/or its affiliates.

ore.neural vs. nnet

OML4R...

- Is scalable
- Allows choosing wide range of activation functions
- Provides generic topology
  - unrestricted number of hidden layers, including none
- Has a parallel implementation

Copyright © 2020 Oracle and/or its affiliates.

Generalized Linear Models

Copyright © 2020 Oracle and/or its affiliates.

Generalized Linear Models

Fits generalized linear models using a Fisher scoring iteratively re-weighted least squares (IRLS) algorithm for logistic regression, probit regression, and poisson regression

Instead of the traditional step halving to prevent the selection of less optimal coefficient estimates, a line search is used to select new coefficient estimates at each iteration starting from the current coefficient estimates and moving through the Fisher scoring suggested estimates using the formula (1 - alpha) * old + alpha * suggested where alpha in [0, 2]

When the 'interp' control argument is 'TRUE', the deviance is approximated by a cubic spline interpolation; and when 'FALSE', the deviance is calculated using a follow-up data scan

Each iteration consists of two or three embedded R map/reduce operations: an IRLS operation, an initial line search operation, and an optional follow-up line search operation if 'interp = FALSE'

The IRLS map/reduce operations are on the matrix cross-products based off of 'model.matrix' or 'sparse.model.matrix' function calls depending on the underlying scarcity of the model matrix.

After the algorithm has either converged or reached the maximum number of iterations, a final embedded R map/reduce operation is used to generate the complete set of model-level statistics.

Copyright © 2020 Oracle and/or its affiliates.

ore.glm

Generalized Linear Model

```
ore.glm(formula,                # 'formula' object representing the model to be fit
        data,                   # 'ore.frame' object specifying the data for the model
        weights,                # optional 'ore.number' object specifying the model’s analytic weights
        family = gaussian(),    # 'family' object specifying the generalized linear model family details.
                                # Same type of object used for 'glm' function in the 'stats' package
        start = NULL,           # optional 'numeric' vector specifying initial coefficient estimates in
                                # the linear predictor
        control = list(...),    # optional 'list' object containing a list of fit control parameters to
                                # be interpreted by the 'ore.glm.control' function
        contrasts = NULL,       # optional named 'list' to be supplied to 'contrasts.arg' argument of 'model.matrix'
        xlev = NULL,            # optional named 'list' of 'character' vectors specifying the 'levels'
                                # for each 'ore.factor' variable
        ylev = NULL,            # optional 'character' vector to specify the response variable levels
                                # in 'binomial' generalized linear models
        yprob = NULL,           # optional numeric value between 0 and 1 specifying overall probability
                                # of 'y != ylev[1]' in 'binomial' linear models
        ...)
```

Copyright © 2020 Oracle and/or its affiliates.

ore.glm examples

Generalized Linear Model

```
library(rpart)

# Logistic regression
KYPHOSIS <- ore.push(kyphosis)
kyphFit1 <- ore.glm(Kyphosis ~ ., data = KYPHOSIS, family = binomial())
kyphFit2 <- glm(Kyphosis ~ ., data = kyphosis, family = binomial())
summary(kyphFit1)
summary(kyphFit2)

# Poisson regression
SOLDER <- ore.push(solder)
solFit1 <- ore.glm(skips ~ ., data = SOLDER, family = poisson())
solFit2 <- glm(skips ~ ., data = solder, family = poisson())
summary(solFit1)
summary(solFit2)
```

Copyright © 2020 Oracle and/or its affiliates.

ore.glm results

Generalized Linear Model

```
R> summary(kyphFit1)

Call:
ore.glm(formula = Kyphosis ~ ., data = KYPHOSIS, family = binomial())

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3124  -0.5484  -0.3632  -0.1659   2.1613

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.036934   1.449622  -1.405  0.15998
Age          0.010930   0.006447   1.696  0.08997 .
Number       0.410601   0.224870   1.826  0.06786 .
Start       -0.206510   0.067700  -3.050  0.00229 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 83.234  on 80  degrees of freedom
Residual deviance: 61.380  on 77  degrees of freedom
AIC: 69.38

Number of Fisher Scoring iterations: 4
```

```
R> summary(kyphFit2)

Call:
glm(formula = Kyphosis ~ ., family = binomial(), data =
   kyphosis)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3124  -0.5484  -0.3632  -0.1659   2.1613

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.036934   1.449575  -1.405  0.15996
Age          0.010930   0.006446   1.696  0.08996 .
Number       0.410601   0.224861   1.826  0.06785 .
Start       -0.206510   0.067699  -3.050  0.00229 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 83.234  on 80  degrees of freedom
Residual deviance: 61.380  on 77  degrees of freedom
AIC: 69.38

Number of Fisher Scoring iterations: 5
```

Copyright © 2020 Oracle and/or its affiliates.

Compare ore.glm results

```
R> summary(solFit1)
Call:
ore.glm(formula = skips ~ ., data = SOLDER, family = poisson())
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.4105  -1.0897  -0.4408   0.6406   3.7927
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.25506    0.10069 -12.465  < 2e-16 ***
OpeningM     0.25851    0.06656   3.884 0.000103 ***
OpeningS     1.89349    0.05363  35.305  < 2e-16 ***
SolderThin   1.09973    0.03864  28.465  < 2e-16 ***
MaskA3       0.42819    0.07547   5.674 1.40e-08 ***
MaskB3       1.20225    0.06697  17.953  < 2e-16 ***
MaskB6       1.86648    0.06310  29.580  < 2e-16 ***
PadTypeD6   -0.36865    0.07138  -5.164 2.41e-07 ***
PadTypeD7   -0.09844    0.06620  -1.487 0.137001
PadTypeL4    0.26236    0.06071   4.321 1.55e-05 ***
PadTypeL6   -0.66845    0.07841  -8.525  < 2e-16 ***
PadTypeL7   -0.49021    0.07406  -6.619 3.61e-11 ***
PadTypeL8   -0.27115    0.06939  -3.907 9.33e-05 ***
PadTypeL9   -0.63645    0.07759  -8.203 2.35e-16 ***
PadTypeW4   -0.11000    0.06640  -1.657 0.097591 .
PadTypeW9   -1.43759    0.10419 -13.798  < 2e-16 ***
Panel        0.11818    0.02056   5.749 8.97e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for poisson family taken to be 1)
    Null deviance: 6855.7  on 719  degrees of freedom
Residual deviance: 1165.4  on 703  degrees of freedom
AIC: 2781.6
Number of Fisher Scoring iterations: 4
```

```
R> summary(solFit2)
Call:
glm(formula = skips ~ ., family = poisson(), data = solder)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.4105  -1.0897  -0.4408   0.6406   3.7927
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.25506    0.10069 -12.465  < 2e-16 ***
OpeningM     0.25851    0.06656   3.884 0.000103 ***
OpeningS     1.89349    0.05363  35.305  < 2e-16 ***
SolderThin   1.09973    0.03864  28.465  < 2e-16 ***
MaskA3       0.42819    0.07547   5.674 1.40e-08 ***
MaskB3       1.20225    0.06697  17.953  < 2e-16 ***
MaskB6       1.86648    0.06310  29.580  < 2e-16 ***
PadTypeD6   -0.36865    0.07138  -5.164 2.41e-07 ***
PadTypeD7   -0.09844    0.06620  -1.487 0.137001
PadTypeL4    0.26236    0.06071   4.321 1.55e-05 ***
PadTypeL6   -0.66845    0.07841  -8.525  < 2e-16 ***
PadTypeL7   -0.49021    0.07406  -6.619 3.61e-11 ***
PadTypeL8   -0.27115    0.06939  -3.907 9.33e-05 ***
PadTypeL9   -0.63645    0.07759  -8.203 2.35e-16 ***
PadTypeW4   -0.11000    0.06640  -1.657 0.097590 .
PadTypeW9   -1.43759    0.10419 -13.798  < 2e-16 ***
Panel        0.11818    0.02056   5.749 8.97e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for poisson family taken to be 1)
    Null deviance: 6855.7  on 719  degrees of freedom
Residual deviance: 1165.4  on 703  degrees of freedom
AIC: 2781.6
Number of Fisher Scoring iterations: 5
```

Copyright © 2020 Oracle and/or its affiliates.

Singular Value Decomposition (SVD)

Copyright © 2020 Oracle and/or its affiliates.

Singular Value Decomposition

svd overloaded

- Execute in parallel
- Accept ore.frame objects

In-database execution to improve scalability and performance

No data movement

Copyright © 2020 Oracle and/or its affiliates.

SVD

See ?svd

- svd(x,
  nu = min(n, p),
  nv = min(n, p))
- x: a numeric ore.frame
- nu: number of left singular vectors to be computed
  0 < nu < n=nrow(x)
- nv: number of right singular vectors to be computed
  0 < nv < p=ncol(x)

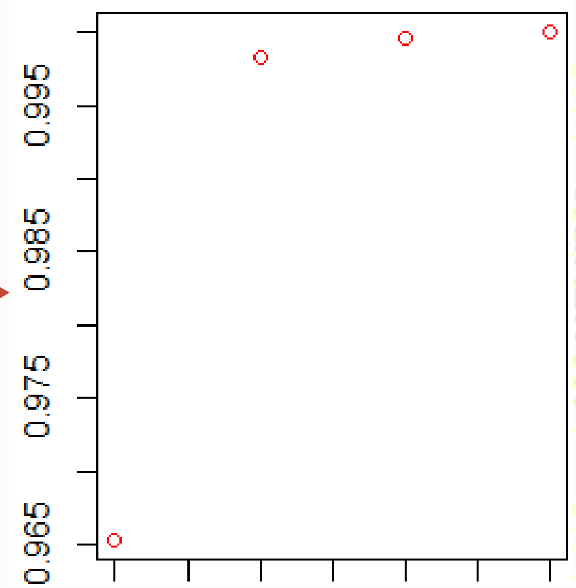
```
hilbert <- function(n) {
  i <- 1:n
  1 / outer(i - 1, i, "+")
}
X <- ore.push(as.data.frame(hilbert(9)[, 1:6]))
(s <- svd(X))
```

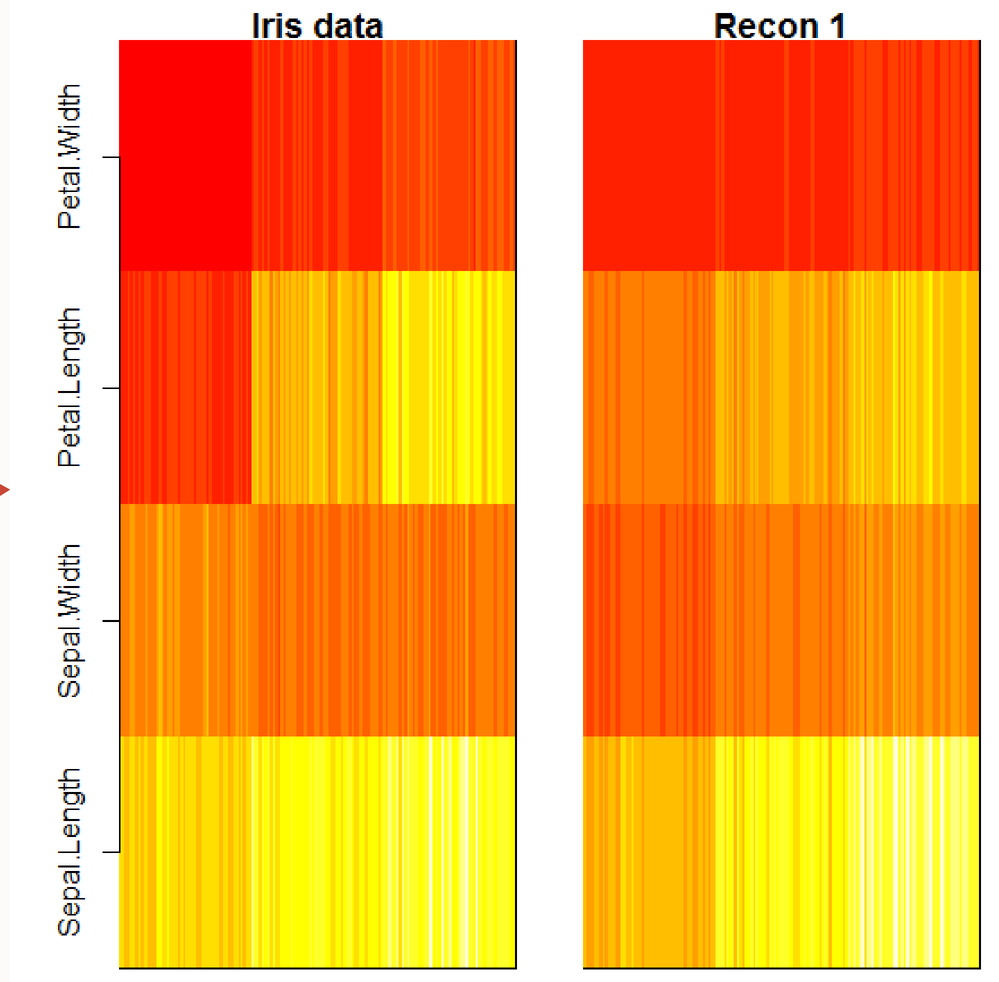
```
R> hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, "+") }
R> X <- ore.push(as.data.frame(hilbert(9)[, 1:6]))
R> (s <- svd(X))
$d
[1] 1.668433e+00 2.773727e-01 2.223722e-02 1.084693e-03 3.243788e-05 5.234864e-07

$v
          [,1]       [,2]       [,3]        [,4]        [,5]         [,6]
[1,] 0.7364928 -0.6225002 -0.2550021  0.06976287 -0.01328234 -0.001588146
[2,] 0.4432826  0.1818705  0.6866860 -0.50860089  0.19626669  0.041116974
[3,] 0.3274789  0.3508553  0.2611139  0.50473697 -0.61605641 -0.259215626
[4,] 0.2626469  0.3921783 -0.1043599  0.43747940  0.40833605  0.638901622
[5,] 0.2204199  0.3945644 -0.3509658 -0.01612426  0.46427916 -0.675826789
[6,] 0.1904420  0.3831871 -0.5110654 -0.53856351 -0.44663632  0.257248908
```

Copyright © 2020 Oracle and/or its affiliates.

SVD example using ore.frame

```
# Set up the data
dat <- iris[,-5]; mat <- as.matrix(dat); dat$IDX <- seq_len(nrow(dat))
ore.create(dat,table="DAT")
ore.exec("alter table DAT add constraint DAT primary key (\"IDX\")")
ore.sync(table = "DAT", use.keys = TRUE)

# Compute svd on ore.frame
sol <- svd(DAT[,-5])
plot(cumsum(sol$d^2/sum(sol$d^2)),col="red") # % explained variance

# Derive the U matrix since not provided with model
sol.U <- as.matrix(DAT[,-5]) %*% (sol$v) %*% diag(1./sol$d)
class(sol.U)  # ore.tblmatrix

k<-1  # use one singular vector
recon1 <- (sol.U)[,1:k,drop=FALSE] %*%
            diag((sol$d)[1:k,drop=FALSE],nrow=k,ncol=k) %*%
            t((sol$v)[,1:k,drop=FALSE])
class(recon1) # ore.tblmatrix

myviz(mat,ore.pull(recon1),lab1="Iris data", lab2="Recon 1")
```





Example inspiration: StackExchange Cross Validate

Copyright © 2020 Oracle and/or its affiliates.

Visualization function

```
myviz <- function(m1,m2,lab1, lab2) {
  x11(6,6)
  par(mfcol=c(1,2), mar=c(1,1,1,1), oma=c(0,3,1,0))
  zlim=range(m1, m2)
  image(m1, zlim=zlim, yaxt="n", xaxt="n", ylab="",
xlab="", main=lab1)
  axis(2, at=seq(0,1,,ncol(m1)), labels=colnames(m1))
  image(m2, zlim=zlim, yaxt="n", xaxt="n", ylab="",
xlab="", main=lab2)
}
```

Example inspiration: StackExchange Cross Validated

Copyright © 2020 Oracle and/or its affiliates.

prcomp and princomp

Copyright © 2020 Oracle and/or its affiliates.

Principal Component Analysis

See ?prcomp and ?princomp

Overloaded **prcomp** uses ORE’s parallel SVD

Overloaded **princomp** uses Eigen decomposition of the correlation matrix, and an ORE-specific scheme to calculate a small correlation matrix, and call R's Eigen decomposition

Copyright © 2020 Oracle and/or its affiliates.

OREpredict Package

Copyright © 2020 Oracle and/or its affiliates.

Exadata storage tier scoring for R models

Fastest way to operationalize R-based models for scoring in Oracle Database

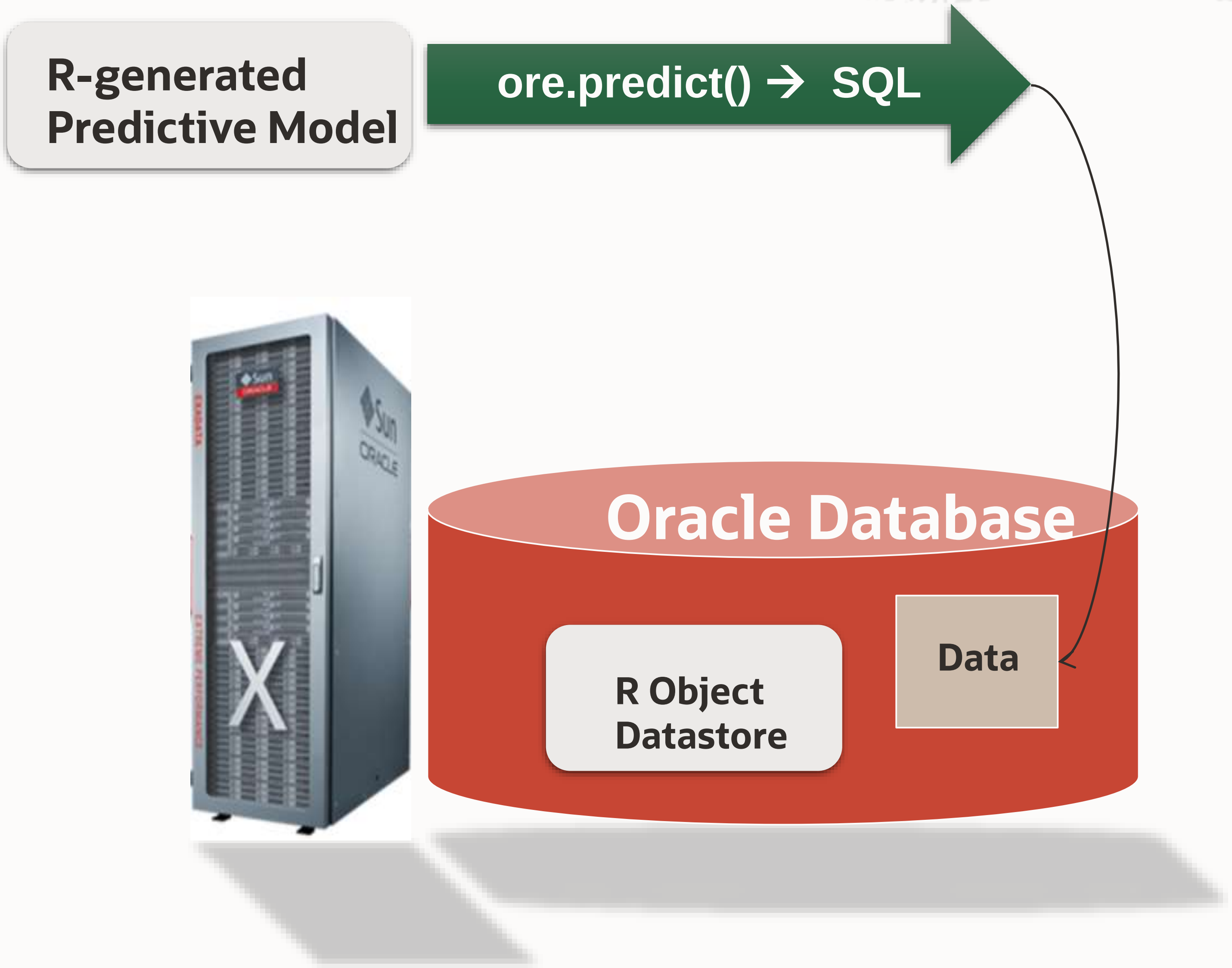
Go from model to SQL scoring in one step

- No dependencies on PMML or any other plugins

R models supported out-of-the-box include

- glm, glm.nb, hclust, kmeans, lm, multinom, nnet, rpart

Models can be managed in-database using OML4R datastore



Copyright © 2020 Oracle and/or its affiliates.

OREpredict Package

Provide a commercial grade scoring engine
- High performance
- Scalable
- Simplify application workflow

Use R-generated models to score in-database on ore.frame

Maximizes use of Oracle Database as compute engine

Function `ore.predict`
- S4 generic function
- A specific method for each model OML4R supports

Copyright © 2020 Oracle and/or its affiliates.

ore.predict 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

Copyright © 2020 Oracle and/or its affiliates.

Interface function signatures

```
  # lm, based on stats:::predict.lm
ore.predict(object, newdata, se.fit = FALSE, scale = NULL,
            df = Inf, interval = c("none","confidence","prediction"),
            level = 0.95, na.action = na.pass, pred.var = NULL,
            weights = NULL, ...)

# glm, based on stats:::predict.glm
ore.predict(object, newdata, type = c("link", "response"),
            se.fit = FALSE, dispersion = NULL, na.action = na.pass,
            ...)

# rpart, based on rpart:::predict.rpart
ore.predict(object, newdata, type = c("vector", "prob",
               "class", "matrix"),na.action = na.pass, ...)
```

```
  # matrix (for use in hclust problems)
ore.predict(object, newdata, type = c("classes",
            "distances"), method = "euclidean", p = 2,
             na.action = na.pass, ...)

# kmeans
ore.predict(object, newdata, type = c("classes",
            "distances"), na.action = na.pass, ...)

# nnet, based on nnet:::predict.nnet
ore.predict(object, newdata, type = c("raw", "class"),
            na.action = na.pass, ...)

# multinom, based on nnet:::predict.multinom
ore.predict(object, newdata, type = c("class", "probs"),
            na.action = na.pass, ...)
```

Copyright © 2020 Oracle and/or its affiliates.

Example using lm

```
irisModel <- lm(Sepal.Length ~ ., data = iris)
IRIS      <- ore.push(iris)
IRISpred  <- ore.predict(irisModel, IRIS, se.fit = TRUE,
                         interval = "prediction")
IRIS <- cbind(IRIS, IRISpred)
head(IRIS)
```

Build a typical R lm model

Use ore.predict to score data in Oracle Database using ore.frame, e.g., IRIS

```
R> irisModel <- lm(Sepal.Length ~ ., data = iris)
R> IRIS <- ore.push(iris)
R> IRISpred <- ore.predict(irisModel, IRIS, se.fit = TRUE,
+                          interval = "prediction")
R> IRIS <- cbind(IRIS, IRISpred)
R> head(IRIS)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species     PRED    SE.PRED LOWER.PRED UPPER.PRED
1          5.1         3.5          1.4         0.2  setosa 5.004788 0.04479188   4.391895   5.617681
2          4.9         3.0          1.4         0.2  setosa 4.756844 0.05514933   4.140660   5.373027
3          4.7         3.2          1.3         0.2  setosa 4.773097 0.04690495   4.159587   5.386607
4          4.6         3.1          1.5         0.2  setosa 4.889357 0.05135928   4.274454   5.504259
5          5.0         3.6          1.4         0.2  setosa 5.054377 0.04736842   4.440727   5.668026
6          5.4         3.9          1.7         0.4  setosa 5.388886 0.05592364   4.772430   6.005342
```

Copyright © 2020 Oracle and/or its affiliates.

Example using glm

Build an R glm model

Use ore.predict to score data in Oracle Database using ore.frame, e.g., INFERT

```
infertModel <- glm(case ~ age + parity + education + spontaneous + induced,
                   data = infert, family = binomial())
INFERT <- ore.push(infert)
INFERTpred <- ore.predict(infertModel, INFERT, type = "response", se.fit = TRUE)
INFERT <- cbind(INFERT, INFERTpred)
head(INFERT)
```

```
R> head(INFERT)
  education age parity induced case spontaneous stratum pooled.stratum      PRED     SE.PRED
1    0-5yrs  26      6       1    1           2       1              3 0.5721916 0.20630954
2    0-5yrs  42      1       1    1           0       2              1 0.7258539 0.17196245
3    0-5yrs  39      6       2    1           0       3              4 0.1194459 0.08617462
4    0-5yrs  34      4       2    1           0       4              2 0.3684102 0.17295285
5   6-11yrs  35      3       1    1           1       5             32 0.5104285 0.06944005
6   6-11yrs  36      4       2    1           1       6             36 0.6322269 0.10117919
```

Copyright © 2020 Oracle and/or its affiliates.

OREeda Package: Exponential Smoothing

Copyright © 2020 Oracle and/or its affiliates.

Time Series Exponential Smoothing

Used to produce smoothed data for presentation or for forecasting, i.e., making predictions

Assigns exponentially decreasing weights over time

Commonly applied to financial market and economic data

Simplest form

$$s_1 = x_0$$
$$s_t = \alpha x_{t-1} + (1-\alpha)s_{t-1} = s_{t-1} + \alpha(x_{t-1} - s_{t-1}), t > 1$$

http://en.wikipedia.org/wiki/Exponential_smoothing

Copyright © 2020 Oracle and/or its affiliates.

Time Series Exponential Smoothing

Used to produce smoothed data for presentation or for forecasting, i.e., making predictions

Assigns exponentially decreasing weights over time

Commonly applied to financial market and economic data

Simplest form

$$s_1 = x_0$$
$$s_t = \alpha x_{t-1} + (1-\alpha)s_{t-1} = s_{t-1} + \alpha(x_{t-1} - s_{t-1}), t > 1$$

http://en.wikipedia.org/wiki/Exponential_smoothing

Copyright © 2020 Oracle and/or its affiliates.

ore.esm function signature

```
ore.esm(x,
        interval = NULL,
        model = "simple",
        accumulate = "NONE",
        setmissing = "PREV",
        optim.start = c(alpha=0.3, beta=0.1),
        optim.control = list())

fitted(object, start = NULL, end = NULL, ...)

predict(object, n.ahead = 12L,  ...)

forecast.ore.esm(object, h = 12L, ...)
```

Copyright © 2020 Oracle and/or its affiliates.

ore.esm arguments

x – An ordered 'ore.vector' of time series data or transactional data. The ordering column could be either integers from 1 to the length of the time series or of type 'ore.datetime'.

interval – The interval of the time series, or the time interval by which the transactional data is to be accumulated. If the ordering column of the argument 'x' is of type 'ore.datetime', 'interval' must be specified.

- Possible values: '"YEAR"', '"QTR"', '"MONTH"', '"WEEK"', '"DAY"', '"HOUR"', '"MINUTE"', '"SECOND"‘

model – The exponential smoothing model name. Possible values: '"simple"', '"double"‘

accumulate – The method of accumulation. Possible values:

- NONE No accumulation occurs. Argument 'x' is required to be equally spaced time series observations.
- TOTAL Accumulation based on the sum of the observed values
- AVERAGE Accumulation based on the average of the observed values. The value could be abbreviated to '"AVG"'.
- MINIMUM Accumulation based on the minimum of the observed values. The value could be abbreviated to '"MIN"'.
- MAXIMUM Accumulation based on the maximum of the observed values. The value could be abbreviated to '"MAX"‘
- NOBSAccumulation based on the number of observations
- NMISS Accumulation based on the number of missing observations

Copyright © 2020 Oracle and/or its affiliates.

ore.esm arguments

setmissing: The method of treating missing values. Possible values:

- AVERAGE Missing values are set to average of the accumulated values. The value could be abbreviated to '"AVG"'
- MINIMUM Missing values are set to minimum of the accumulated values. The value could be abbreviated to '"MIN“’
- MAXIMUM Missing values are set to maximum of the accumulated values. The value could be abbreviated to '"MAX“’
- MEDIAN Missing values are set to median of the accumulated values. The value could be abbreviated to '"MED"'.
- FIRST Missing values are set to first accumulated non-missing value
- LAST Missing values are set to last accumulated non-missing value
- PREVIOUS Missing values are set to previous accumulated non-missing value. The value could be abbreviated to '"PREV"'
- NEXT Missing values are set to the next accumulated non-missing value.

optim.start: A vector with named components 'alpha' and 'beta’ containing the starting values for the optimizer. Ignored in the 'simple' model case.

optim.control: Optional list with additional control parameters passed to 'optim' in the 'double' model case. Ignored in the 'simple' model case.

Copyright © 2020 Oracle and/or its affiliates.

predict and *forecast* arguments for ore.esm

```
predict(object, n.ahead = 12L,  ...)
forecast.ore.esm(object, h = 12L, ...)
```

object: object of type 'ore.esm‘

n.ahead: number of time periods to forecast

h: number of time periods to forecast

Copyright © 2020 Oracle and/or its affiliates.

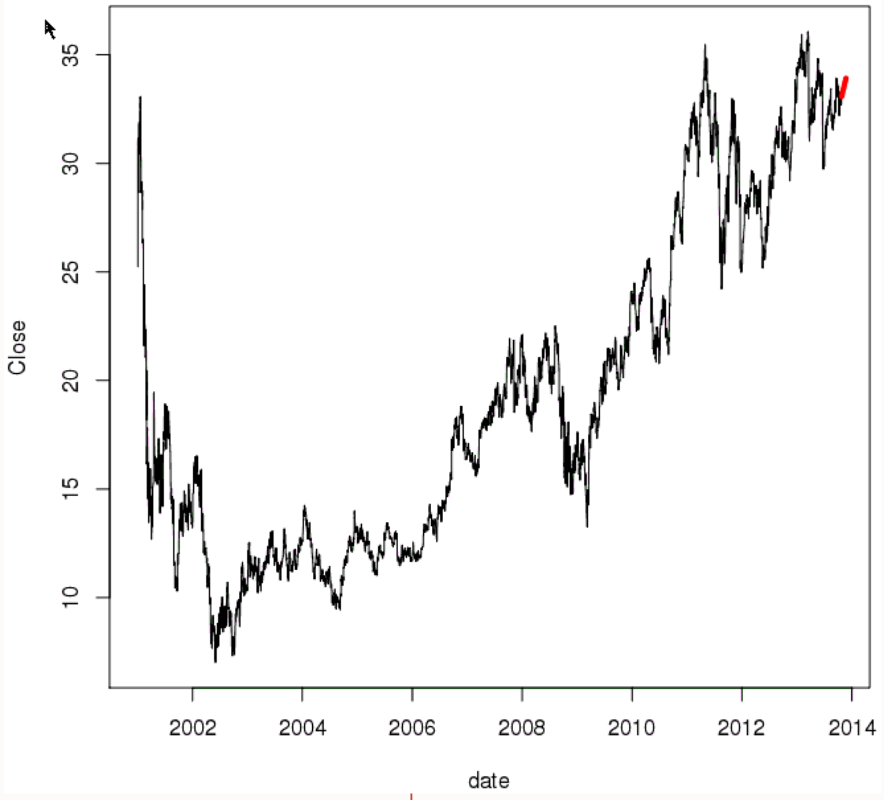
Stock Data with ore.esm

```
library(TTR)
library(zoo)
# Get data for selected stocks in XTS format
stocks <- c("orcl","ibm","sap","msft")
list.data <- vector("list",length(stocks))
for(s in stocks) {
  xts.data <- getYahooData(s, 20050101, 20180206)
  df.data  <- data.frame(xts.data)
  df.data$date   <- index(xts.data)
  df.data$symbol <- s
  df.data$Split  <- NULL
  list.data[[s]] <- df.data
}
stock.data <- data.frame(do.call("rbind",list.data))
ore.drop("STOCKS")
ore.create(stock.data,table="STOCKS")
rownames(STOCKS) <- STOCKS$date
head(STOCKS)
```

Copyright © 2020 Oracle and/or its affiliates.

Stock Data with ore.esm

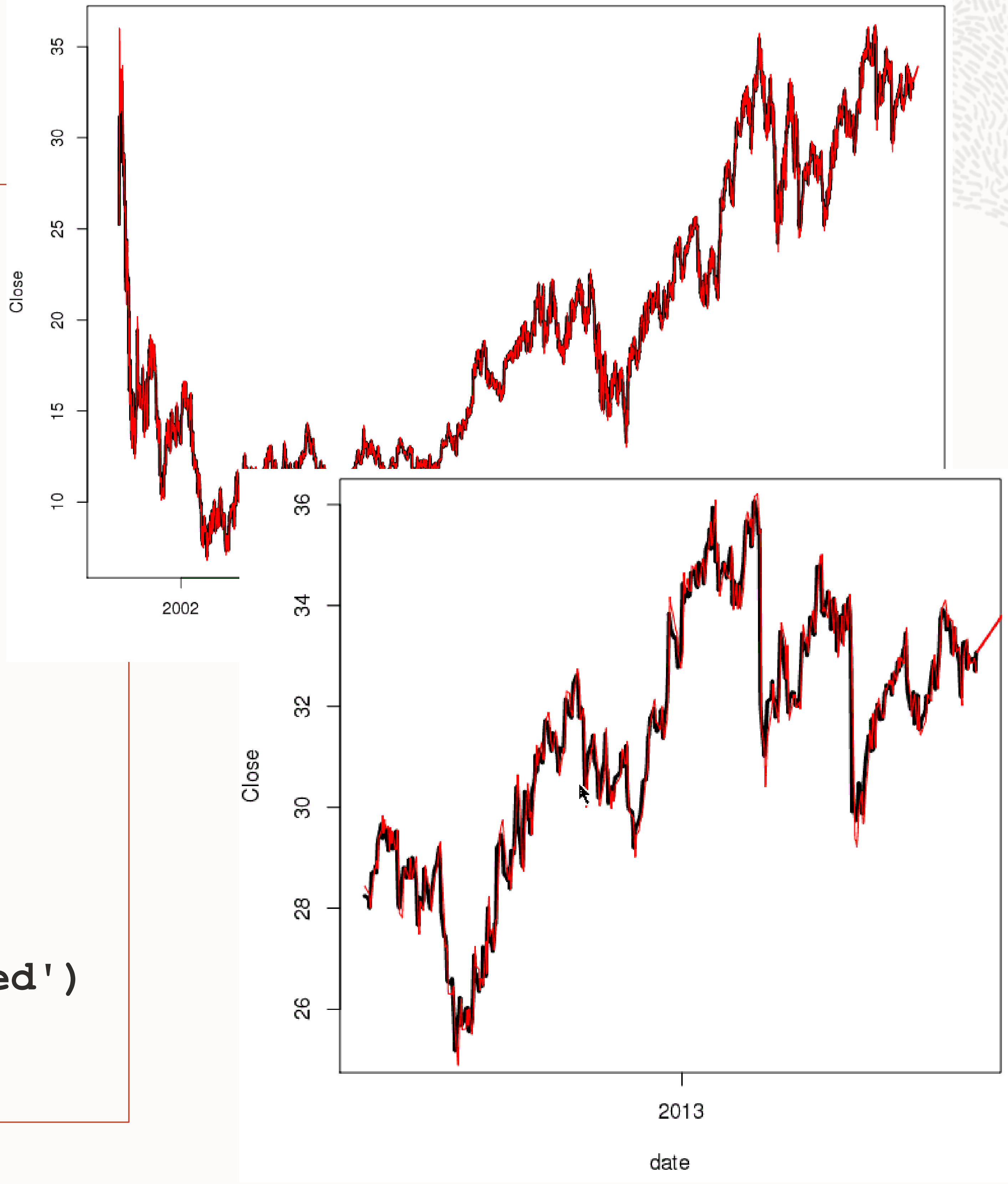
```r
orcl.stock <- ore.pull(STOCKS[STOCKS$symbol=='orcl',c("date","Close","symbol")])

ts.orcl.stock <- ts(orcl.stock$Close)

ts.sm.orcl <-ts(SMA(ts.orcl.stock,n=30),frequency=365, start=c(2008,1) )

plot(orcl.stock$date,orcl.stock$Close,type="l",col="red",xlab="Date",ylab="US$",
     main="ORCL Stock Close CLIENT-side Smoothed Series n=30 days")
lines(orcl.stock$date,ts.sm.orcl,col="blue")
legend("topleft", c("Closing","MA(30) of Closing"),
       col=c("red","blue"),lwd=2,title = "Series",bty="n")

orcl.stock <- STOCKS[STOCKS$symbol=='orcl',c("date","Close")]

dESM.mod <- ore.esm(orcl.stock$Close, "DAY", model = "double")
dESM.predict <- predict(dESM.mod, 30)
plot(orcl.stock,type="l")
lines(dESM.predict,col="red",lwd=4)
```



Copyright © 2020 Oracle and/or its affiliates.

Using supplemental functions

```
dESM.mod <- ore.esm(orcl.stock$Close, "DAY",
                    model = "double",
                    optim.start=c(alpha=0.5,beta=0.5))
dESM.predict <- predict(dESM.mod, 30)
dESM.fitted <-  fitted(dESM.mod)
plot(orcl.stock,type="l",lwd=2)
lines(dESM.predict,col="red",lwd=2)
lines(orcl.stock[,1], dESM.fitted, col='red',lwd=1)

row.idx = 1500:2722
plot(orcl.stock[row.idx,], type="l",lwd=3)
lines(orcl.stock[row.idx,1], dESM.fitted[row.idx], col='red')
lines(dESM.predict,col="red",lwd=2)
```



Copyright © 2020 Oracle and/or its affiliates.

Summary

OREdm

- Oracle Data Mining algorithms exposed through R interface
- Attribute Importance, Decision Trees, GLM, KMeans, O-Cluster, Naïve Bayes, SVD, SVM, NMF, Association Rules, Explicit Semantic Analysis

OREeda

- Functions for exploratory data analysis for Base SAS equivalents

OREmodels

- ore.lm, ore.stepwise, ore.neural, ore.glm, ore.randomForest

OREpredict

- Score R models in the database

OREstats

- In-database statistical computations exposed through R interface

Copyright © 2020 Oracle and/or its affiliates.

For more information…

oracle.com/machine-learning

See also AskTOM OML Office Hours

Copyright © 2020 Oracle and/or its affiliates.

Thank You

Mark Hornick
Oracle Machine Learning Product Management