ORACLE

Session 6: Oracle Machine Learning for R

Statistics Engine

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.

Oracle R Enterprise Statistics Engine

Example Features

Special Functions

- Gamma function
- Natural logarithm of the Gamma function
- Digamma function
- Trigamma function
- Error function
- Complementary error function

Tests

- Chi-square, McNemar, Bowker
- Simple and weighted kappas
- Cochran-Mantel-Haenzel correlation
- Cramer's V
- Binomial, KS, t, F, Wilcox

Base SAS equivalents

- Freq, Summary, Sort
- Rank, Corr, Univariate

Density, Probability, and Quantile Functions

- Beta distribution
- Binomial distribution
- Cauchy distribution
- Chi-square distribution
- Exponential distribution
- F-distribution
- Gamma distribution
- Geometric distribution
- Log Normal distribution
- Logistic distribution
- Negative Binomial distribution
- Normal distribution
- Poisson distribution
- Sign Rank distribution
- Student's t distribution
- Uniform distribution
- Weibull distribution
- Density Function
- Probability Function
- Quantile

In-Database SQL-based Statistical Functions

Supporting OML4R scalability through in-database execution

Native Oracle Database Statistical Functions

SQL for Analysis and Reporting

DMBS_STATS_FUNCS Reference

Descriptive Statistics

Hypothesis Testing - Parametric Tests

Crosstab Statistics

Hypothesis Testing - Non-Parametric Tests

Non-Parametric Correlation

Statistical Tests – Examples

```
ore.create(iris,table="IRIS_TABLE")
IRIS_TABLE$PETALBINS=ifelse(IRIS_TABLE$Petal.Length < 2, 1, 2)
# Binomial Test
binom.test(IRIS_TABLE$PETALBINS)

# Chi Square Test
chisq.test(IRIS_TABLE$PETALBINS)

# One sample K-S Test for given probabilities
ks.test(IRIS_TABLE$Petal.Length, "pexp", rate=4)

# Two sample K S Test
ks.test(IRIS_TABLE$Petal.Length, IRIS_TABLE$Sepal.Length)

# T-test with different alternate hypothesis possibilities */
t.test(IRIS_TABLE$Petal.Length, alternative="two.sided", mu=0, conf.level=0.9)

# F test to compare variances
var.test(IRIS_TABLE$Petal.Length, IRIS_TABLE$Sepal.Length, ratio=0.75,
         alternative="two.sided", conf.level=0.9)
```

Statistical Tests – Results

```
> IRIS_TABLE$PETALBINS=ifelse(IRIS_TABLE$Petal.Length < 2, 1, 2)
> # Binomial Test
> binom.test(IRIS_TABLE$PETALBINS)

 Exact binomial test

data:  x
number of successes = 50, number of trials = 150, p-value =
5.448e-05
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.2585564 0.4148430
sample estimates:
probability of success
             0.3333333

>
> # Chi Square Test
> chisq.test(IRIS_TABLE$PETALBINS)

 Chi-squared test for given probabilities

data:  x
X-squared = 16.6667, df = 1, p-value = 4.456e-05

>
> # One sample K-S Test for given probabilities
> ks.test(IRIS_TABLE$Petal.Length, "pexp", rate=4)

 One-sample Kolmogorov-Smirnov test

data:  IRIS_TABLE$Petal.Length
D = 0.975, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
> # Two sample K S Test
> ks.test(IRIS_TABLE$Petal.Length, IRIS_TABLE$Sepal.Length)

 Two-sample Kolmogorov-Smirnov test

data:  IRIS_TABLE$Petal.Length and IRIS_TABLE$Sepal.Length
D = 0.56, p-value < 2.2e-16
alternative hypothesis: two-sided

>
> # T-test with different alternate hypothesis possibilities */
> t.test(IRIS_TABLE$Petal.Length, alternative="two.sided", mu=0,
conf.level=0.9)

 One Sample t-test

data:  IRIS_TABLE$Petal.Length
t = 26.0726, df = 149, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 3.519434 3.996566
sample estimates:
mean of x
    3.758

>
> # F test to compare variances
> var.test(IRIS_TABLE$Petal.Length, IRIS_TABLE$Sepal.Length, ratio=0.75,
+          alternative="two.sided", conf.level=0.9)

 F test to compare two variances

data:  IRIS_TABLE$Petal.Length and IRIS_TABLE$Sepal.Length
F = 6.0596, num df = 149, denom df = 149, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal to 0.75
90 percent confidence interval:
 3.468071 5.955583
sample estimates:
ratio of variances
           4.54471
```

Oracle R Enterprise “PROCs”

SUMMARY / MEANS

RANK

SORT

CROSSTAB

FREQ

CORR

UNIVARIATE

ore.summary

```
ore.summary(data, var, stats = c("n", "mean", "min", "max"),
            class = NULL, types = NULL, ways = NULL, weight
= NULL,
            order = NULL, maxid = NULL, minid = NULL, mu =
0,
            no.type = FALSE, no.freq = FALSE)
```

Provides descriptive statistics and extensive analysis of columns in an ore.frame with flexible row aggregations

Statistics, e.g.,

- n, count/cnt, num miss, mean/avg, min, max, sum, sumwgt
- corrected and uncorrected sum of squares, range of values, stddev, stderr, variance
- t-test for testing hypothesis that the population mean is zero
- kurtosis, skew, Coefficient of Variation
- quantiles: p1,p5,p10,p25,p50,p75,p90,p95,p99,qrange
- 1 and 2 sided Confidence Limits for the mean: clm, rclm, lclm
- extreme value tagging, …and others

Simple syntax abstracting complex SQL queries

ore.summary – Parameters

data: ore.frame on which to compute descriptive statistics

var: vector of column names *on which to apply statistics functions*

stats: list of statistics functions available to be applied on *var columns* - *e.g., mean,min,max,cnt,n,nmiss,css,uss,cv,sum,sumwgt,range,stddev,stderr,var,t,probt,kurt, skew,p1,p5,p10,p25,p50,p75,p90,p95,p99,qrange,lclm,rclm,clm,mode*

class: vector of column names *to aggregate (i.e., SQL group by)*

types: list of character string vectors specifying the combinations of column names in *class* within which the aggregations will be executed in the returning summary

ways: integer vector with each value indicating number of columns in *class* used to generate types

weight: column name whose numeric values provide a multiplicative factor for *var columns*

order: string vector specifying sorting criteria: 'freq' or '-freq', 'type' or '-type', 'class' or '-class'

maxid, minid: for each group optionally list max/min value from other columns in

mu: single number or a vector whose elements correspond to each value in *var* to supply additional numeric parameters for some statistics

Examples

```
IRIS <- ore.push(iris)
ore.summary(IRIS, c("Sepal.Length", "Petal.Length"))
ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "std", "p10"), class="Species")
ore.summary(IRIS, list(c("Sepal.Length", "Petal.Length"), "Sepal.Width"), c(AVG="mean", "std"),
class="Species")
ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "std"), class="Species",
weight="Sepal.Width")
ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "std"),
class=c("Species", "Petal.Width"), types=list("Species", c("Species", "Petal.Width")),
order=c("type", "-freq", "class"))
ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "std"),
class=c("Species", "Petal.Width"), ways=1, order=c("type", "-freq", "class"))
ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "prt"), class="Species", mu=c(5.8, 3.7))
ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), "mean",
class="Species", maxid=c(Sepal.Length="Sepal.Width", Petal.Length="Petal.Width"))
ore.summary(IRIS, c("Sepal.Length", "Sepal.Width"), "max",
maxid=c(Sepal.Length="Species", Sepal.Width="Species"))
```

ore.summary – Results

```
R> IRIS <- ore.push(iris)
R> ore.summary(IRIS, c("Sepal.Length", "Petal.Length"))
  FREQ N(Sepal.Length) N(Petal.Length) MEAN(Sepal.Length) MEAN(Petal.Length) MIN(Sepal.Length) MIN(Petal.Length) MAX(Sepal.Length)
1  150             150             150          5.843333              3.758               4.3                 1               7.9
  MAX(Petal.Length)
1               6.9
R> ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "std", "p10"), class="Species")
     Species FREQ TYPE MEAN(Sepal.Length) MEAN(Petal.Length) STD(Sepal.Length) STD(Petal.Length) P10(Sepal.Length)
1     setosa   50    0          5.006000              1.462        0.3524897        0.1736640              4.59
2 versicolor   50    0          5.936000              4.260        0.5161711        0.4699110              5.38
3  virginica   50    0          6.588000              5.552        0.6358796        0.5518947              5.80
4       <NA>  150    1          5.843333              3.758        0.8280661        1.7652982              4.80
  P10(Petal.Length)
1              1.30
2              3.59
3              4.90
4              1.40
R> ore.summary(IRIS, list(c("Sepal.Length", "Petal.Length"), "Sepal.Width"), c(AVG="mean", "std"), class="Species")
     Species FREQ TYPE AVG(Sepal.Length) AVG(Petal.Length)       STD
1     setosa   50    0          5.006000              1.462 0.3790644
2 versicolor   50    0          5.936000              4.260 0.3137983
3  virginica   50    0          6.588000              5.552 0.3224966
4       <NA>  150    1          5.843333              3.758 0.4358663
```

```
R> ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "std"), class="Species", weight="Sepal.Width")
     Species FREQ TYPE MEAN(Sepal.Length) MEAN(Petal.Length) STD(Sepal.Length) STD(Petal.Length)
1     setosa   50    0           5.034364           1.465344         0.3517111         0.1734107
2 versicolor   50    0           5.966137           4.289242         0.5095037         0.4548644
3  virginica   50    0           6.618897           5.575521         0.6276732         0.5454365
4       <NA>  150    1           5.829546           3.650894         0.8343801         1.8126149
R> ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "std"), class=c("Species", "Petal.Width"), types=list("Species", c("Species", "Petal.
Width")), order=c("type", "-freq", "class"))
      Species Petal.Width FREQ TYPE MEAN(Sepal.Length) MEAN(Petal.Length) STD(Sepal.Length) STD(Petal.Length)
1      setosa         0.2   29    0           4.972414           1.444828         0.3544579         0.17234974
2  versicolor         1.3   13    0           5.884615           4.176923         0.3804518         0.25869495
3   virginica         1.8   11    0           6.445455           5.381818         0.4568668         0.51926522
4  versicolor         1.5   10    0           6.190000           4.580000         0.4724640         0.20976177
5   virginica         2.3    8    0           6.912500           5.700000         0.5436320         0.59761430
6      setosa         0.3    7    0           4.971429           1.428571         0.3988077         0.13801311
7      setosa         0.4    7    0           5.300000           1.571429         0.2449490         0.18898224
8  versicolor         1.0    7    0           5.414286           3.628571         0.4450789         0.31997024
9  versicolor         1.4    7    0           6.357143           4.500000         0.6133437         0.30550505
10  virginica         2.0    6    0           6.650000           5.550000         0.9710819         0.78676553
11  virginica         2.1    6    0           6.916667           5.783333         0.4070217         0.43550737
12     setosa         0.1    5    0           4.820000           1.380000         0.3271085         0.16431677
13 versicolor         1.2    5    0           5.780000           4.240000         0.2167948         0.32093613
14  virginica         1.9    5    0           6.340000           5.320000         0.6542171         0.44944410
15 versicolor         1.1    3    0           5.400000           3.566667         0.2645751         0.49328829
16 versicolor         1.6    3    0           6.100000           4.766667         0.1732051         0.30550505
17  virginica         2.2    3    0           6.866667           6.033333         0.7234178         0.58594653
18  virginica         2.4    3    0           6.266667           5.433333         0.4509250         0.28867513
19  virginica         2.5    3    0           6.733333           5.933333         0.4509250         0.20816660
20  virginica         1.5    2    0           6.150000           5.050000         0.2121320         0.07071068
21     setosa         0.5    1    0           5.100000           1.700000                NA                 NA
22     setosa         0.6    1    0           5.000000           1.600000                NA                 NA
23 versicolor         1.7    1    0           6.700000           5.000000                NA                 NA
24 versicolor         1.8    1    0           5.900000           4.800000                NA                 NA
25  virginica         1.4    1    0           6.100000           5.600000                NA                 NA
26  virginica         1.6    1    0           7.200000           5.800000                NA                 NA
27  virginica         1.7    1    0           4.900000           4.500000                NA                 NA
28     setosa          NA   50    1           5.006000           1.462000         0.3524897         0.17366400
29 versicolor          NA   50    1           5.936000           4.260000         0.5161711         0.46991098
30  virginica          NA   50    1           6.588000           5.552000         0.6358796         0.55189470
31       <NA>          NA  150    3           5.843333           3.758000         0.8280661         1.76529823
```

```
R> ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "std"), class=c("Species", "Petal.Width"), ways=1, order=c("type", "-freq", "class"))
      Species Petal.Width FREQ TYPE MEAN(Sepal.Length) MEAN(Petal.Length) STD(Sepal.Length) STD(Petal.Length)
1      setosa          NA   50    1           5.006000           1.462000         0.3524897         0.1736640
2  versicolor          NA   50    1           5.936000           4.260000         0.5161711         0.4699110
3   virginica          NA   50    1           6.588000           5.552000         0.6358796         0.5518947
4        <NA>         0.2   29    2           4.972414           1.444828         0.3544579         0.1723497
5        <NA>         1.3   13    2           5.884615           4.176923         0.3804518         0.2586949
6        <NA>         1.5   12    2           6.183333           4.658333         0.4323999         0.2644319
7        <NA>         1.8   12    2           6.400000           5.333333         0.4631905         0.5228129
8        <NA>         1.4    8    2           6.325000           4.637500         0.5750776         0.4808846
9        <NA>         2.3    8    2           6.912500           5.700000         0.5436320         0.5976143
10       <NA>         0.3    7    2           4.971429           1.428571         0.3988077         0.1380131
11       <NA>         0.4    7    2           5.300000           1.571429         0.2449490         0.1889822
12       <NA>         1.0    7    2           5.414286           3.628571         0.4450789         0.3199702
13       <NA>         2.0    6    2           6.650000           5.550000         0.9710819         0.7867655
14       <NA>         2.1    6    2           6.916667           5.783333         0.4070217         0.4355074
15       <NA>         0.1    5    2           4.820000           1.380000         0.3271085         0.1643168
16       <NA>         1.2    5    2           5.780000           4.240000         0.2167948         0.3209361
17       <NA>         1.9    5    2           6.340000           5.320000         0.6542171         0.4494441
18       <NA>         1.6    4    2           6.375000           5.025000         0.5678908         0.5737305
19       <NA>         1.1    3    2           5.400000           3.566667         0.2645751         0.4932883
20       <NA>         2.2    3    2           6.866667           6.033333         0.7234178         0.5859465
21       <NA>         2.4    3    2           6.266667           5.433333         0.4509250         0.2886751
22       <NA>         2.5    3    2           6.733333           5.933333         0.4509250         0.2081666
23       <NA>         1.7    2    2           5.800000           4.750000         1.2727922         0.3535534
24       <NA>         0.5    1    2           5.100000           1.700000                NA                NA
25       <NA>         0.6    1    2           5.000000           1.600000                NA                NA
26       <NA>          NA  150    3           5.843333           3.758000         0.8280661         1.7652982
R> ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), c("mean", "prt"), class="Species", mu=c(5.8, 3.7))
     Species FREQ TYPE MEAN(Sepal.Length) MEAN(Petal.Length) PRT(Sepal.Length) PRT(Petal.Length)
1     setosa   50    0           5.006000              1.462     5.204621e-21      2.403948e-56
2 versicolor   50    0           5.936000              4.260     6.845138e-02      4.284727e-11
3  virginica   50    0           6.588000              5.552     1.334983e-11      1.605835e-28
4       <NA>  150    1           5.843333              3.758     5.225603e-01      6.879680e-01
R> ore.summary(IRIS, c("Sepal.Length", "Petal.Length"), "mean", class="Species", maxid=c(Sepal.Length="Sepal.Width", Petal.Length="Petal.Width"))
     Species FREQ TYPE MEAN(Sepal.Length) MEAN(Petal.Length) MAXID(Sepal.Length->Sepal.Wi1) MAXID(Petal.Length->Petal.Wi2)
1     setosa   50    0           5.006000              1.462                           4.0                           0.2
2 versicolor   50    0           5.936000              4.260                           3.2                           1.6
3  virginica   50    0           6.588000              5.552                           3.8                           2.3
4       <NA>  150    1           5.843333              3.758                           3.8                           2.3
```

ore.summary - Examples

Compute the mean, standard deviation, and count for arrival delay for each destination airport

```
res <- ore.summary(data=ONTIME_S,
                   var='ARRDELAY',
                   class='DEST',
                   stats=c('mean','std','n'))
head(res)
```

```
R> res <- ore.summary(data=ONTIME_S,
+                     var='ARRDELAY',
+                     class='DEST',
+                     stats=c('mean','std','n'))
R> res
    DEST   FREQ TYPE         MEAN        STD      N
1    ABE    237    0   5.68141593  26.678086    226
2    ABI     34    0  21.38235294  61.406255     34
3    ABQ   1357    0   6.99331352  24.111867   1346
4    ABY     10    0  18.80000000  44.263855     10
5    ACK      3    0  11.66666667  16.072751      3
6    ACT     33    0   1.63636364  16.607262     33
7    ACV     41    0   8.07317073  17.757238     41
8    ACY     18    0   3.27777778  16.640862     18
9    ADK      2    0  -0.50000000  23.334524      2
10   ADQ     20    0  12.78947368  30.817706     19
11   AEX     24    0  13.70833333  41.577041     24
12   AGS     81    0  14.33750000  33.809947     80
```

- Compute the maximum arrival and departure delay for each airline and the corresponding destination airport

```
ore.summary(ONTIME_S, class="UNIQUECARRIER",
            var = list("ARRDELAY", "DEPDELAY"),
            stats=c(MAXARRDELAY="max", MAXDEPDELAY="max"),
            maxid=c(ARRDELAY="DEST"),
            minid=c(DEPDELAY="DEST"))
```

Results next slide…

ore.summary Results

```
R> ore.summary(ONTIME_S, class="UNIQUECARRIER",
+              var = list("ARRDELAY", "DEPDELAY"),
+              stats=c(MAXARRDELAY="max", MAXDEPDELAY="max"),
+              maxid=c(ARRDELAY="DEST"),
+              minid=c(DEPDELAY="DEST"))
   UNIQUECARRIER   FREQ TYPE MAXARRDELAY MAXDEPDELAY MAXID MINID
1             9E    747    0         247         249   MSP   BDL
2             AA  27655    0        1216        1200   DFW   BOS
3             AQ    234    0         123         114   HNL   HNL
4             AS   5174    0         333         364   SEA   ANC
5             B6   1187    0         322         347   LAS   JFK
6             CO  15299    0         929         947   IAH   MOT
7             DH   1045    0         348         339   IAD   CVG
8             DL  30305    0         530        1438   LAX   RIC
9             EA   2326    0         378         403   ATL   LGA
10            EV   2338    0         683         687   ATL   AMA
11            F9    481    0         191         197   DEN   DEN
12            FL   1760    0         407         415   ATL   HPN
13            HA    353    0         162         158   ITO   HNL
14            HP   6820    0         462         476   IND   SUX
15         ML(1)    134    0         167          60   MDW   BOS
16            MQ   5943    0        1025        1024   LAX   BOS
17            NW  18656    0        1423        1432   DTW   MSO
18            OH   2100    0         415         350   CVG   CVG
19            OO   4294    0         338         316   PDX   LAX
20         PA(1)    734    0         341         390   LAX   JFK
21            PI   2320    0         340         333   CLT   SAV
22            PS    416    0         190         188   YKM   SFO
23            TW   7477    0         412         420   PDX   PUB
24            TZ    300    0         910         132   MDW   MDW
25            UA  24307    0         484         493   ORD   MIA
26            US  25745    0         427         433   GSO   CLT
27            WN  27323    0         455         455   HOU   MSY
28            XE   3300    0         395         329   IAH   IAH
29            YV   1159    0         385         366   IAD   ORD
30          <NA> 219932    1        1423        1438   DTW   RIC
```

ore.rank

Enables investigation of distribution of values in numeric columns of an ore.frame

Highlights

- Ranking within groups
- Partition rows into groups based on rank tiles
- Cumulative percentages and percentiles
- Treatment of ties
- Calculation of normal scores from ranks

Simple syntax abstracting complex SQL queries

ore.rank – Parameters

Returns an ore.frame as output in all cases

data : ore.frame of the data to compute rankings on

var : numeric columns in data to rank

desc : ranks in descending (asc is default) order if TRUE

groups : partition rows into #groups based on ranks. For percentiles, #groups=100, For deciles #groups=10, For quartiles #groups=4.

group.by : rank each group identified by group.by columns separately

ties : specification of tie treatment. Assign largest of/smallest of/mean of corresponding ranks of tied values

fraction : rank of a column value ÷ # non missing column values

nplus1 : rank of a column value ÷ # non missing column values + 1

- Fraction and nplus1 options can be used to estimate cumulative distribution function

percent : (rank of a column value ÷ # non missing column values) * 100

Scoring Methods

- To compute Exponential scores from ranks use savage
- To compute normal scores – Use one of blom, tukey or vw (Van Der Waerden)

ore.rank – Examples

```
# Rank 2 columns and report them as derived columns
x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass')
class(x)

y <- ore.sort(data=x, by='RankOfAge')

# Handling of ties
x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass', ties='low')
head(x,10)

# Rank within groups
x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass', group.by='COUNTRY')
head(x,10)
```

ore.rank – Examples

```
# Partition rows into groups e.g. Deciles
x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass',groups=10)
head(x)

# Partition rows into groups e.g. Quartiles
x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass',groups=4)
head(x)

# Estimating cumulative distribution function
x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass',nplus1=TRUE)
head(x)

# Scores calculation
x <- ore.rank(data=NARROW, var='AGE=RankOfAge,
              CLASS=RankOfClass',score='savage', groups=100, group.by='COUNTRY')
head(x)
x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass',score='blom')
head(x)
```

ore.sort

Enables flexible sorting of columns in a data frame

Can be used with other data pre-processing functions

- Sorting happens in the database
- (Top k) results of sorting can be provided as input to R visualization

Supports database nls.sort option

ore.sort - Parameters

Returns an ore.frame as output in all cases

data : ore.frame of the data to be sorted

by : columns in data to sort

nls.sort: A character string specifying Oracle Database NLS_SORT options

reverse: Allows optional reversal of collation order for character variables (TRUE/FALSE)

stable : Allows relative order to be maintained within sorted groups (TRUE/FALSE)

unique.keys : Allows optional removal of rows with duplicate values in the column(s) being sorted from appearing in the result (TRUE/FALSE)

unique.data: Allows optional removal of duplicate rows from appearing in the result (TRUE/FALSE)

ore.sort – Examples

```
# Sort all specified columns in desc order
x <- ore.sort(data=NARROW,by='AGE,GENDER', reverse=TRUE)
head(x)
# Sort AGE in desc order but GENDER in ascending order
x <- ore.sort(data=NARROW,by='-AGE,GENDER')
head(x)
# Keep just 1 row per unique value of AGE
x <- ore.sort(data=NARROW,by='AGE', unique.key=TRUE)
head(x)
# Remove duplicate rows
x <- ore.sort(data=NARROW,by='AGE', unique.data=TRUE)
head(x)
# Remove duplicate rows as well as rows with duplicate values for AGE
x <- ore.sort(data=NARROW,by='AGE', unique.data=TRUE, unique.key = TRUE)
head(x)
# Maintain relative order within sorted output
x <- ore.sort(data=NARROW,by='AGE', stable=TRUE)
head(x)
```

ore.sort – Examples

Sort the ONTIME_S data by airline descending and departure delay ascending

```
sortedOntime <- ore.sort(data=ONTIME_S, by='-UNIQUECARRIER,DEPDELAY')
head(sortedOntime[,c(11,18)], 20)
```

Sort ONTIME_S by airline and departure delay, but select one of each combination, e.g., unique key

```
sortedOntime2 <-
ore.sort(ONTIME_S,by='UNIQUECARRIER,DEPDELAY',unique.key=TRUE)
head(sortedOntime2[,c(11,18)], 20)
```


ore.crosstab - Basics

Input data set

	gender	age
Jill	female	12
Tom	male	11
Bob	male	12
Sally	female	13
Leigha	female	11
Phil	male	11

One way tables

gender	male	female
frequency	3	3

age	11	12	13
frequency	3	2	1

2 way table

	11	12	13
female	1	1	1
male	2	0	1

A general KxL table

	Column 1	Column 2	...	Column j	...	Column L	Total
Row 1	x_{11}	x_{12}	...	x_{1j}	...	x_{1L}	$x_{1.}$
Row 2	x_{21}	x_{22}	...	x_{2j}	...	x_{2L}	$x_{2.}$
⋮	⋮	⋮		⋮		⋮	⋮
Row i	x_{i1}	x_{i2}	...	x_{ij}	...	x_{iL}	$x_{i.}$
⋮	⋮	⋮		⋮		⋮	⋮
Row K	x_{K1}	x_{K2}	...	x_{Kj}	...	x_{KL}	$x_{K.}$
Total	$x_{.1}$	$x_{.2}$	...	$x_{.j}$	...	$x_{.L}$	n

ore.crosstab

Enables cross column frequency analysis of an ore.frame

A sophisticated variant of R's function **table ()** for two variables

Builds tables of frequency counts across columns of a data frame

Required as a pre-cursor to frequency analysis using ore.freq

R translated to 100% SQL

ore.crosstab – Parameters

Returns an ore.frame as output in all cases, except when multiple tables are created – in this case an ore.list is returned

expr: cross tabulation definition in the form

[COLUMN_SPEC] ~ COLUMN_SPEC [* <WEIGHTING COLUMN>]
[/ <GROUPING COLUMN>]
[^ <STRATIFICATION COLUMN>]
[| ORDER_SPECIFICATION]

COLUMN_SPEC is <column-name>[+COLUMN_SET][+COLUMN_RANGE]

COLUMN_SET is <column_name>[+COLUMN_SET]

COLUMN_RANGE is <FROM COLUMN>-<TO COLUMN>
ORDER_SPEC is one of [-]NAME, [-]DATA, [-]FREQ, or INTERNAL

data: ore.frame of data to cross tabulate

group.by: as many cross tabulations as unique values in grouping columns

order: optional sorting of table data

[-] NAME: Sort by tabulation column names, [-]FREQ: Sort by frequency counts in the table

weights : Column of data that indicates frequency of occurrence of the corresponding row

where: An optional character vector specifying arbitrary partitions of argument data

strata : Column name used to cluster, or group, the data in combination

ore.crosstab – Examples

```
# For comparison, look at R's table function on 2 columns
table(NARROW$MARITAL_STATUS, NARROW$GENDER)

# Corresponding ore.crosstab(), extensible to more than 2 columns
ore.crosstab(MARITAL_STATUS ~ GENDER, data=NARROW)

# MARITAL_STATUS x GENDER and MARITAL_STATUS x CLASS
x=ore.crosstab(MARITAL_STATUS ~ GENDER+CLASS, data=NARROW)

# One way table
ore.crosstab(~AGE, data=NARROW)

# Weight values in AGE and GENDER using values in CLASS
x=ore.crosstab(AGE~GENDER*CLASS, data=NARROW)

# Order rows of cross tab by frequency counts
x=ore.crosstab(AGE~GENDER|FREQ, data=NARROW)
```


ore.crosstab – Results

```
> table(NARROW$MARITAL_STATUS, NARROW$GENDER)

           F   M
  Divorc.  85  56
  Mabsent   6   7
  Mar-AF    1   0
  Married  55 471
  NeverM  147 201
  Separ.   23  18
  Widowed  27   4
```

```
> ore.crosstab(MARITAL_STATUS ~ GENDER, data=NARROW)
   MARITAL_STATUS GENDER ORE$FREQ ORE$STRATA ORE$GROUP
1         Divorc.      F       85          1         1
2         Divorc.      M       56          1         1
3         Mabsent      F        6          1         1
4         Mabsent      M        7          1         1
5          Mar-AF      F        1          1         1
6         Married      F       55          1         1
7         Married      M      471          1         1
8          NeverM      F      147          1         1
9          NeverM      M      201          1         1
10         Separ.      F       23          1         1
11         Separ.      M       18          1         1
12        Widowed      F       27          1         1
13        Widowed      M        4          1         1
```

```
> x=ore.crosstab(MARITAL_STATUS ~ GENDER+CLASS, data=NARROW)
>
> x
$`MARITAL_STATUS~GENDER`
   MARITAL_STATUS GENDER ORE$FREQ ORE$STRATA ORE$GROUP
1         Divorc.      F       85          1         1
2         Divorc.      M       56          1         1
3         Mabsent      F        6          1         1
4         Mabsent      M        7          1         1
5          Mar-AF      F        1          1         1
6         Married      F       55          1         1
7         Married      M      471          1         1
8          NeverM      F      147          1         1
9          NeverM      M      201          1         1
10         Separ.      F       23          1         1
11         Separ.      M       18          1         1
12        Widowed      F       27          1         1
13        Widowed      M        4          1         1

$`MARITAL_STATUS~CLASS`
   MARITAL_STATUS CLASS ORE$FREQ ORE$STRATA ORE$GROUP
1         Divorc.     0      150          1         1
2         Divorc.     1       13          1         1
3         Mabsent     0       17          1         1
4          Mar-AF     0        1          1         1
5         Married     0      336          1         1
6         Married     1      284          1         1
7          NeverM     0      395          1         1
8          NeverM     1       24          1         1
9          Separ.     0       42          1         1
10         Separ.     1        4          1         1
11        Widowed     0       33          1         1
12        Widowed     1        1          1         1
```

ore.crosstab – Examples

```
# 4, 2 way cross tabs (GENDER,AGE,MARITAL_STATUS,COUNTRY)~CLASS
x=ore.crosstab(GENDER-COUNTRY~CLASS, data=NARROW)
length(x)

# 1 way table with as many rows as unique values of COUNTRY for each unique value
of AGE
x = ore.crosstab(~AGE/COUNTRY, data=NARROW)

# Same as above, but 2 way
x = ore.crosstab(AGE~GENDER/COUNTRY, data=NARROW)

# Post-process output - 2 2-way tables AGExGENDER and AGExCLASS
x=ore.crosstab(AGE ~ GENDER+CLASS, data=NARROW)
class(x)
class(x[[1]])
names(x[[1]])
z <- x[[1]][,c(1,2,3)]
```

ore.crosstab – Results

```
> x=ore.crosstab(GENDER-COUNTRY~CLASS, data=NARROW)
> length(x)
[1] 4
>
> head(x)
$`GENDER~CLASS`
  GENDER CLASS ORE$FREQ ORE$STRATA ORE$GROUP
1      F     0      378          1         1
2      F     1       43          1         1
3      M     0      598          1         1
4      M     1      282          1         1

$`AGE~CLASS`
    AGE CLASS ORE$FREQ ORE$STRATA ORE$GROUP
1    17     0       14          1         1
2    18     0       16          1         1
3    19     0       30          1         1
4    20     0       23          1         1
5    21     0       22          1         1
6    22     0       39          1         1
7    23     0       29          1         1
8    24     0       24          1         1
9    25     0       34          1         1
10   25     1        2          1         1
```

```
> x = ore.crosstab(~AGE/COUNTRY, data=NARROW)
>
> x
    AGE ORE$FREQ ORE$STRATA ORE$GROUP
1    17        1          1         1
2    17        1          1         3
3    17       12          1        19
4    18       16          1        19
5    19       30          1        19
6    20       23          1        19
7    21        1          1         1
8    21        1          1         9
9    21       20          1        19
10   22        2          1         1
11   22        1          1         8
12   22       36          1        19
13   23        1          1         3
14   23       28          1        19
15   24        3          1         1
16   24        2          1         9
17   24        1          1        15
18   24        1          1        18
```

```
> x = ore.crosstab(AGE~GENDER/COUNTRY, data=NARROW)
>
> x
    AGE GENDER ORE$FREQ ORE$STRATA ORE$GROUP
1    17      F        4          1        19
2    17      F        1          1         3
3    17      M        8          1        19
4    17      M        1          1         1
5    18      F        6          1        19
6    18      M        7          1        19
7    19      F       15          1        19
8    19      M       13          1        19
9    20      F        9          1        19
10   20      M       13          1        19
11   21      F       10          1        19
12   21      F        1          1         1
13   21      F        1          1         9
14   21      M        6          1        19
15   22      F        1          1         8
16   22      F       15          1        19
17   22      M       19          1        19
18   22      M        1          1         1
19   23      F        9          1        19
20   23      M       18          1        19
21   23      M        1          1         3
22   24      F        1          1         9
23   24      F        1          1         1
24   24      F        1          1        18
25   24      F        6          1        19
26   24      F        1          1        15
```

ore.crosstab – Results

```
> x=ore.crosstab(AGE ~ GENDER+CLASS, data=NARROW)
> class(x)
[1] "list"
> class(x[[1]])
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
> names(x[[1]])
[1] "AGE"        "GENDER"     "ORE$FREQ"   "ORE$STRATA" "ORE$GROUP"
> z <- x[[1]][,c(1,2,3)]
>
> z
    AGE GENDER ORE$FREQ
1    17      F        5
2    17      M        9
3    18      F        6
4    18      M        7
5    19      F       15
6    19      M       13
7    20      F        9
8    20      M       13
9    21      F       12
10   21      M        6
```

```
> x
$`AGE~GENDER`
    AGE GENDER ORE$FREQ ORE$STRATA ORE$GROUP
1    17      F        5          1          1
2    17      M        9          1          1
3    18      F        6          1          1
4    18      M        7          1          1
5    19      F       15          1          1
6    19      M       13          1          1
7    20      F        9          1          1
8    20      M       13          1          1
9    21      F       12          1          1
10   21      M        6          1          1
11   22      F       16          1          1
12   22      M       20          1          1
13   23      F        9          1          1
14   23      M       19          1          1
15   24      F       10          1          1
16   24      M       13          1          1

$`AGE~CLASS`
    AGE CLASS ORE$FREQ ORE$STRATA ORE$GROUP
1    17     0       14          1          1
2    18     0       16          1          1
3    19     0       30          1          1
4    20     0       23          1          1
5    21     0       22          1          1
6    22     0       39          1          1
7    23     0       29          1          1
8    24     0       24          1          1
9    25     0       34          1          1
10   25     1        2          1          1
11   26     0       17          1          1
12   26     1        5          1          1
13   27     0       42          1          1
```

ore.crosstab – Examples

```
# Stratification - As many 2 way tables as unique values of CLASS
x <- ore.crosstab(AGE~GENDER^CLASS, data=NARROW)

# Custom binning and subsequent cross tabulation
NARROW$AGEBINS=ifelse(NARROW$AGE<20, 1,
                ifelse(NARROW$AGE<30,2,
                ifelse(NARROW$AGE<40,3,4)))
ore.crosstab(GENDER~AGEBINS, NARROW)
```

ore.crosstab – Results

```
> x <- ore.crosstab(AGE~GENDER^CLASS, data=NARROW)
>
> x
    AGE GENDER ORE$FREQ ORE$STRATA ORE$GROUP
1    17      F        5           1         1
2    17      M        9           1         1
3    18      F        6           1         1
4    18      M        7           1         1
5    19      F       15           1         1
6    19      M       13           1         1

114  77      M        1           1         1
115  80      M        2           1         1
116  82      M        1           1         1
117  90      F        1           1         1
118  90      M        2           1         1
119  25      F        1           2         1
120  25      M        1           2         1
121  26      F        2           2         1
122  26      M        2           2         1
123  27      M        3           2         1
```

```
> NARROW$AGEBINS=ifelse(NARROW$AGE<20, 1,
+                 ifelse(NARROW$AGE<30,2,
+                 ifelse(NARROW$AGE<40,3,4)))
> ore.crosstab(GENDER~AGEBINS, NARROW)
  GENDER AGEBINS ORE$FREQ ORE$STRATA ORE$GROUP
1      F       1       26           1         1
2      F       2      108           1         1
3      F       3       86           1         1
4      F       4      164           1         1
5      M       1       29           1         1
6      M       2      177           1         1
7      M       3      230           1         1
8      M       4      381           1         1
```

ore.freq

Operates on output of ore.crosstab() and automatically determines the techniques relevant to the nature of the result

1-way cross tables

- Goodness of fit tests for equal proportions or specified null proportions, confidence limits, and tests for equivalence

2-way cross tables

- Various statistics that describe relationships between columns in the cross tabulation
- Chi-square tests, cochran-mantel-haenzsel statistics, measures of association, strength of association, risk differences, odds ratio and relative risk for 2x2 tables, tests for trend

N-way cross tables

- N 2-way cross tables
- Statistics across and within strata

Leverages database SQL functions when available

ore.freq - Parameters

Returns an OML4R data frame as output in all cases

x, crosstab: ore.frame output from ore.crosstab()

stats: List of statistics required

- ChiSquare: AJCHI, LRCHI, MHCHI, PCHISQ -- Kappa:KAPPA, WTKAP,
- Lambda:LAMCR, LAMRC, LAMDAS -- Correlation:KENTB,PCORR, SCORR
- Stuart'c Tau, Somer's D|C:STUTC,SMDCR,SMDRC -- Fisher's, Cochran's Q:FISHER, COCHQ
- Odds Ratio:OR, MHOR, LGOR -- Relative Risk:RR,MHRR,ALRR
- Others: MCNEM, PHI, CRAMV, CONTGY, TSYM, TREND, GAMMA,

params: Control parameters to the specific statistical function

- SCORE: TABLE|RANK|RIDIT|MODRIDIT
- ALPHA: <number>
- WEIGHTS: <number>

skip.failed: (TRUE/FALSE) if a statistical test required fails on the cross table because it is found to be in-applicable to the table then return immediately

skip.missing: (TRUE/FALSE) skip cells with missing values in the cross table

ore.freq – Examples

Compute cross tabulation for number of diverted flights for each airline. Compute the Pearson CHISQ for the results.

```
ct <- ore.crosstab(UNIQUECARRIER~DIVERTED, data=ONTIME_S)
ct
freq <- ore.freq(ct)
freq
```

For each airline, compute cross tabulation for number of diverted flights and day of week. Compute the Pearson CHISQ for each result.

```
ct <-
ore.crosstab(UNIQUECARRIER~DIVERTED+DAYOFWEEK,data=ONTIME_S)
ct
freq <- ore.freq(ct)
freq
```

```
R> ct <- ore.crosstab(UNIQUECARRIER~DIVERTED, data=ONTIME_S)
R> ct
  UNIQUECARRIER DIVERTED ORE$FREQ ORE$STRATA ORE$GROUP
0            9E        0      741          1         1
1            9E        1        6          1         1
2            AA        0    27564          1         1
3            AA        1       91          1         1
4            AQ        0      234          1         1
5            AS        0     5159          1         1
```

```
R> freq <- ore.freq(ct)
R> freq
  METHOD     FREQ DF      PVALUE              DESCR GROUP
0 PCHISQ 54.58397 28 0.001907040 Pearson Chi-Square     1
```

```
R> ct <- ore.crosstab(UNIQUECARRIER~DIVERTED+DAYOFWEEK,data=ONTIME_S)
R> ct
$`UNIQUECARRIER~DIVERTED`
  UNIQUECARRIER DIVERTED ORE$FREQ ORE$STRATA ORE$GROUP
0            9E        0      741          1         1
1            9E        1        6          1         1
4            AA        0    27564          1         1
5            AA        1       91          1         1

$`UNIQUECARRIER~DAYOFWEEK`
   UNIQUECARRIER DAYOFWEEK ORE$FREQ ORE$STRATA ORE$GROUP
0             9E         1      117          1         1
1             9E         2      115          1         1
2             9E         3      113          1         1
3             9E         4      120          1         1
4             9E         5       99          1         1
```

```
R> freq <- ore.freq(ct)
R> freq
$`UNIQUECARRIER~DIVERTED`
  METHOD     FREQ DF      PVALUE              DESCR GROUP
0 PCHISQ 54.58397 28 0.001907040 Pearson Chi-Square     1

$`UNIQUECARRIER~DAYOFWEEK`
  METHOD     FREQ  DF       PVALUE              DESCR GROUP
0 PCHISQ 260.6275 168 6.014678e-06 Pearson Chi-Square     1
```

ore.corr

Correlation analysis across numeric columns in an ore.frame

Supports partial correlations with a control column

Enables aggregations prior to correlations

Allows post-processing of results and integration into R code flow

Output can be made to conform to output of R's cor() function and so can be post-processed by any CRAN function or graphics

ore.corr – Parameters

Returns an ore.frame as output in all cases except when group.by is used in which case an ore.list object is returned

data: ore.frame of the data to compute correlation coefficients

var: numeric column(s) of the *data* for which to build correlation matrix

stats: pearson (default), spearman, kendall

group.by: as many correlation matrices as unique values in group.by columns

freq: character string specifying a numeric column within argument *data* to use as a frequency count

with: character vector specifying the numeric columns in argument *data* to pair with columns specified in argument *var*

weight: column of *data* whose numeric values provide a multiplicative factor for *var columns*

partial: columns of data to use as control variables for partial correlation

Use OREeda:::ore.corr.as.matrix() to convert into R's cor() compatible output format

ore.corr – Examples

```
# R's cor –project out non-numeric columns first
names(NARROW)
N <- ore.pull(NARROW[,c(3,8,9)])
cor(N, use="complete.obs")
cor(N, method='spearman' , use="complete.obs")
cor(N, method='kendall' , use="complete.obs")

# Corresponding ore.corr
x1 <- ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS')
x2 <- ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='spearman')
x3 <- ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='kendall')
cor_compatible_matrix = OREeda:::.ore.corr.as.matrix(x3)
class(cor_compatible_matrix)
```

ore.corr - Results

```
> names(NARROW)
[1] "ID"             "GENDER"         "AGE"            "MARITAL_STATUS" "COUNTRY"
[6] "EDUCATION"      "OCCUPATION"     "YRS_RESIDENCE"  "CLASS"
> N <- ore.pull(NARROW[,c(3,8,9)])
> cor(N, use="complete.obs")
                    AGE YRS_RESIDENCE     CLASS
AGE           1.0000000     0.6568534 0.2206276
YRS_RESIDENCE 0.6568534     1.0000000 0.3362991
CLASS         0.2206276     0.3362991 1.0000000
> cor(N, method='spearman' , use="complete.obs")
                    AGE YRS_RESIDENCE     CLASS
AGE           1.0000000     0.7477093 0.2632748
YRS_RESIDENCE 0.7477093     1.0000000 0.3646760
CLASS         0.2632748     0.3646760 1.0000000
> cor(N, method='kendall' , use="complete.obs")
                    AGE YRS_RESIDENCE     CLASS
AGE           1.0000000     0.6332196 0.2173413
YRS_RESIDENCE 0.6332196     1.0000000 0.3200559
CLASS         0.2173413     0.3200559 1.0000000
```

```
> x1 <- ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS')
> x1
            ROW           COL PEARSON_T PEARSON_P PEARSON_DF
1           AGE YRS_RESIDENCE 0.6568534     0e+00       1098
2           AGE         CLASS 0.2200960     1e-15       1298
3 YRS_RESIDENCE         CLASS 0.3561869     0e+00       1298
> x2 <- ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='spearman')
> x2
            ROW           COL SPEARMAN_T SPEARMAN_P SPEARMAN_DF
1           AGE YRS_RESIDENCE  0.7462684      0e+00        1098
2           AGE         CLASS  0.2601221      1e-15        1298
3 YRS_RESIDENCE         CLASS  0.3835252      0e+00        1298
> x3 <- ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='kendall')
> x3
            ROW           COL KENDALL_T    KENDALL_P KENDALL_DF
1           AGE YRS_RESIDENCE 0.6332196 0.000000e+00       <NA>
2           AGE         CLASS 0.2147107 4.285594e-31       <NA>
3 YRS_RESIDENCE         CLASS 0.3362078 1.094478e-73       <NA>
> cor_compatible_matrix = OREeda:::.ore.corr.as.matrix(x3)
> class(cor_compatible_matrix)
[1] "matrix"
```

ore.corr – Examples

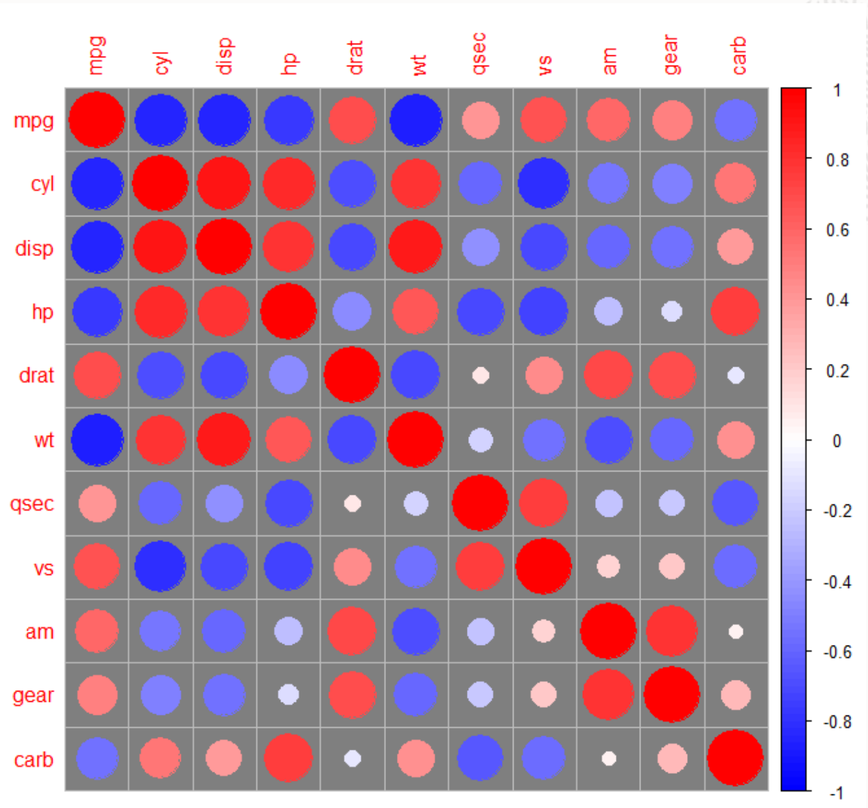
```
#Partial correlation
ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='spearman', partial='GENDER')

#Creating a number of correlation matrices
x <- ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS',
              stats='spearman', partial='GENDER', group.by='COUNTRY')
class(x)
cor_compatible_matrix <- OREeda:::.ore.corr.as.matrix(x[[1]])
```

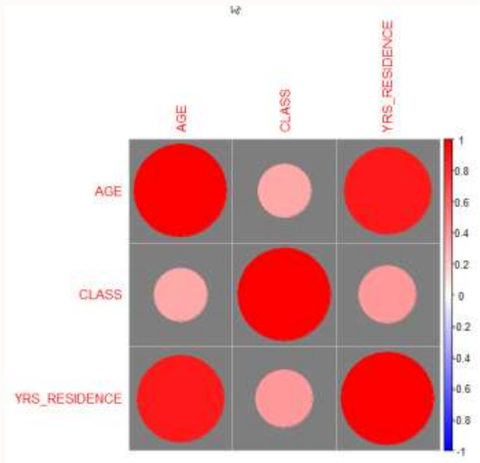

ore.corr

Post-processing matrix using CRAN visualization

```
library(corrplot)
corrplot( cor(mtcars), order = "original", bg =
"gray50",
   col = colorRampPalette(c("blue","white","red"))(100))
corrplot(cor_compatible_matrix,
  order = "original", bg = "gray50",
  col = colorRampPalette(c("blue","white","red"))(100))
```

http://addictedtor.free.fr/graphiques/RGraphGallery.php?graph=152





ore.univariate

Enables distribution analysis of numeric variables in an ore.frame

Statistics

- All statistics reported by ore.summary()
- Signed rank test, Student's t-test
- Extreme values reporting

Graphics

- QQ plots
- Scatterplots

ore.univariate – Parameters

Returns an ore.frame as output in all cases except when group.by is used in which case an R list object is returned

data: ore.frame of the data whose columns are to be analyzed

var: numeric column(s) of the *data* for which to compute statistics

weight: A column of the *data* whose numeric values provide a multiplicative factor for *var columns*

stats: optional specification of a subset of statistics to be printed
- momemts – n,sumwgt,mean,sum,stddev,var,skew,kurt.,uss.css.cv.stderr
- measures – mean,stddev,median,var,mode,range,iqr
- quantiles – p100,p99,p95,p90,p75,p50,p25,p10,p5,p1,p0
- location – studentt,studentp,signt,signp,srankt,srankp
- normality
- loccount – loc<,loc>,loc!
- extremes

ore.univariate - Examples

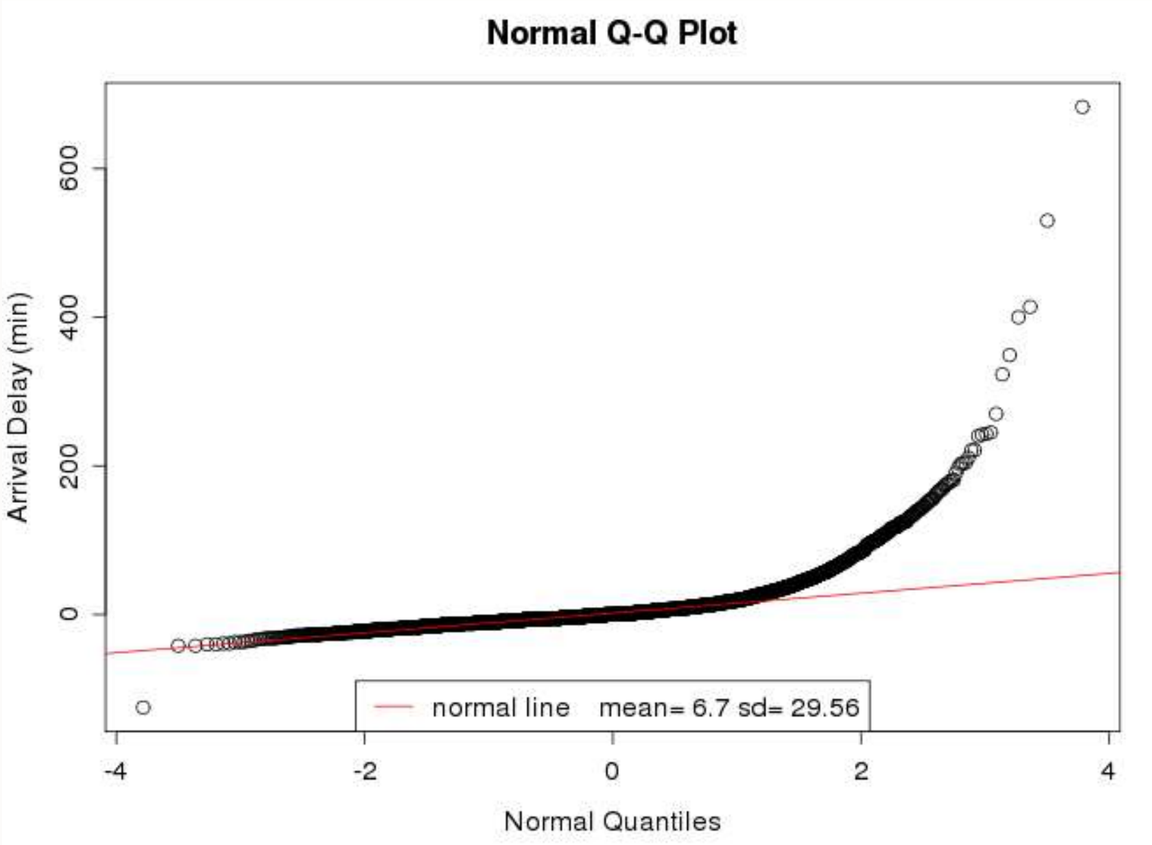
```
# Default univariate statistics
ore.univariate(NARROW, var="AGE,YRS_RESIDENCE,CLASS")

# Compute location statistics on YRS_RESIDENCE
# (Student's t-test and two-tailed p-value for student's t-test)
ore.univariate(NARROW, var="YRS_RESIDENCE",stats="location")

# Compute quantiles statistics on AGE and YRS_RESIDENCE
ore.univariate(NARROW, var="AGE,YRS_RESIDENCE",stats="quantiles")
```

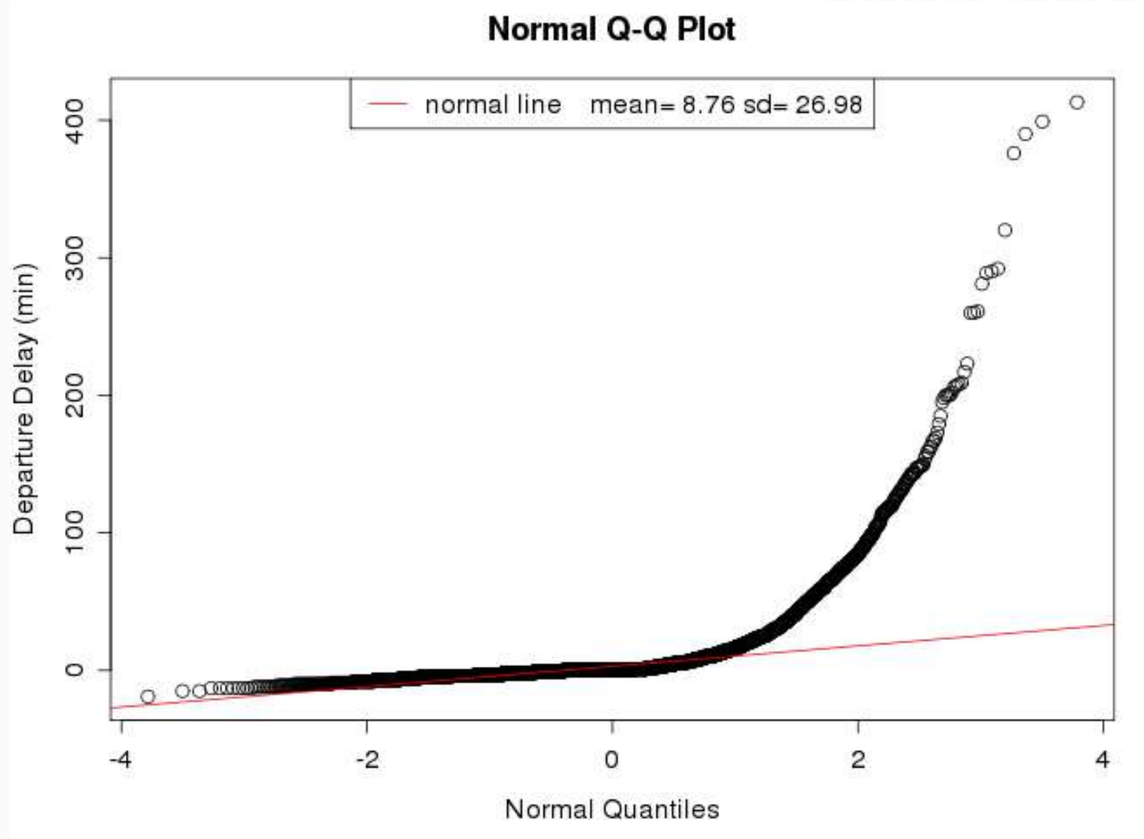

ore.univariate – Results

```
R> # Default univariate statistics
R> ore.univariate(NARROW, var="AGE,YRS_RESIDENCE,CLASS")
  N(AGE) N(YRS_RESIDENCE) N(CLASS) SUMWGT(AGE) SUMWGT(YRS_RESIDENCE) SUMWGT(CLASS) MEAN(AGE) MEAN(YRS_RESIDENCE) MEAN(CLASS)
1   1300             1300     1500        1300                 1300          1500  39.12769            4.081538   0.2533333
  SUM(AGE) SUM(YRS_RESIDENCE) SUM(CLASS) STDDEV(AGE) STDDEV(YRS_RESIDENCE) STDDEV(CLASS) VAR(AGE) VAR(YRS_RESIDENCE) VAR(CLASS)
1    50866               5306        380    13.69127              1.932932     0.4350652  187.451           3.736225  0.1892817
  USS(AGE) USS(YRS_RESIDENCE) USS(CLASS) CSS(AGE) CSS(YRS_RESIDENCE) CSS(CLASS)  CV(AGE) CV(YRS_RESIDENCE) CV(CLASS) STDERR(AGE)
1  2233768              26510        380 243498.8           4853.357   283.7333 34.99126          47.35792  171.7363   0.3797276
  STDERR(YRS_RESIDENCE) STDERR(CLASS) SKEW(AGE) SKEW(YRS_RESIDENCE) SKEW(CLASS)  KURT(AGE) KURT(YRS_RESIDENCE) KURT(CLASS)
1            0.05360988    0.01123334 0.6001366            0.821637    1.135444 0.01924574            1.739197  -0.7117187
R>
R> # Compute location statistics on YRS_RESIDENCE
R> ore.univariate(NARROW, var="YRS_RESIDENCE",stats="location")
         T PRT
1 76.13407   0
R>
R> # Compute quantiles statistics on AGE and YRS_RESIDENCE
R> ore.univariate(NARROW, var="AGE,YRS_RESIDENCE",stats="quantiles")
  P100(AGE) P100(YRS_RESIDENCE) P99(AGE) P99(YRS_RESIDENCE) P95(AGE) P95(YRS_RESIDENCE) P90(AGE) P90(YRS_RESIDENCE) P75(AGE)
1        90                  14       75                  10       64                  7     58.1                  7       48
  P75(YRS_RESIDENCE) P50(AGE) P50(YRS_RESIDENCE) P25(AGE) P25(YRS_RESIDENCE) P10(AGE) P10(YRS_RESIDENCE) P5(AGE) P5(YRS_RESIDENCE)
1                  5       38                  4       28                  3       22                  2      20                  1
  P1(AGE) P1(YRS_RESIDENCE) P0(AGE) P0(YRS_RESIDENCE)
1      17                 0      17                 0
```

Normal QQ Plot – Univariate Graphics

Arrival and Departure Delay





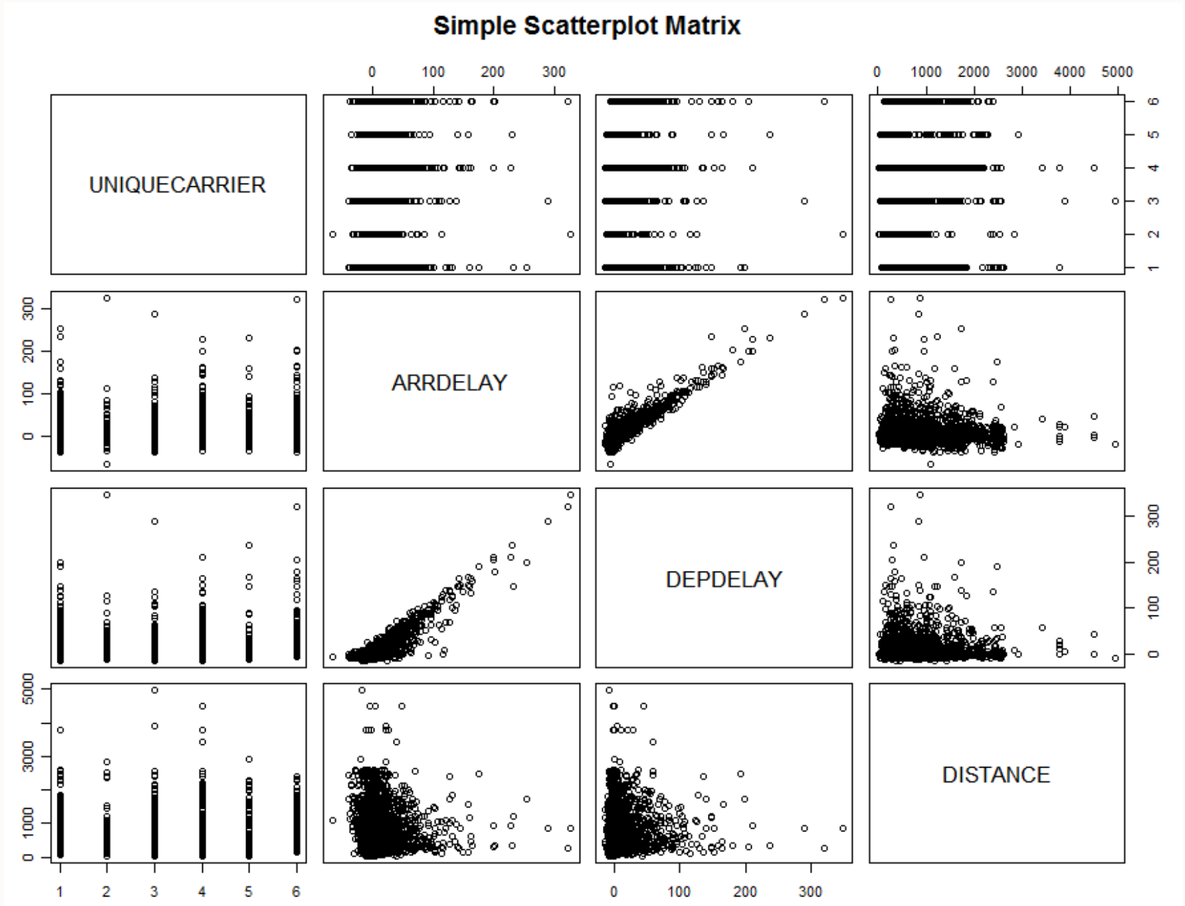
Normal QQ Plot – Univariate Graphics

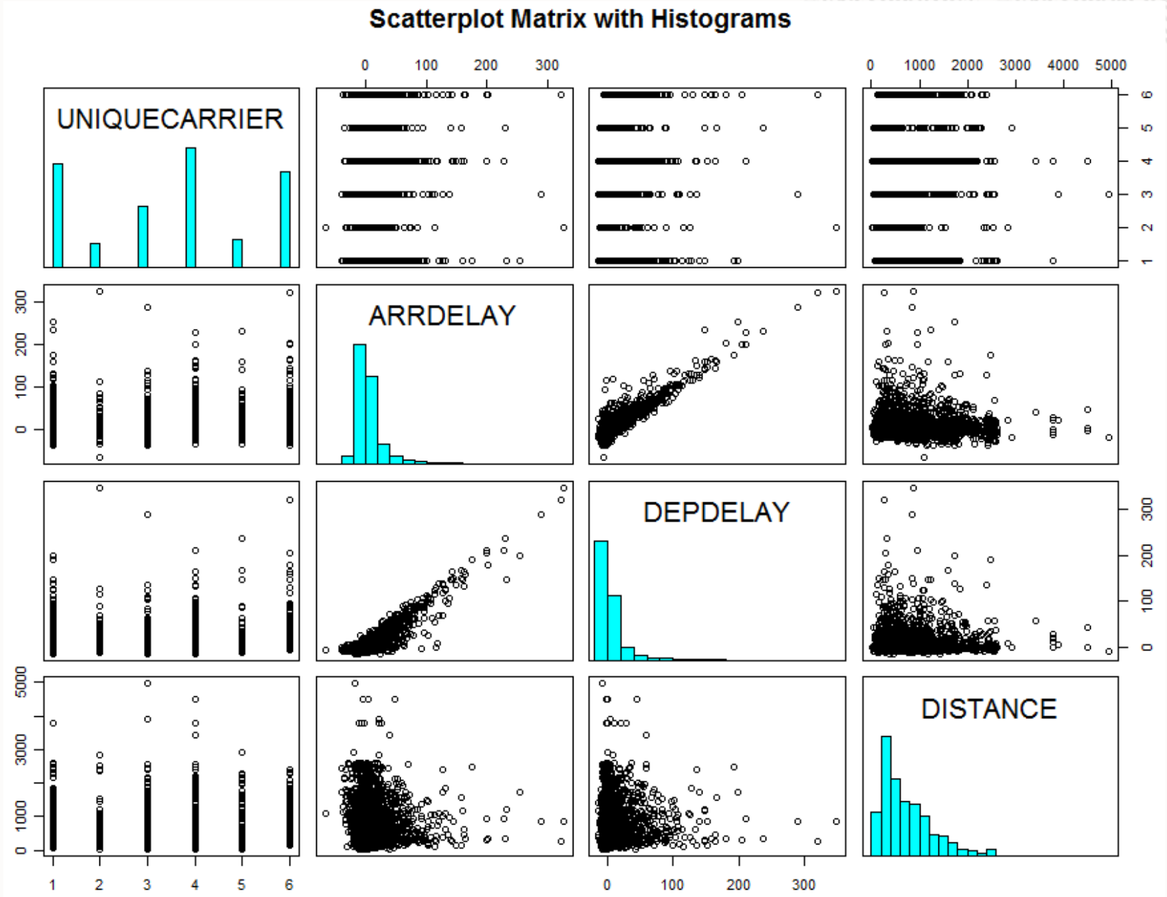
Arrival and Departure Delay

```
ontime <- ONTIME_S
ontimeSubset <- ontime[ontime$UNIQUECARRIER =="AA", ,drop = TRUE]
ontimeSubset <- ontime[, c("ARRDELAY", "DEPDELAY")]
ontimeSubset <- ore.pull(ontimeSubset)
ontimeSubset <-ontimeSubset[sample(nrow(ontimeSubset),nrow(ontimeSubset)*.03),]

dat  = ontimeSubset$ARRDELAY
type = "Arrival"
#dat  = ontimeSubset$DEPDELAY
#type = "Departure"

qqnorm(dat, xlab="Normal Quantiles", ylab=paste(type, " Delay (min)",sep="" ));
qqline(dat, col = 2)

mean <- mean(dat, na.rm=TRUE)
sd <- sd(dat, na.rm=TRUE)

legend("top", paste("normal line    mean= ",round(mean,2),
                    " sd= ",round(sd,2),sep=""), lty=1, col="red")
```

Scatterplot Matrix

Airline, Arrival Delay, Departure Delay, Distance





Scatterplot Matrix

Airline, Arrival Delay, Departure Delay, Distance

```
ontime <- ONTIME_S
ontimeSubset <- ontime[ontime$UNIQUECARRIER %in% c("AA", "AS", "CO", "DL","HP","WN"), ,
          drop = TRUE]
ontimeSubset <- ore.pull(ontimeSubset)
ontimeSubset <-ontimeSubset[sample(nrow(ontimeSubset),nrow(ontimeSubset)*.03),]

pairs(~UNIQUECARRIER+ARRDELAY+DEPDELAY+DISTANCE,data=ontimeSubset,
   main="Simple Scatterplot Matrix")
```

```
os2 <- with(ontimeSubset, data.frame(UNIQUECARRIER, ARRDELAY, DEPDELAY, DISTANCE))

panel.hist <- function(x, ...)  {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5) )
  h <- hist(x, breaks=20, plot = FALSE)
  breaks <- h$breaks; nB <- length(breaks)
  y <- h$counts; y <- y/max(y)
  rect(breaks[-nB], 0, breaks[-1], y, col="cyan", ...)
}
pairs(os2, diag.panel=panel.hist, main="Scatterplot Matrix with Histograms")
```

For more information…

oracle.com/machine-learning

See also AskTOM OML Office Hours

Thank You

Mark Hornick
Oracle Machine Learning Product Management