useR! 2006

Data Profiling with R

Discovering Data Quality Issues as Early as Possible

Loyalty Matrix

June 2006

Loyalty Matrix, Inc.
580 Market Street
Suite 600
San Francisco, CA 94104
(415) 296-1141
http://www.loyaltymatrix.com

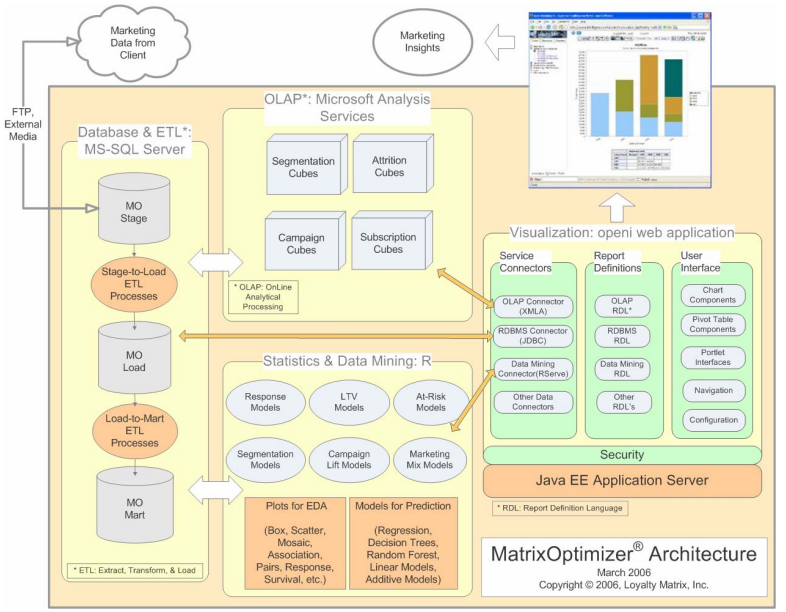
Agenda

- Background & Problem Statement
- High Level Design
- R & SQL Integration Examples
- Grid Graphics for Summary Panels
- Some Real World Data Profiles
- Demo Profile Run (After Break)

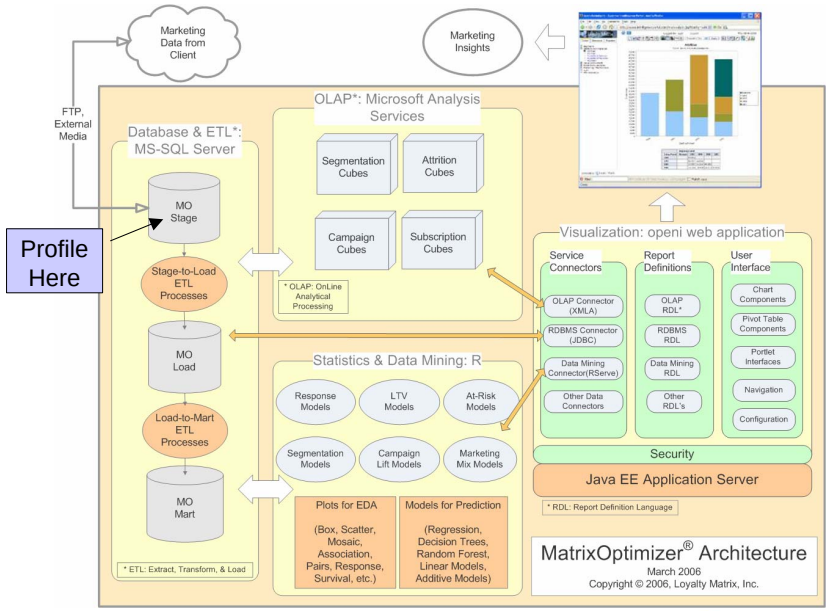
Loyalty Matrix Background

- 15-person San Francisco firm with an offshore team in Nepal
- Provide customer data analytics to optimize direct marketing resources
- OnDemand platform MatrixOptimizer® (version 3.1)
- Over 20 engagements with Fortune 500 clients
- Deliver actionable marketing actions based on real customer behavior

MatrixOptimizer®: Architecture Overview



MatrixOptimizer®: Profile Point

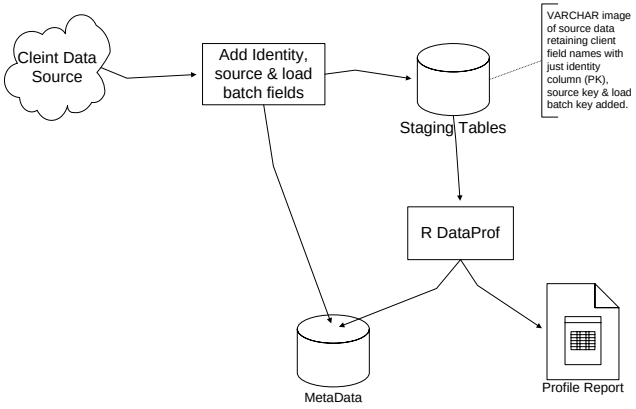


Ideal Data Profiler Requirements

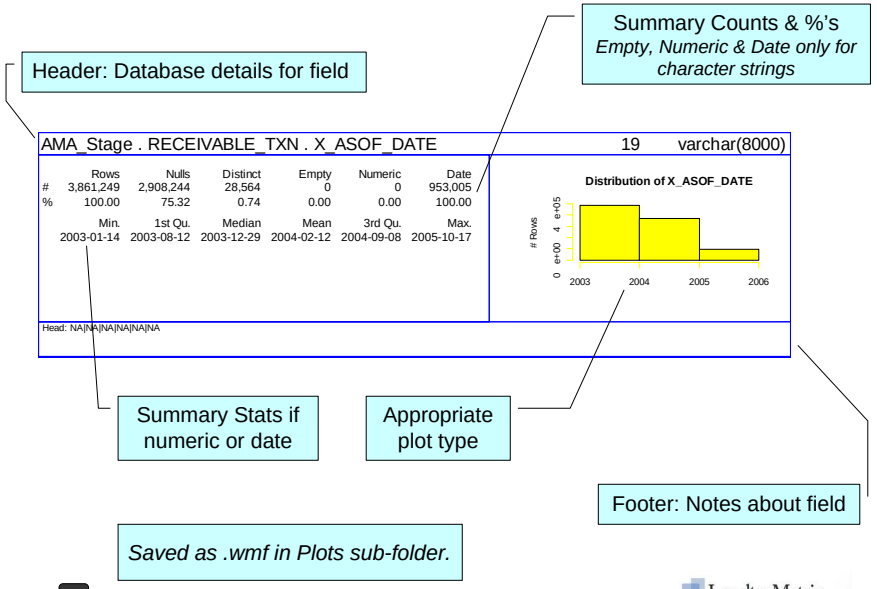
- Require minimum input from analyst to run
- Intuitive output for DB pros – share results with client DBA.
- Column profiling (each column treated independently)
  - Simple statistics & plots
  - Patterns, exceptions & common domain detection
- Dependency profiling for intra-table dependencies
- Redundancy profiling for between table keys, overlaps
- Easy to use reports for analysts & clients
- Save findings in accessible data structure for subsequent use
- Low Cost or "Free"

- See: *Data Quality – The Accuracy Dimension* by Jack E. Olson

Profiling Data Flow



Profiler Output Panel



High Level Design

- User picks database to profile
- User selects specific tables or <all>
- For all selected tables:
  - For all columns in table:
    - Get basic stats
    - Select most appropriate plot type
      - Get data for plot
    - Write panel text & plot to .wmf

R & SQL Integration (1)

- Let SQL do the heavy lifting & minimize data sent to R
- RODBC also reads tables & columns:

```
odbcTables(cODBC)
lTables <- odbcFetchRows(cODBC)   ## fetch the list of tables
lTableNames <- lTables$data[[3]][lTables$data[[4]] == "TABLE" &
                                 lTables$data[[3]] != "dtproperties"]
```

```
odbcColumns(cODBC, ithTable)
lColumns <- odbcFetchRows(cODBC)
Columns <- data.frame(Name = lColumns$data[[4]],
                      Type = lColumns$data[[6]],
                      Width = lColumns$data[[8]])
```

- Get Number of Nulls

```
NumNull <- as.integer(sqlQuery(cODBC,
                      paste("SELECT COUNT(*) FROM ", ithTable,
                            " WHERE [", ColName, "] IS NULL",
                            sep = "")))
```


R & SQL Integration (2)

- Get Number of Distincts

```
NumDistinct <- as.integer(sqlQuery(cODBC,
                          paste("SELECT COUNT(DISTINCT[",
                                ColName, "]) FROM ", ithTable,
                                sep = "")))
```

- Get Number of Empties

```
NumEmpty <- as.integer(sqlQuery(cODBC,
                       paste("SELECT COUNT(*) FROM ", ithTable,
                             " WHERE LEN(LTRIM(RTRIM([",
                             ColName, "])) = 0", sep = "")))
```

- Get Number of Numerics

```
NumNumeric <- as.integer(sqlQuery(cODBC,
                         paste("SELECT SUM(ISNUMERIC([", ColName,
                         "])) FROM ", ithTable, sep = "")))
```

- Get Number of Dates

```
NumDate <- as.integer(sqlQuery(cODBC,
                      paste("SELECT SUM(ISDATE([", ColName,
                      "])) FROM ", ithTable, sep = "")))
```


R & SQL Integration (3)

- Look for reasonable plot type & pull plot data:

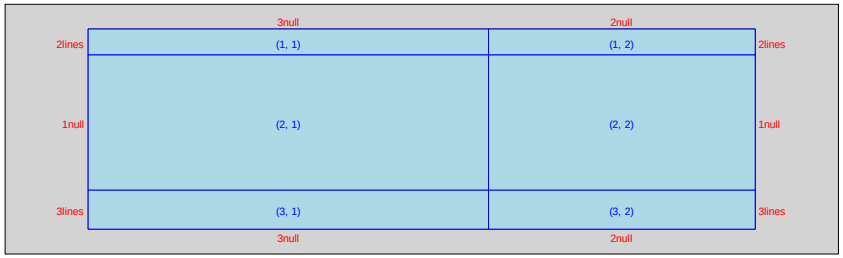
```
## Come up with a reasonable plot type based on data types, coverage, etc.
    PlotType <- ""

    ## when NumDistinct a small number of categories
    if (PlotType == "" & NumDistinct <= 10) {
      PlotType <- "Category"
      PlotValues <- sqlQuery(cODBC, paste("SELECT [", ColName,
                    "] ColValue, COUNT(*) NumRows FROM ", ithTable,
                    " GROUP BY [", ColName, "] ORDER BY COUNT(*)",
                    sep = ""))
    }
```

- And so on for
  - Numbers or strings that mostly look like numbers
  - Dates or strings that mostly look like dates
  - Large number of categories
- Setting up for plot function
  - PlotType
  - PlotValues

Grid Graphics Tricks (1)

- Set up panel

```
windows(width = 10.5, height = 3, pointsize = 10)
TopLayout <- grid.layout(nrow = 3, ncol = 2,
                         widths = unit(c(3, 2), c("null", "null")),
                         heights = unit(c(2, 1, 3),
                         c("lines", "null", "lines")))
#grid.show.layout(TopLayout)       ## <<<<<<Debug only
```



Grid Graphics Tricks (2)

- Walk through the viewports starting with header:

```
pushViewport(vpTopLayout)
 grid.rect(gp = gpar(col = "blue", lwd = 3))

 pushViewport(viewport(layout.pos.col = 1:2, layout.pos.row = 1))
   grid.rect(gp = gpar(col = "blue", lwd = 2))
   grid.text(paste(ColDesc$DB, ColDesc$Table,
                   ColDesc$Column, sep = " . "),
             x = unit(0.2, "char"), y = unit(0.6, "lines"),
             just = "left", gp = gpar(col="black", fontsize=18))
   grid.text(ColDesc$ColSeqNum,
             x = unit(0.8, "npc"), y = unit(0.6, "lines"),
             just = "right",gp = gpar(col="black", fontsize=18))
   grid.text(paste(ColDesc$ColType, "(", ColDesc$ColWidth,
                   ") ", sep = ""),
             x = unit(1, "npc"), y = unit(0.6, "lines"),
             just = "right", gp=gpar(col="black", fontsize=18))
 popViewport()
```

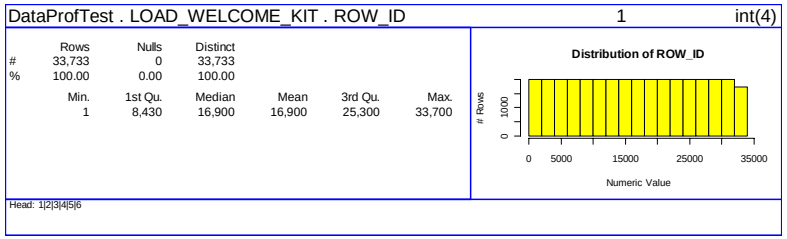

Grid Graphics Tricks (3)

- Only tricky bit is allowing base graphics to do it's thing in plot area:

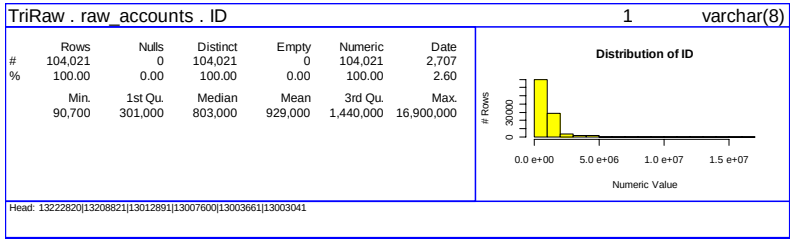
```
## Plot
  pushViewport(viewport(layout.pos.col = 2, layout.pos.row = 2))
    op <- par(no.readonly = TRUE)  ## around all of plot options below
    par(fig = gridFIG(), new = TRUE)
    par(mfg = c(1, 1))
    # a Category plot
    if (ColDesc$ColPlot == "Category") {
      par(mar = c(4.5, 10, 1.8, 2) + 0.1)
      pPlot <-barplot(PlotValues$NumRows,
                      names.arg = as.character(PlotValues$ColValue),
                      horiz = TRUE, las = 1, col = "yellow",
                      main = paste("Categories in", ColDesc$Column),
                      ylab = NULL, xlab = "# Rows")
    }

    # etc for NumHist, CharHist, …
```


Concluding Examples of Profiler Runs (1)

- A Surrogate Key



- Probable Business Key

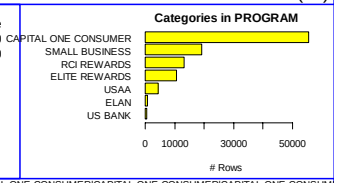


Concluding Examples of Profiler Runs (2)

- A Few Categories

TriRaw . raw_accounts . PROGRAM 3 varchar(20)

	Rows	Nulls	Distinct	Empty	Numeric	Date
#	104,021	0	7	0	0	0
%	100.00	0.00	0.007	0.00	0.00	0.00

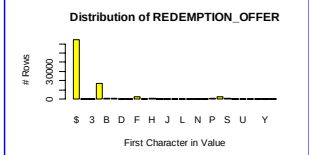


Head: CAPITAL ONE CONSUMER|CAPITAL ONE CONSUMER|CAPITAL ONE CONSUMER|CAPITAL ONE CONSUMER|CAPITAL ONE CONSUMER|CAPITAL ONE CONSUM

- Many Categories

TriRaw . raw_redemptions . REDEMPTION_OFFER 3 varchar(30)

	Rows	Nulls	Distinct	Empty	Numeric	Date
#	90,651	0	866	0	0	0
%	100.00	0.00	0.955	0.00	0.00	0.00



Head: Weber Portable Gas Grill|Weber Portable Gas Grill|Weber Portable Gas Grill|Sony AM/FM Sport Walkman|Sony AM/FM Sport Walkman|Sony AM/FM Sport Walkman

Loyalty Matrix

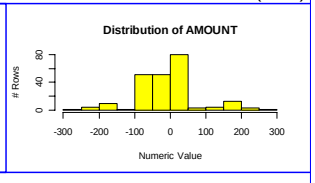
Concluding Examples of Profiler Runs (3)

- Numeric Value

AMA_Stage . RECEIVABLE_TXN_DETAIL . AMOUNT 3 varchar(8000)

	Rows	Nulls	Distinct	Empty	Numeric	Date
#	8,103,330	0	18,600	0	8,103,330	3,820
%	100.00	0.00	0.23	0.00	100.00	0.047

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-295	-50	-25	0	50	295



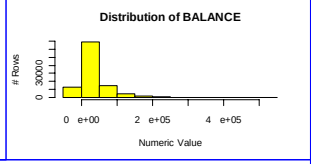
Head: 30|-30|247|-247|30|-30

- Another Numeric Value

TriRaw . raw_accounts . BALANCE 5 varchar(6)

	Rows	Nulls	Distinct	Empty	Numeric	Date
#	104,021	0	53,663	0	104,021	21,118
%	100.00	0.00	51.59	0.00	100.00	20.30

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-47,700	4,160	17,300	31,600	43,100	516,000



Head: 6416|5116|10195|10000|10000|10000

Loyalty Matrix

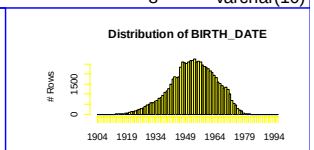
Concluding Examples of Profiler Runs (4)

- Reasonable Dates

TriRaw . raw_accounts . BIRTH_DATE 8 varchar(10)

	Rows	Nulls	Distinct	Empty	Numeric	Date
#	104,021	25,695	17,750	0	0	78,326
%	100.00	24.70	17.06	0.00	0.00	100.00

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1904-07-02	1945-07-06	1953-04-12	1952-10-29	1961-03-08	1995-01-29



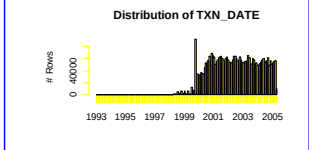
Head: 3/30/1954|3/17/1969|4/5/1971|5/11/1939|4/14/1952|2/10/1960

- Unusual Dates

AMA_Stage . RECEIVABLE_TXN . TXN_DATE 8 varchar(8000)

	Rows	Nulls	Distinct	Empty	Numeric	Date
#	3,861,249	0	1,773	0	0	3,861,190
%	100.00	0.00	0.046	0.00	0.00	100.00

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1993-01-29	2001-07-31	2002-11-30	2002-12-02	2004-04-30	2005-10-17



Head: 1/27/2000 0:00:00|1/27/2000 0:00:00|1/27/2000 0:00:00|1/27/2000 0:00:00|1/27/2000 0:00:00|1/27/2000 0:00:00|1/27/2000 0:00:00

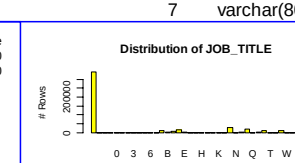
Loyalty Matrix

Concluding Examples of Profiler Runs (5)

- Customer Job Title not too useful

AMA_Stage . CUSTOMER . JOB_TITLE 7 varchar(8000)

	Rows	Nulls	Distinct	Empty	Numeric	Date
#	471,400	340,240	34,242	127	5	0
%	100.00	72.18	7.26	0.027	0.004	0.00

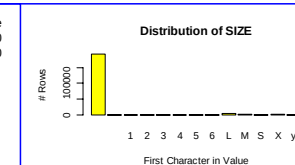


Head: NA|NA|NA|NA|NA|NA

- T-Shirt Size also not reliable

OliviaMatrix . RS_CONTACT_INFO . SIZE 11 varchar(16)

	Rows	Nulls	Distinct	Empty	Numeric	Date
#	204,051	192,277	29	1	0	0
%	100.00	94.23	0.014	0.00	0.00	0.00



Head: NA|NA|Large|NA|XXL|NA

Loyalty Matrix

Conclusion

- Even Current Version Useful in Production
  - Just column details picks up data quality problems
- Next To Do
  - Add DBMS interface layer & support MySQL, etc.
  - Enumerate patterns like 99999-9999, 9* A* A*, etc.
  - Add Hints for common domains
  - Metadata back to database
  - Dependency
  - Redundancy

Questions? Comments?

- Email JPorzak@loyaltymatrix.com
- Call 415-296-1141
- Visit http://www.loyaltymatrix.com
- Come by at:
  580 Market Street, Suite 600
  San Francisco, CA 94104