Data Profiling with R

*Discovering Data Quality Issues as Early as Possible*

---

**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**

---

**Los Angeles Times**

---

---
**MatrixOptimizer®: Profile Point**

- **Profile Here**

**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**

- **Client Data Source**
- **Add Identity, source & load batch fields**
- **Staging Tables**
- **R DataProf**

**Profiler Output Panel**

- **Header: Database details for field**

```
AMA_Stage . RECEIVABLE TXN . X_ASOF_DATE 19 varchar(8000)
```

- **Summary Counts & %’s**
  - Empty, Numeric & Date only for character strings
  - Summary Stats if numeric or date
  - Appropriate plot type
  - Saved as .wmf in Plots sub-folder.

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

R & SQL Integration (2)

- Get Number of Distincts
  ```r
  NumDistinct <- as.integer(sqlQuery(cODBC,
    paste("SELECT COUNT(DISTINCT[",
      ColName, "] FROM ", ithTable,
    " GROUP BY ", sep = ")
  )
  ```

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

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

- Get Number of Dates
  ```r
  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:
  ```r
  ## 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

```r
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:

```r
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:

```r
## 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

<p>| DataProfTest . LOAD_WELCOME_KIT . ROW_ID | 1 | int(4) |</p>
<table>
<thead>
<tr>
<th>%</th>
<th>Rows</th>
<th>Nulls</th>
<th>Distinct</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Min</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>8,430</td>
<td>16,900</td>
<td>16,900</td>
</tr>
<tr>
<td>Median</td>
<td>16,900</td>
<td>25,300</td>
<td>33,700</td>
</tr>
<tr>
<td>Mean</td>
<td>16,900</td>
<td>25,300</td>
<td>33,700</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>25,300</td>
<td>33,700</td>
<td>33,700</td>
</tr>
<tr>
<td>Max</td>
<td>33,700</td>
<td>33,700</td>
<td>33,700</td>
</tr>
</tbody>
</table>

- Probable Business Key

<p>| TriRaw . raw_accounts . ID | 1 | varchar(8) |</p>
<table>
<thead>
<tr>
<th>%</th>
<th>Rows</th>
<th>Nulls</th>
<th>Distinct</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0</td>
<td>104,021</td>
<td>2,707</td>
</tr>
<tr>
<td>Min</td>
<td>90,700</td>
<td>0</td>
<td>83,000</td>
<td>0</td>
<td>929,000</td>
<td>1,440,000</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>301,000</td>
<td>0</td>
<td>16,900</td>
<td>0</td>
<td>16,900</td>
<td>16,900</td>
</tr>
<tr>
<td>Median</td>
<td>1,440,000</td>
<td>0</td>
<td>16,900</td>
<td>0</td>
<td>16,900</td>
<td>16,900</td>
</tr>
<tr>
<td>Mean</td>
<td>1,440,000</td>
<td>0</td>
<td>16,900</td>
<td>0</td>
<td>16,900</td>
<td>16,900</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>1,440,000</td>
<td>0</td>
<td>16,900</td>
<td>0</td>
<td>16,900</td>
<td>16,900</td>
</tr>
<tr>
<td>Max</td>
<td>1,440,000</td>
<td>0</td>
<td>16,900</td>
<td>0</td>
<td>16,900</td>
<td>16,900</td>
</tr>
</tbody>
</table>
### Concluding Examples of Profiler Runs (2)

#### A Few Categories

TriRaw . raw_accounts . PROGRAM 3 varchar(20)

<table>
<thead>
<tr>
<th># Rows</th>
<th>Nulls</th>
<th>District</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>104,021</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### Many Categories

TriRaw . raw_redemptions . REDEMPTION_OFFER 3 varchar(30)

<table>
<thead>
<tr>
<th># Rows</th>
<th>Nulls</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>83,612</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Concluding Examples of Profiler Runs (3)

#### Numeric Value

AMA_Stage . RECEIVABLE_TXN_DETAIL . AMOUNT 3 varchar(8000)

<table>
<thead>
<tr>
<th># Rows</th>
<th>Nulls</th>
<th>District</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>8,133,330</td>
<td>0.00</td>
<td>0.23</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### Another Numeric Value

TriRaw . raw_accounts . BALANCE 5 varchar(6)

<table>
<thead>
<tr>
<th># Rows</th>
<th>Nulls</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>104,021</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Concluding Examples of Profiler Runs (4)

#### Reasonable Dates

TriRaw . raw_accounts . BIRTH_DATE 8 varchar(10)

<table>
<thead>
<tr>
<th># Rows</th>
<th>Nulls</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>104,021</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Concluding Examples of Profiler Runs (5)

#### Unusual Dates

AMA_Stage . RECEIVABLE_TXN . TXN_DATE 8 varchar(10)

<table>
<thead>
<tr>
<th># Rows</th>
<th>Nulls</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,861,249</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### Customer Job Title not too useful

AMA_Stage . CUSTOMER . JOB_TITLE 7 varchar(8000)

<table>
<thead>
<tr>
<th># Rows</th>
<th>Nulls</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>471,400</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### T-Shirt Size also not reliable

OliviaMatrix . RS_CONTACT_INFO . SIZE 11 varchar(16)

<table>
<thead>
<tr>
<th># Rows</th>
<th>Nulls</th>
<th>Empty</th>
<th>Numeric</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>204,051</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
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