

Data Profiling with R & MySQL

July 2011

Lightning Talk at SFBA useR Group



Jim Porzak – Data Science

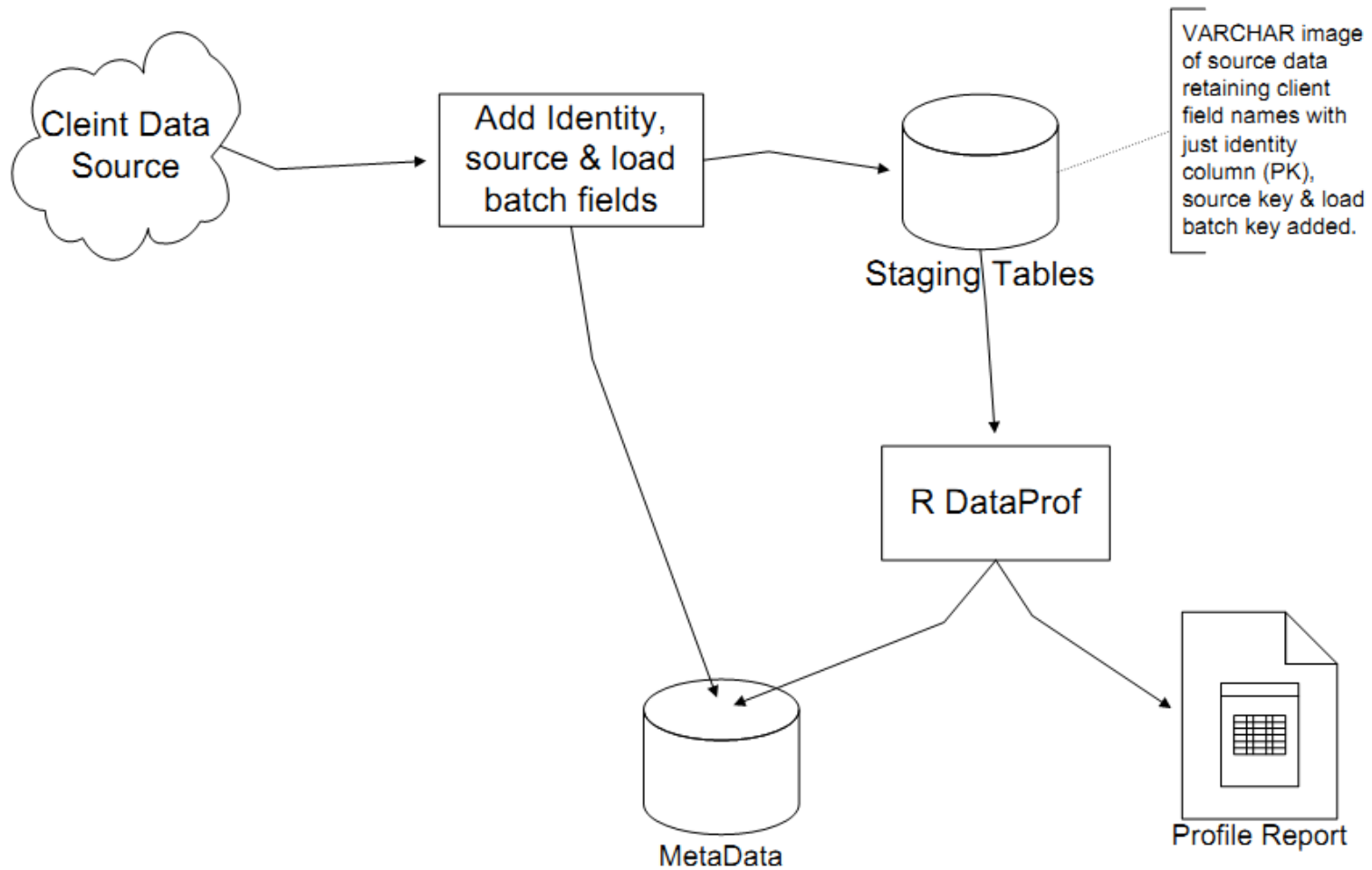
viadeo 

Background

- “*Discover Data Quality Issues as Early as Possible*”
 - Know quality limitations before building your model!
- First version originally presented at useR! 2006, Vienna
- Original Goals:
 - Very simple to run – just point to database and select tables to examine
 - Column by column profiling – don’t attempt JOINS or interactions
 - Output should be obvious to data geeks & DBA’s
 - Intelligent profiling based on true or estimated data type
- This Version:
 - Leverage JDBC instead of ODBC connection
 - Pick up MySQL meta data
 - Offload more heavy lifting to SQL queries
 - Use ggplot2 for seamless integration with *grid*
- See: *Data Quality – The Accuracy Dimension* by Jack E. Olson



Data Flow for a Profiler

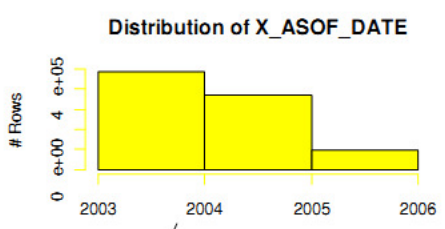


The Profiler Writes Output Panels

Header: Database details for field

AMA_Stage . RECEIVABLE_TXN . X_ASOF_DATE						19	varchar(8000)
	Rows	Nulls	Distinct	Empty	Numeric	Date	
#	3,861,249	2,908,244	28,564	0	0	953,005	
%	100.00	75.32	0.74	0.00	0.00	100.00	
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	
	2003-01-14	2003-08-12	2003-12-29	2004-02-12	2004-09-08	2005-10-17	
Head: NA NA NA NA NA NA							

Summary Counts & %'s
Empty, Numeric & Date only for character strings



Summary Stats if numeric or date

Appropriate plot type

Footer: Notes about field

From original version.



SQL Tricks

Not for this talk

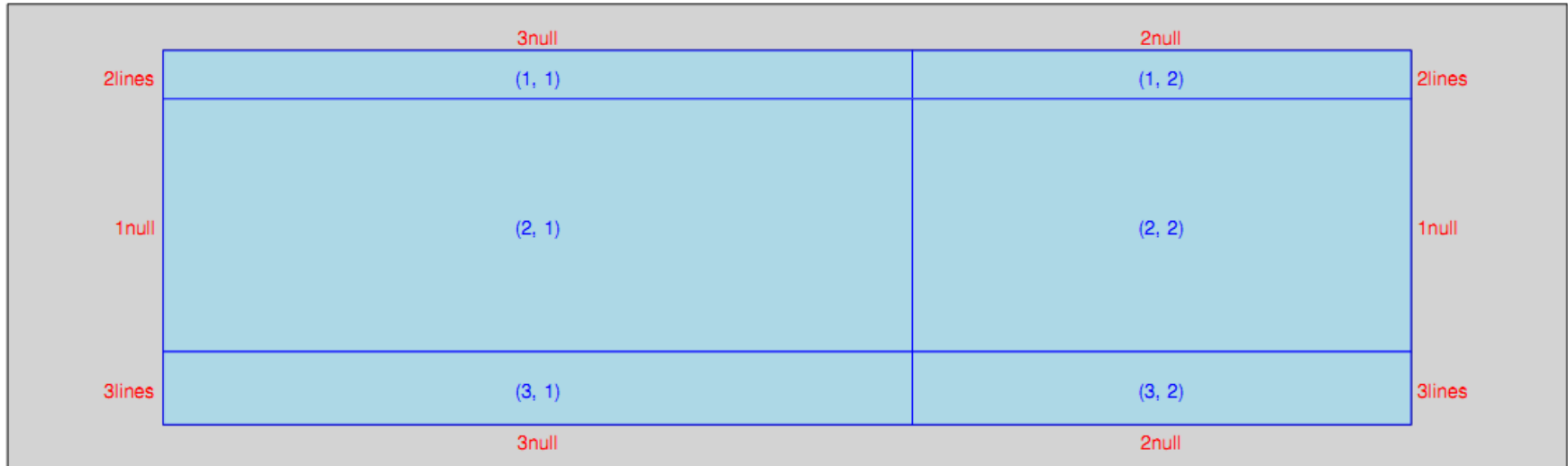


Grid Graphics Tricks (1)

- Set up panel name, output, & layout. Note: it's all about "viewports"!

```
pname <- paste("Plots/ColPlot_", ColDesc$Table, "_C", ithCol, ".png", sep = "")
png(pname, width = 10.5, height = 3, units = "in", res = 600, 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
vpTopLayout <- viewport(layout = TopLayout)
```



Grid Graphics Tricks (2)

- Walk through the viewports starting with the 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$ColType, "(",
                 ifelse(is.na(ColDesc$ColWidth), "", 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)

- Using grid based ggplot2 means no more fiddling with gridbase!

```
### generate plot as function of ColPlot value
p <- NULL      ## one of following should build a plot p, if not throw error plot
# a Category plot
if (ColDesc$ColPlot == "Category") {
  p <- ggplot(ColValue, NumRows, data = PlotValues, geom = "bar",
              stat = "identity", xlab = "", ylab = "# Rows",
              main = paste("Categories in", ColDesc$Column),
              fill = I("grey50")) + coord_flip()
}

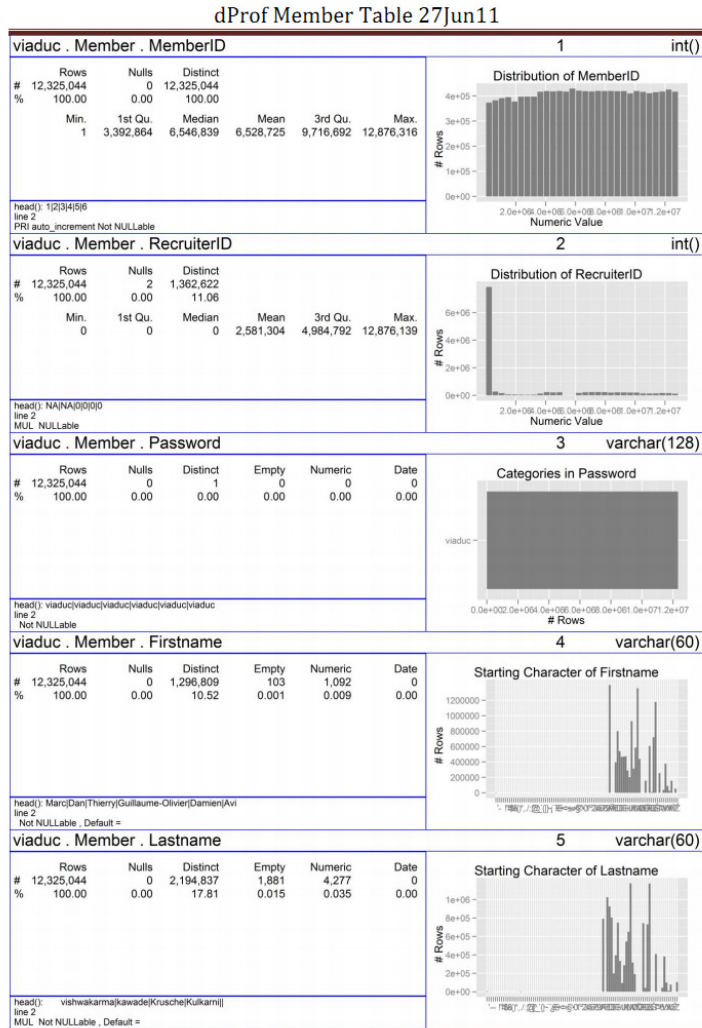
>>> cut <<<

# throw error plot if p not built above
if (is.null(p)) p <- "No plot was built"

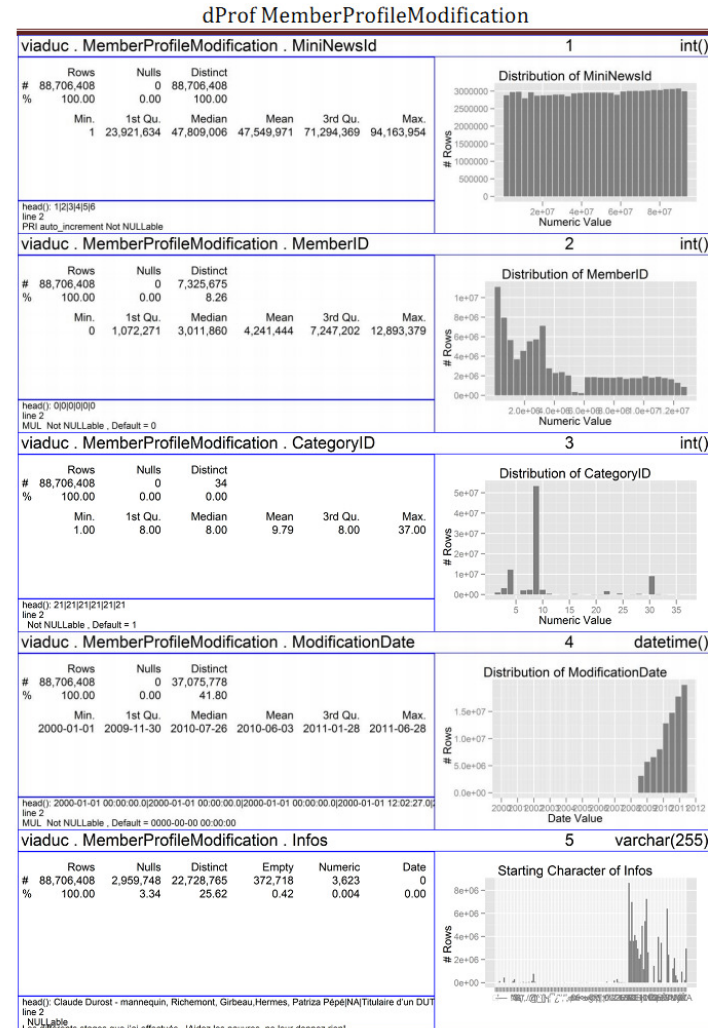
## Lastly output ggplot generated above as p
pushViewport(viewport(layout.pos.col = 2, layout.pos.row = 2:3))
print(p, vp = viewport(layout.pos.row = 2, layout.pos.col = 2))
dev.off()      ## no need to pop since we are now using png() to save file
```



A Couple of Examples (sampled data)



1



1



Thanks!

JPorzak@ViadeoTeam.com



viadeo 