teradataR

Release 1.0.1

October 2011

1 Introduction

The teradataR package is designed to allow users of R to interact with a Teradata database. Users can use many statistical functions directly against the Teradata system without having to extract the data into memory.

The teradataR package allows R users to easily connect to Teradata, establish a td data frame (virtual R data frame) to Teradata and to call in-database analytic functions within Teradata. This allows R users to work within their R console environment while leveraging the in-database functions. This package provides 44 different analytical functions and an additional 20 data connection and R infrastructure functions. In addition, we've added a function that will list the stored procedures within Teradata and provide the capability to call functions from R.

- 20 Functions to enable R infrastructure to operate with Teradata
- tdConnect - Connect to Teradata via ODBC or JDBC
- td.data.frame - Establish data frame connections to a Teradata table
- 44 in-database analytical functions callable from R. Sample of the functions include:
- Descriptive statistics: Overlap, histogram, frequency, statistics, matrix functions, and values analysis
- Reorganization functions: join, merge and samples
- Transformations: bincode, recode, rescale, sigmoid, zscore and null replacement
- K-Means clustering and Score K-Means
- Statistical tests: ks, dagostino.pearson, shapiro.wilk, bionomial, and wilcoxon
- R language features nrow, ncol, min, max, summary, as.dataframe, and dim
- Tool and R functions that allow users to create their own custom analytic functions that's callable by R.
- Teradata Warehouse Miner can capture any analytic stream including UDFs and create a stored procedure
    - Analytic process to create new derived predictive variables can be captured as a stored procedure.
    - Entire process to create or update an analytical data set can be captured as a stored procedure.
    - R function can list all the stored procedures within Teradata.
    - R function can call a stored procedure that runs in-database

teradataR allows R users to leverage all the benefits of in-database processing with Teradata:

- Eliminate data movement from Teradata to the R framework for key data intensive tasks.
- Leverage the speed of Teradata database's parallel processing to run analytics against big data.
- Ability to operate within the R console environment.
- Embed your frequently performed tasks to run in-database.
- R and teradataR are free downloads.

1.1 What's New in 1.0.1

The following updates have been made in the teradataR 1.0.1 release:

- **summary** has been enhanced to run faster
- JDBC support added – allows Windows or Mac users to run the package with JDBC
- Enhanced **td data frame** support, allows for manipulation to add columns and expressions
- **td.data.frame** enhanced to use Teradata 14.0 Fastpath Transform Functions (see Appendix B)
- **td.tapply** – Apply a select group of functions to columns of an array
- This teradataR User Guide Documentation

2 Getting Started with R and the teradataR Package

Support is provided for:

- R 2.11 and later (download R from http://www.r-project.org/)
- Teradata 12.0 and later (for all functions except Fastpath; see Appendix B)

The teradataR package can be installed into R using the downloaded local zip file.

a. Open the Rgui
b. Click on Packages->Install package(s) from local zip files…
c. Browse to the folder containing the teradataR_1.0.zip file, select the zip file and click Open

teradataR also requires the RODBC package (or RJDBC, DBI and rJava packages) to be installed and loaded. You can install that package in the Rgui, as well.

a. Open the Rgui
b. Click on Packages->Install package(s)…
c. Select a CRAN mirror (e.g. "USA (CA 1)") and click OK
d. Select RODBC (or RJDBC, DBI and rJava) and click OK

You must load the teradataR package each time you open an R session.

a. Open the Rgui
b. Click on Packages->Load package…
c. Select “teradataR” and click OK
d. Alternatively you can use the command line… `> library(teradataR)`

3 Basic Usage

teradataR allows R users to easily connect to Teradata, establish td data frames (virtual R data frames) to Teradata and to call in-database analytic functions within Teradata. This allows R users to work within their R console environment while leveraging the in-database functions.

3.1 Help

Help for the TeradataR package can be found by using the “`help`” or “`?`” commands in the R Console.

`> help(teradataR)` – Provides brief information on the TeradataR package. You can click on the “Index” link at the bottom of this help page to see a list and get help on all TeradataR functions.

`> help(`*`function`*`)` – Provides help information on the specified function.

`> RShowDoc(“teradataR”, package=“teradataR”)` – Opens this manual.

3.2 Making a connection to Teradata

Once the teradataR package is loaded, you need to connect to a Teradata database. ODBC (and JDBC) works by setting up a connection or channel from the client (the R Console) to the Teradata database as specified in the DSN. Such connections are normally used throughout a session, but should be closed explicitly at the end of the session - however RODBC will clean up after you if you forget.

The simplest way to make a connection is:

`> tdConnect(“`*`teradata_dsn`*`”)`

In this example, “*teradata_dsn*” is the Data Source Name for a connection to your Teradata database. The Data Source Name should be setup in the ODBC Administrator for ODBC. For JDBC, you just use the Database Server Name or IP address.

If the Teradata user and password are not stored in the DSN, the full usage of tdConnect can be used.

Usage

```
tdConnect(“dsn”, uid = "uid", pwd = "pwd", database = "db", dType =
c(“odbc”,”jdbc”))
```

Arguments

`dsn`	string containing the data source name to connect to.
`uid`	string containing the user id.
`pwd`	string containing the password.
`database`	string containing the default database.
`dType`	String containing the driver type: odbc (default) or jdbc

The tdConnect function will set the connection information in the tdConnection global variable and is the default connection for teradataR functions. When you are finished with the connection to the Teradata database, you can close the connection by typing the following command in the R Console:

```
> tdClose()
```

3.3 Creating a Teradata data frame

A Teradata data frame is an R object that represents a Teradata table. This is the main object that stores the link between a Teradata table and the R environment. The object initializes with certain information upon creation such as total number of rows and what the column names of the table are. No data from the table is actually moved to or resident in the R system, only those initial values of rows and columns are persisted and stored within the td data frame object. All analysis is performed via a td data frame.

To create a Teradata data frame, use the following command:

```
> tdf <- td.data.frame("table_name")
```

In this command, "*table_name*" is a table in the current Teradata database for the connection. If the table resides in another database, you can add the database parameter:

```
> tdf2 <- td.data.frame("table_name", "database")
```

In the above commands, "tdf" and "tdf2" are now the pointers to the Teradata tables. You will use these pointers to run the analytic functions against.

3.4 Running in-database analytic functions

TeradataR provides 44 different analytical functions and an additional 20 data connection and R infrastructure functions (see Appendix A for a list). These functions are called using the Teradata data frame pointer. The functions perform the analysis in the Teradata database (not in R system memory) and the results are returned to the R Console. Below are some common analytic function examples.

1. Collect statistics from a Teradata table. Count, minimum, maximum, mean, sum, uncorrected sum of squares, corrected sum of squares, variance, standard deviation, skewness, kurtosis, standard error, and coefficient of variance are the available statistics to choose from.

> td.stats(tdf, “*income*”)

```
     col xcnt xmin  xmax   xmean   xsum       xuss       xcss      xvar     xstd     xskew      xkurt  xstderr
xcvar
1 income   10    0 55888 22501.7 225017 8592620397 3529355368 352935537 18786.58 0.9013882 -0.1142682 5940.838
83.4896
```

2. Query the Teradata table and return summary results for all the columns associated with the td data frame.

> summary(tdf)

```
    cust_id           income           age       years_with_bank  nbr_children gender marital_status
 Min.   :1362480   Min.   :    0   Min.   :13.0   Min.   :0.0     Min.   :0.0   F:7    Min.   :1
 1st Qu.:1362484   1st Qu.: 7083   1st Qu.:33.0   1st Qu.:2.0     1st Qu.:0.0   M:3    1st Qu.:1
 Median :1362487   Median :15778   Median :38.0   Median :3.5     Median :0.0          Median :1
 Mean   :1362487   Mean   :22502   Mean   :44.8   Mean   :3.6     Mean   :1.1          Mean   :2
 3rd Qu.:1362489   3rd Qu.:40252   3rd Qu.:71.0   3rd Qu.:6.0     3rd Qu.:2.0          3rd Qu.:3
 Max    :1362496   Max    :55888   Max    :77.0   Max    :7.0     Max    :5.0          Max    :3
```

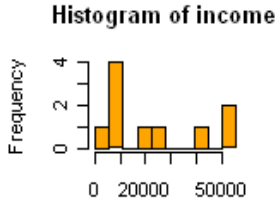
3. Determine the nature and overall quality of the data. Find the number of rows with non-null values, with value 0, with a positive value, with a negative value, the number of unique values and the number of rows containing blanks in the given column (return values based on the type of column).

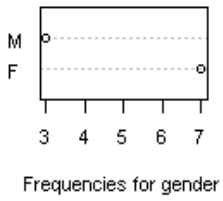
> td.values(tdf, “income”)

```
      col xnull xzero xpos xneg xunique
1 cust_id     0     0   10    0      10
```

4. Build a histogram graph for each numeric column in the td data frame. A frequency graph is produced for non-numeric columns.

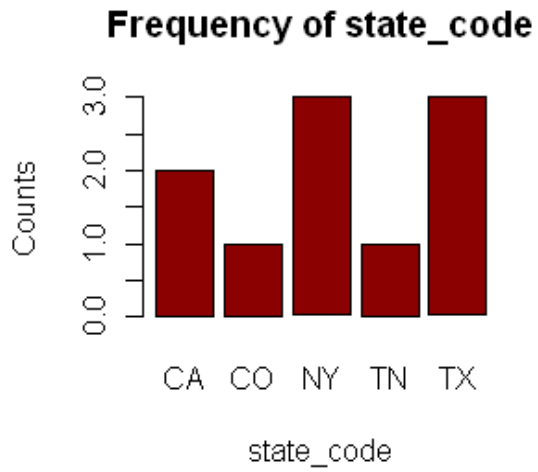
> hist(tdf)





5. Produce a frequency analysis graph of a column in a td data frame.

> td.freq(tdf, “state_code”)



6. Create a new column in a td data frame for analysis.

```
> tdf["newcolumn"] <- tdf["income"]/tdf["age"]
```

This will add a new column to your td.data.frame that will point to the expression of income/age. Any new columns that you add do not alter your database table. They are simply a definition of an expression that you can now use in other calls such as td.sample, td.stats, etc. If after altering a td data frame, you wish to actually persist your new expressions to the database, you can use the following command:

```
> tdf3 <- as.td.data.frame(tdf, tableName="newTable", database="myDb")
```

This will generate a new table within your Teradata database in the myDb database named newTable. It will also return to you a td data frame pointer in tdf3.

7. Create a subset of a td data frame.

```
> tdfSub <- subset(tdf, age >= 65)
```

tdfSub is now a td data frame which points to the same table as tdf but adds the condition of only including the rows where the age is greater than or equal to 65. You can display your td data frame at the prompt and you will see that tdfSub has conditions attached to it.

8. Apply a function to a td data frame using td.tapply.

```
> tdfApp <- td.tapply(tdf["income"], tdf["gender"], td.stats)
```

Just like the tapply function of R, this will apply the td.stats function to the income variable and group your results by gender returning you an array. You can also add the asdf=TRUE to your td.tapply call and it will return you your results as a data frame instead. Currently only sum, min, max, mean, td.stats, and td.values are supported for the functions you can apply.

A Function List

Td.recode	Recode
Td.rescale	Rescale Values of Column
Td.sample	Sample Rows
Td.shapiro.wilk	Shapiro Wilk
Td.sigmoid	Sigmoid Transformation
Td.smirnov	Smirnov Test
Td.solve	Solve a system of equations
Td.stats	General Statistics
Td.t.paired	T Test Paired
Td.t.unpaired	T Test Unpaired
Td.t.unpairedi	T Test - Unpaired Indicator
Td.values	Values
Td.wilcoxon	Wilcoxon Test
Td.zscore	Zscore Transformation
tdClose	Close connection
tdConnect	Connect to Teradata database
tdMetadataDB	Set metadata database
tdQuery	Query Teradata Database
teradataR	Allow access to Teradata via R
[.td.data.frame	Extract Teradata Data Frame
[<-.td.data.frame	Replace value of Teradata Data Frame

B Fastpath Function List

Fastpath functions are available only with Teradata 14.0 or later (* functions also work with 13.10).
(Teradata 14.0 availability planned for Dec 2011.)

ASCII	Return first character as an integer
CEIL *	Returns smallest integer >= input
CHR	Return latin Ascii character given number
DECODE	Compare input to search and return result
EDITDISTANCE	Min edit ops to transform str1 to str2
FLOOR *	Returns largest integer <= input
GREATEST	Return largest input parameter
INSTR	Search input for search string and return position
INTCAP	Capitalize first character in each word
LEAST	Return smallest input parameter
LENGTH	Return number of characters in input
LPAD	Left pad input string to length with fill string
LTRIM	Left trim input based on 2nd input string
NGRAM	Return number of n-gram matches between two input strings
OREPLACE	Replace every occurrence of the search string in the source string with the replace string
OTRANSLATE	Return source string by replacing the from input to the to input strings
POWER	Return input base value raised to the power of the input exponent
ROUND	Round input to places input
RPAD	Right pad input string to length with fill string
RTRIM	Right trim input based on 2nd input string
SIGN	Return the sign of the input numeric
TO_CHAR	Convert date, time, time stamp, interval or numeric value into a character string
TO_NUMBER	Convert string to number data type
TRUNC	Replace value of Teradata Data Frame