RevoScaleR ODBC Data Import Guide

The correct bibliographic citation for this manual is as follows: Microsoft Corporation. 2016. *RevoScaleR ODBC Data Import Guide*. Microsoft Corporation, Redmond, WA.

RevoScaleR ODBC Data Import Guide

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
U.S.A.

Revised on November 9, 2015.

We want our documentation to be useful, and we want it to address your needs. If you have comments on this or any Microsoft R Services document, write to revodoc@microsoft.com.

Contents

Before You Begin

RevoScaleR allows you to read data from virtually any database for which you can obtain an ODBC driver. ODBC stands for "Open Database Connectivity," and is a standard software interface for accessing database management systems, which are usually conflated with the databases they manage and simply called "databases."

ODBC connectivity is managed by an ODBC *Driver Manager*. On Windows, the ODBC driver manager is part of the operating system. On Linux systems, there are two commonly available ODBC driver managers, unixODBC and iodbc, along with commercial implementations of the Windows ODBC driver manager which are sometimes distributed with commercial database drivers. *RevoScaleR works and has been tested with unixODBC, but not iodbc.* It is important that the version of unixODBC you install be compatible with the databases you need to connect to. See the section Configuring unixODBC for complete details.

This guide is not intended for use with Teradata—if you need to connect to a Teradata data warehouse, see the manual *Microsoft R Services Client Installation Guide for Teradata*.

Internally, we have tested the following databases:

- SQLite 3
- SQL Server 2008 and 2014
- Oracle Express
- MySQL® 5.1

For each database you intend to use as a data source, you must do the following:

1. Install and configure the database. The database may be installed locally or on a server.

2. Add data to the database. This will involve different steps with different databases.

3. Install the appropriate ODBC driver(s) on your local machine. These should be 64-bit drivers.

4. Create an *RxOdbcData* data source using information appropriate to your database installation.

5. Use the *rxImport* function to read data from the data source into an .xdf file for further use by RevoScaleR.

This manual will focus on steps 3 through 5 above; some resources to help you with steps 1 and 2 can be found in the Appendix. In most organizations, however, these steps will be handled by your local database administrator; this person is also the one to ask for the database-specific information required in step 4.

Some Quick Examples

If you have a configured database and the necessary ODBC drivers, actually extracting data is quite straightforward. For example, most Windows systems come with SQL Server ODBC drivers pre-installed, so you can go right to work extracting data from a SQL server database with just a username, password, and a few specifics like the actual database name and table you want to extract data from. These can be combined into a single SQL connection string and then used to extract data as follows (note that some drivers and operating systems may require somewhat different syntax, so for example, the curly braces around the driver name may need to be omitted; Linux drivers typically will reject connection strings containing white space, etc.):

```
sConnectionStr <- "Driver={SQL Server};Server=win-database01;
       Database=TestData;Uid=mktest;Pwd=sqlpwd;"
claimsSQL = "SELECT * FROM claims"
claimsDS <- RxOdbcData(sqlQuery = claimsSQL,
  connectionString=sConnectionStr)
claimsFile <- RxXdfData("claimsFromODBC.xdf")
rxImport(claimsDS, claimsFile, overwrite=TRUE)
rxGetInfo(claimsFile, getVarInfo=TRUE, numRows=10)
```

This returns the following:

```
File name: claimsFromODBC.xdf
Number of observations: 128
Number of variables: 6
Number of blocks: 1
Variable information:
  Var 1: RowNum, Type: integer, Low/High: (1, 128)
  Var 2: age, Type: character, Storage: fixed, Width: 10
  Var 3: car.age, Type: character, Storage: fixed, Width: 10
  Var 4: type, Type: character, Storage: fixed, Width: 10
  Var 5: cost, Type: numeric, Storage: float32, Low/High:
  (11.0000, 850.0000)
  Var 6: number, Type: numeric, Storage: float32, Low/High:
  (0.0000, 434.0000)
```

```
Data (10 rows starting with row 1):
   RowNum   age car.age type cost number
1       1 17-20     0-3    A  289      8
2       2 17-20     4-7    A  282      8
3       3 17-20     8-9    A  133      4
4       4 17-20     10+    A  160      1
5       5 17-20     0-3    B  372     10
6       6 17-20     4-7    B  249     28
7       7 17-20     8-9    B  288      1
8       8 17-20     10+    B   11      1
9       9 17-20     0-3    C  189      9
10     10 17-20     4-7    C  288     13
```

Since we are extracting all the data from the table with our SQL query, it can be simpler to just provide the table argument to RxOdbcData:

```
  sConnectionStr <- "Driver={SQL Server};Server=win-database01;
          Database=TestData; Uid=mktest; Pwd=sqlpwd;"
claimsDS2 <- RxOdbcData(table="claims",
   connectionString=sConnectionStr)
  claimsFile <- RxXdfData("claimsFromODBC.xdf")
  rxImport(claimsDS2, claimsFile, overwrite=TRUE)
  rxGetInfo(claimsFile, getVarInfo=TRUE, numRows=10)
```

This gives the same results as before.

Configuring unixODBC

If you intend to use ODBC connectivity in a Linux environment, you must install unixODBC. You can check to see if unixODBC has been installed on your system using the following command:

```
rpm -qa | grep unixODBC
```

If this returns nothing, you can install unixODBC using the following commands:

On RHEL systems:

```
sudo yum install unixODBC
```

On SLES systems:

```
sudo zypper install unixODBC
```

Installing ODBC Drivers

ODBC drivers are available for most commercial and open-source databases, either from the database vendor or third parties. Usually, the simplest way to find the appropriate

driver is to use your favorite search engine and query "ODBC driver for <database>." The download page will generally give you all the necessary installation instructions.

It is important to note that the ODBC drivers need to be installed on your local machine even if you are using a remote database server; the ODBC drivers are used by the database clients that will actually access the database.

Installing SQLite and the SQLite ODBC Drivers on Windows

SQLite is a "serverless database" that allows you store a complete database, with tables, indices, and views, in a single disk file. It is simple to install; you simply download the three Windows binary zip files from http://www.sqlite.org/download.html and extract them into your C:\Windows\System32 folder.

An ODBC driver for SQLite is available at http://www.ch-werner.de/sqliteodbc. We have tested versions .88 and later; version .92 has a known incompatibility in that RevoScaleR is unable to obtain correct row position data using that version. We recommend version .93 or later. Download the 64-bit executable and run the executable to install the appropriate driver.

Installing SQLite and the SQLite ODBC Drivers on Linux

SQLite is a "serverless database" that allows you store a complete database, with tables, indices, and views, in a single disk file. To install SQLite and the SQLite ODBC drivers on Linux, perform the following steps:

1. Ensure that you have unixODBC installed:
   ```
   ls -l /usr/lib64/libodb*
   ```

2. Obtain the SQLite sources from http://www.sqlite.org/download.html.

3. Unpack the sources using tar zxvf <tarball-name>, then cd to the directory containing the unpacked sources.

4. Run the standard configure and make commands:
   ```
   ./configure
   make
   sudo make install
   ```

5. Obtain the SQLite ODBC driver source from http://www.ch-werner.de/sqliteodbc. We have tested versions .88 and later; version .92 has a known incompatibility in that RevoScaleR is unable to obtain correct row position data using that version. We recommend version .88 if you are running

Red Hat Enterprise Linux 5 or equivalent with standard yum install of unixODBC, otherwise version .93 or later.

6. Unpack the sources using tar zxvf <tarball-name>, then cd to the directory containing the unpacked sources.

7. Run the standard configure and make commands:

```
./configure
make
sudo make install
```

8. Edit the file /etc/odbcinst.ini, adding the following section:

```
[SQLite3 ODBC Driver]
Description=SQLite ODBC Driver
Driver=/usr/local/lib/libsqlite3odbc.so
Setup=/usr/local/lib/libsqlite3odbc.so
Threading=2
```

Installing MySQL ODBC Drivers on Windows

MSI installers for 64-bit MySQL ODBC drivers are available at the MySQL Web site, http://dev.mysql.com/downloads/connector/odbc/. Download and run the appropriate installer to install the driver.

Installing MySQL ODBC Connector on Red Hat Enterprise Linux

MySQL has connectors that work with either the standard yum installs of the unixODBC driver manager or with the latest unixODBC 2.3.1. If you are using the standard yum install, use the mysql-connector-odbc-5.1.12-1.rhel5.x86_64.rpm which you can download from their website, http://dev.mysql.com/downloads/connector/odbc/. Download and run the appropriate installer to install the connector.

If you are using the unixODBC 2.3.1 driver manager, use the mysql-connector-odbc-5.2.4-1.rhel5.x86_64.rpm, available on the same site.

Once you have installed the connector, configure your installation by editing the files 'odbc.ini' and 'odbcinst.ini'. The default location for these is the system /etc folder. If you create them somewhere else, you can ensure they are found by setting the following environment variables

export ODBCINI=/path/to/odbc.ini

export ODBCSYSINI=/path/to/odbcinst.ini

Using ODBC Data Sources

Working with SQLite Files

SQLite is a special database system in that it is file-based and serverless. Each file corresponds to a database, which may contain multiple tables, indexes, etc. Because there is no server, there are no credentials to worry about (file permissions remain important, however). So, in general, working with a SQLite database is quite simple—the connection string needs to specify only the driver (on Windows, the name of the driver is the name shown in the Drivers tab of the ODBC Data Source Administrator under Name) and the file name, as in the following example:

```
  SQLiteFile <- file.path(system.file("SampleData",
          package="RevoScaleR"), "claims.sqlite")
SQLiteConnString <- sprintf(
          "Driver={SQLite3 ODBC Driver};Database=%s",
          SQLiteFile)
  SQLiteDS <- RxOdbcData(sqlQuery = "SELECT * FROM claims",
                         connectionString=SQLiteConnString)
  SQLiteOutfile <- "claimsFromSQLite.xdf"
  rxImport(SQLiteDS, SQLiteOutfile, overwrite=TRUE)
  rxGetInfo(SQLiteOutfile, getVarInfo=TRUE, numRows=10)
```

Working with SQL Server

In the section Some Quick Examples, we gave an example of extracting data from a SQL Server database using a connection string. We now show the same example, but instead of using a connection string, we use the server, dsn, user, and password arguments:

```
  claimsSQL = "SELECT * FROM claims"
  claimsDS2<- RxOdbcData(sqlQuery = claimsSQL,
    connectionString = "DSN=MSSQLDSN;Uid=mktest;Pwd=passwd;")
  claimsFile <- RxXdfData("claimsFromODBC.xdf")
  rxImport(claimsDS2, claimsFile, overwrite=TRUE)
  rxGetInfo(claimsFile, getVarInfo=TRUE, numRows=10)
```

As before, we get the claims data:

```
File name: claimsFromODBC.xdf
Number of observations: 128
Number of variables: 6
Number of blocks: 1
```

```
Variable information:
Var 1: RowNum, Type: integer, Low/High: (1, 128)
Var 2: age, Type: character
Var 3: car.age, Type: character
Var 4: type, Type: character
Var 5: cost, Type: numeric, Storage: float32, Low/High:
  (11.0000, 850.0000)
Var 6: number, Type: numeric, Storage: float32, Low/High:
  (0.0000, 434.0000)
Data (10 rows starting with row 1):
   RowNum   age car.age type cost number
1       1 17-20     0-3    A  289      8
2       2 17-20     4-7    A  282      8
3       3 17-20     8-9    A  133      4
4       4 17-20     10+    A  160      1
5       5 17-20     0-3    B  372     10
6       6 17-20     4-7    B  249     28
7       7 17-20     8-9    B  288      1
8       8 17-20     10+    B   11      1
9       9 17-20     0-3    C  189      9
10     10 17-20     4-7    C  288     13
```

Working with Oracle Express

Oracle Express is a free version of the popular Oracle database management system. It is intended as a "starter" version; working with other offerings from Oracle uses the same ODBC drivers. Here we give the Oracle SQL statement to show all the tables in a database (this differs from standard SQL implementations):

```
tablesDS <- RxOdbcData(sqlQuery="select * from user_tables",
connectionString = "DSN=ORA10GDSN;Uid=system;Pwd=X8dzlkjWQ")
OracleTableDF <- rxImport(tablesDS, overwrite=TRUE)
OracleTableDF[,1]
```

This yields a list of tables similar to the following:

```
 [1] "MVIEW$_ADV_WORKLOAD"           "MVIEW$_ADV_BASETABLE"
 [3] "MVIEW$_ADV_SQLDEPEND"          "MVIEW$_ADV_PRETTY"
 [5] "MVIEW$_ADV_TEMP"               "MVIEW$_ADV_FILTER"
 [7] "MVIEW$_ADV_LOG"
"MVIEW$_ADV_FILTERINSTANCE"
 [9] "MVIEW$_ADV_LEVEL"              "MVIEW$_ADV_ROLLUP"
[11] "MVIEW$_ADV_AJG"                "MVIEW$_ADV_FJG"
[13] "MVIEW$_ADV_GC"                 "MVIEW$_ADV_CLIQUE"
[15] "MVIEW$_ADV_ELIGIBLE"           "MVIEW$_ADV_OUTPUT"
[17] "MVIEW$_ADV_EXCEPTIONS"         "MVIEW$_ADV_PARAMETERS"
[19] "MVIEW$_ADV_INFO"               "MVIEW$_ADV_JOURNAL"
[21] "MVIEW$_ADV_PLAN"               "AQ$_QUEUE_TABLES"
[23] "AQ$_QUEUES"                    "AQ$_SCHEDULES"
[25] "AQ$_INTERNET_AGENTS"           "AQ$_INTERNET_AGENT_PRIVS"
```

```
 [27] "DEF$_AQCALL"                    "DEF$_AQERROR"
 [29] "DEF$_ERROR"                     "DEF$_DESTINATION"
 [31] "DEF$_CALLDEST"                  "DEF$_DEFAULTDEST"
 [33] "DEF$_LOB"                       "DEF$_TEMP$LOB"
 [35] "DEF$_PROPAGATOR"                "DEF$_ORIGIN"
 [37] "DEF$_PUSHED_TRANSACTIONS"       "OL$"
 [39] "OL$HINTS"                       "OL$NODES"
 [41] "LOGMNR_SESSION_EVOLVE$"         "LOGMNR_HEADER1$"
 [43] "LOGMNR_HEADER2$"                "LOGMNR_UID$"
 [45] "LOGMNRC_DBNAME_UID_MAP"         "LOGMNR_DICTSTATE$"
 [47] "LOGMNR_DICTIONARY$"             "LOGMNR_OBJ$"
 [49] "LOGMNR_USER$"                   "LOGMNRC_GTLO"
 [51] "LOGMNRC_GTCS"                   "LOGMNRC_GSII"
 [53] "LOGMNR_TAB$"                    "LOGMNR_COL$"
 [55] "LOGMNR_ATTRCOL$"                "LOGMNR_TS$"
 [57] "LOGMNR_IND$"                    "LOGMNR_TABPART$"
 [59] "LOGMNR_TABSUBPART$"             "LOGMNR_TABCOMPART$"
 [61] "LOGMNR_TYPE$"                   "LOGMNR_COLTYPE$"
 [63] "LOGMNR_ATTRIBUTE$"              "LOGMNR_LOB$"
 [65] "LOGMNR_CDEF$"                   "LOGMNR_CCOL$"
 [67] "LOGMNR_ICOL$"                   "LOGMNR_LOBFRAG$"
 [69] "LOGMNR_INDPART$"                "LOGMNR_INDSUBPART$"
 [71] "LOGMNR_INDCOMPART$"             "LOGMNRP_CTAS_PART_MAP"
 [73] "LOGMNRT_MDDL$"                  "LOGMNR_LOG$"
 [75] "LOGMNR_PROCESSED_LOG$"          "LOGMNR_SPILL$"
 [77] "LOGMNR_AGE_SPILL$"
 "LOGMNR_RESTART_CKPT_TXINFO$"
 [79] "LOGMNR_ERROR$"                  "LOGMNR_RESTART_CKPT$"
 [81] "LOGMNR_FILTER$"                 "LOGMNR_PARAMETER$"
 [83] "LOGMNR_SESSION$"                "LOGSTDBY$PARAMETERS"
 [85] "LOGSTDBY$EVENTS"                "LOGSTDBY$APPLY_PROGRESS"
 [87] "LOGSTDBY$APPLY_MILESTONE"       "LOGSTDBY$SCN"
 [89] "LOGSTDBY$PLSQL"
 "LOGSTDBY$SKIP_TRANSACTION"
 [91] "LOGSTDBY$SKIP"                  "LOGSTDBY$SKIP_SUPPORT"
 [93] "LOGSTDBY$HISTORY"               "REPCAT$_REPCAT"
 [95] "REPCAT$_FLAVORS"                "REPCAT$_REPSCHEMA"
 [97] "REPCAT$_SNAPGROUP"              "REPCAT$_REPOBJECT"
 [99] "REPCAT$_REPCOLUMN"              "REPCAT$_KEY_COLUMNS"
[101] "REPCAT$_GENERATED"              "REPCAT$_REPPROP"
[103] "REPCAT$_REPCATLOG"              "REPCAT$_DDL"
[105] "REPCAT$_REPGROUP_PRIVS"         "REPCAT$_PRIORITY_GROUP"
[107] "REPCAT$_PRIORITY"               "REPCAT$_COLUMN_GROUP"
[109] "REPCAT$_GROUPED_COLUMN"         "REPCAT$_CONFLICT"
[111] "REPCAT$_RESOLUTION_METHOD"      "REPCAT$_RESOLUTION"
[113] "REPCAT$_RESOLUTION_STATISTICS"
 "REPCAT$_RESOL_STATS_CONTROL"
[115] "REPCAT$_PARAMETER_COLUMN"       "REPCAT$_AUDIT_ATTRIBUTE"
[117] "REPCAT$_AUDIT_COLUMN"           "REPCAT$_FLAVOR_OBJECTS"
[119] "REPCAT$_TEMPLATE_STATUS"        "REPCAT$_TEMPLATE_TYPES"
[121] "REPCAT$_REFRESH_TEMPLATES"
 "REPCAT$_USER_AUTHORIZATIONS"
[123] "REPCAT$_OBJECT_TYPES"
 "REPCAT$_TEMPLATE_REFGROUPS"
[125] "REPCAT$_TEMPLATE_OBJECTS"       "REPCAT$_TEMPLATE_PARMS"
```

```
[127] "REPCAT$_OBJECT_PARMS"           "REPCAT$_USER_PARM_VALUES"
[129] "REPCAT$_TEMPLATE_SITES"         "REPCAT$_SITE_OBJECTS"
[131] "REPCAT$_RUNTIME_PARMS"          "REPCAT$_TEMPLATE_TARGETS"
[133] "REPCAT$_EXCEPTIONS"
  "REPCAT$_INSTANTIATION_DDL"
[135] "REPCAT$_EXTENSION"              "REPCAT$_SITES_NEW"
[137] "SQLPLUS_PRODUCT_PROFILE"        "HELP"
[139] "TestFile"                       "1987"
```

Working with MySQL Files on Red Hat Enterprise Linux

You first need to specify the name of your DSN. On Linux, this will be the same as the name you specify for the ODBC configuration.

```
### Test of import from 'centos-database01' ###

sConnectionStr = "DSN=ScaleR-ODBC-test"
sUserSQL = "SELECT * FROM airline1987"
airlineXDFName <- file.path(getwd(), "airlineimported.xdf")
airlineODBCSource <- RxOdbcData(sqlQuery = sUserSQL,
  connectionString = sConnectionStr, useFastRead = TRUE)
rxImport(airlineODBCSource, airlineXDFName, overwrite = TRUE)
```

Limitations on SQL Queries

The sqlQuery argument to *RxOdbcData* is limited to data extraction queries (SELECT and SHOW statements, primarily) because *RxOdbcData* is currently intended to be used for reading data from the database into a .xdf file. In particular, INSERT queries are not supported. Also, because each query is used to populate a single .xdf file, multiple queries (that is, queries separated by a semicolon ";") are not supported. Compound queries, however, that produce a single extracted table, (that is, queries linked by AND or OR, or involving multiple FROM clauses) are supported.

Specifying Variable Data Types

Data stored in databases may be stored differently from how you want to store the data in R. You can use the *colClasses*, *colInfo*, and *stringsAsFactors* arguments to *RxOdbcData* to specify how columns are stored in R. The *stringsAsFactors* argument is the simplest to use; if specified, any character string column not otherwise accounted for by the *colClasses* or *colInfo* argument is stored as a factor in R, with levels defined according to the unique character strings found in the column. The *colClasses* argument allows you to specify a particular class for a particular variable; *colInfo* is similar, but it also allows you to specify a set of levels for a factor variable. The following example shows all three

arguments being combined to allow us to modify the data types of several variables in the claims data:

```
sConnectionStr <- "Driver={SQL Server};Server=win-database01;
Database=TestData;Uid=mktest;Pwd=sqlpwd;";
colInfoList <- list("car.age" = list(type = "factor", levels =
c("0-3",
  "4-7", "8-9", "10+")))
claimsFile <- RxXdfData("claimsFromODBC_Modified.xdf")
claimsSQL = "SELECT * FROM claims"
claimsDS <- RxOdbcData(sqlQuery = claimsSQL,
connectionString=sConnectionStr,
  colClasses = c(number="integer"), colInfo = colInfoList,
  stringsAsFactors=TRUE)
rxImport(claimsDS, claimsFile, overwrite=TRUE)
rxGetInfo(claimsFile, getVarInfo = TRUE)
```

This returns the following:

```
File name: claimsFromODBC_Modified.xdf
Number of observations: 128
Number of variables: 6
Number of blocks: 1
Variable information:
Var 1: RowNum, Type: integer, Low/High: (1, 128)
Var 2: age
       8 factor levels: 17-20 21-24 25-29 30-34 35-39 40-49 50-
  59 60+
Var 3: car.age
       4 factor levels: 0-3 4-7 8-9 10+
Var 4: type
       4 factor levels: A B C D
Var 5: cost, Type: numeric, Storage: float32, Low/High:
  (11.0000, 850.0000)
Var 6: number, Type: integer, Low/High: (0, 434)
```

Appendix: Setting Up Databases

RevoScaleR assumes you have data; configuring databases is beyond the scope of this document. However, if you are new to databases, it can help to have some place to start. This appendix is intended as a pointer to additional resources that can help you get started building your own databases.

Major Database Web Sites

- **SQLite**: http://www.sqlite.org/

- **MySQL**: http://www.mysql.com/
- **SQL Server**: http://www.microsoft.com/sqlserver/en/us/default.aspx
- **Oracle**: http://www.oracle.com/
- **Sybase**: http://www.sybase.com/

Getting Started Documentation

- **SQLite**: http://www.sqlite.org/quickstart.html
- **MySQL**: http://dev.mysql.com/tech-resources/articles/mysql_intro.html
- **SQL Server**: http://technet.microsoft.com/en-us/library/bb500434.aspx
- **Oracle Express**: http://download.oracle.com/docs/cd/B25329_01/doc/admin.102/b25610/toc.htm
- **Sybase**: http://download.sybase.com/pdfdocs/asg1150e/tutorial.pdf