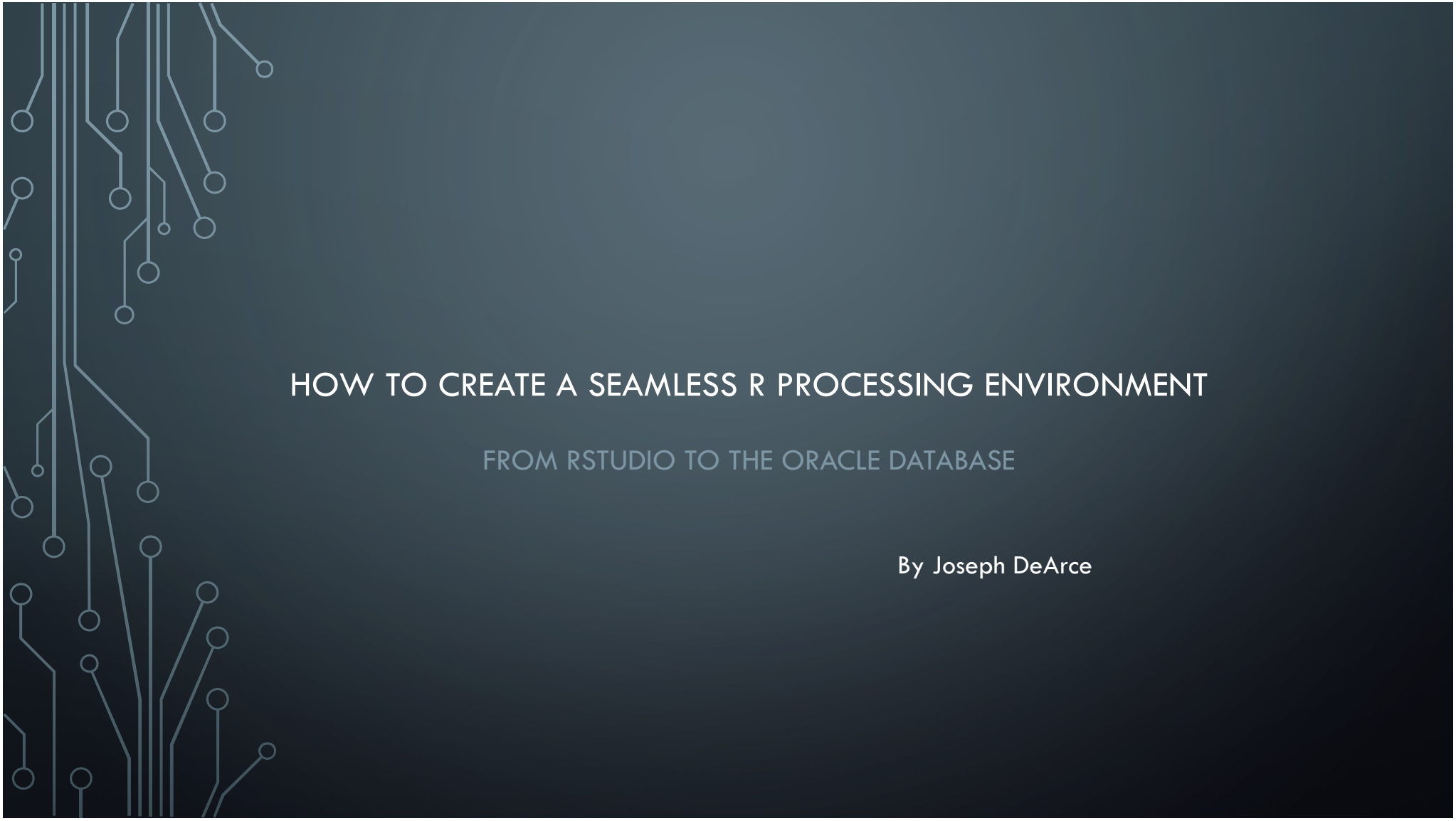

HOW TO CREATE A SEAMLESS R PROCESSING ENVIRONMENT

FROM RSTUDIO TO THE ORACLE DATABASE

By Joseph DeArce

WHO AM I

- I am a Sr. Oracle DBA with over 21 years of experience with RDBMS architecture, design, programming and administration.
- Currently I am the Database Manager for the CUNY Research Foundation in NYC working with ODI, OBIEE and EXADATA.

AGENDA

- History of R as a programing language.
- The hardware used.
- The four software packages that are needed to create a seamless data processing environment.
- What is DBI and ROracle and how do they work together.
- 64 versus 32 bit - why do we care?

AGENDA CONTINUED

- ROracle's native connection to the database.
- ROracle/DBI native commands.
- Making a connection to a database.
- Benchmarks for ROracle and SQL Developer.
- Conclusions.
- Helpful web sites.

HARDWARE TEST PLATFORM

- An Acer laptop running Windows 10 with a Core7i quad processor.
- 16GB of memory.
- 1TB hard drive (not SSD).
- Installed is an Oracle 12c Enterprise database with a SGA of 6GB.
- The database has a custom configuration.
- There are four PDBs on the Oracle 12c R1 instance.

HISTORY OF R: WHERE DID IT COME FROM?

- R is an open source programming language and software environment for statistical computing.
- R is an implementation of the S programming language.
- R was created by Ross Ihaka and Robert Gentleman in 1992.
- R has over 10,000 packages and a large developer community.
- R is supported by major companies such as Microsoft and Oracle with R server and R Enterprise.
- Microsoft and Oracle have integrated R into their RDBMS.

SOFTWARE NEEDED TO CREATE AN R ENVIRONMENT

- R base software.
- RStudio (which is an IDE environment and is optional).
- The Oracle client (the 64 bit version) that has the OCI libraries.
- Configuration of the Oracle client and the software environment.
- ROracle driver/package which will allow for a fast, native connection to the Oracle database.

THE R BASE SOFTWARE: WHAT IS IT?

- R is a language and environment for statistical computing and visualization.
- You can download the R software from the 'www.r-project.org' website which also has tutorials, forums, documentation and FAQs.
- Next, install it on your PC, the installation is simple and no configuration is needed.
- The installation will create the .RData file which will hold your workspace objects and will be loaded into memory each time R is started.

RSTUDIO IS AN INTERACTIVE DEVELOPMENT ENVIRONMENT

- RStudio is a rapid development environment for the R language.
- RStudio can be downloaded from the ‘rstudio.com’ website, the installation is straightforward with no configuration needed.
- The RStudio web site has many resources and the company is the creator of the Shiny package for interactive web applications.

ORACLE 12C CLIENT FOR CONNECTING THE DATABASE

- The Oracle 12c client 64bit can be downloaded from the oracle.com web site; this will not be a simple install and will require several configuration steps.
- You will need to copy the tnsnames.ora file to the NETWORK/Admin directory underneath the client install directory.
- You may also want to customize the SQL*Plus environment but this step is optional.
- You will need to create two environmental variables to access the Oracle client's OCI libraries.
    1) set OCI_INC=C:\app\OraDt16\product\12.1.0\client_1\oci\include
    2) set OCI_LIB64= C:\app\OraDt16\product\12.1.0\client_1\bin

RORACLE NEEDS TO BE INSTALLED IN R

- Download the ROracle package from the Oracle.com web site.
- This package is a 64bit compiled version ROracle package.
- Extract the package from the ROracle_1.3-1.zip file by running the command you see below in R or RStudio.
  install.packages("C:\\DATATREE_NEW_HOME\\R_ENTERPRISE\\ROracle\\ROracle_1.3-1.zip", repos=NULL)
- This command will install the package in R and allow you to use it.

WHAT IS THE DBI INTERFACE AND WHAT IS RORACLE

- DBI is a database interface definition for communication between R and relational database management systems. This interface was written by Hadley Wickham and Kirill Müller.
- All classes in this package are virtual and need to be extended by the various RDBMS implementations.
- ROracle is a set of Oracle database extensions written by Oracle to interface with R and create a native connection.
- The extensions are in the ROracle package which connects R to the Oracle database.

THE RORACLE NATIVE CONNECTION TO THE ORACLE DATABASE

LET'S PAUSE

64 OR 32 BIT - WHY DO WE CARE

- For statistical programming it does matter if you are 32 or 64 bit.
- In terms of memory space utilization 32 bit allows you to access only 3-4 GB of memory.
- Another reason is that if the Oracle client is not 64bit your installation will fail and you will get the error below.

Loading required package: DBI

Error in inDL(x, as.logical(local), as.logical(now), ...) :

unable to load shared object 'C:/Users/jdeacre/Documents/R/win-library/3.3/ROracle/libs/x64/ROracle.dll':

LoadLibrary failure: %1 is not a valid Win32 application.

THE RORACLE NATIVE CONNECTION TO THE ORACLE DATABASE

- RStudio/ROracle native connection to the database is the same as SQL*Plus or SQL Developer in terms of performance.
- In the tests that I ran using identical queries that ran either through SQL Developer, SQL*Plus or RStudio all ran for about the same time some times Rstudio was a bit faster.
- Queries that returned results to RStudio ran for about the same time.
- Queries that returned table contents to the R workspace ran ten times faster than queries that read a table and wrote to another table within Oracle.
- This is correct since writing to the R workspace is a memory operation and it is not writing to a mechanical disk.

THE ALTERNATIVES TO THE RORACLE PACKAGE

- RODBC 2.5 times slower than ROracle in reading from the database.
- RJDBC 79 times slower than ROracle in reading from the database.
- ROracle is faster in reading from the database than either of these methods.
- For reading data across a range of 1000 to 1 million rows, and 10 to 1000 columns.

THE ALTERNATIVES TO THE RORACLE PACKAGE CONTINUED

- RODBC 61 times slower than ROracle in writing to the database.
- RJDBC 630 times slower than ROracle in writing to the database.
- ROracle creates a native connection to the Oracle database and is as fast as SQL Developer or SQL*Plus.
- There is no reason to choose anything but ROracle and this package is maintained by Oracle Corporation.

THE .RDATA WORKSPACE STORAGE FILE

- The .RData file is a binary file that on startup of R is loaded into memory and then on exit from the system is written to disk.
- The .RData file uses gzip to compress any data.frames that are written to it.
- The .RData file can also be organized into a hierarchically-organized set of R workspaces, each corresponding to a directory.
- The mvbutils package has tools like the cd () function which allows you to set up and move through a hierarchically-organized directory tree.

THE .RDATA WORKSPACE STORAGE FILE CONTINUED

- If you store lots of data in the .RData file you will take a few minutes to come up. The save.image() command will also take some time to save the workspace too.
- Why is the .RData file 10X faster than the Oracle database ?
- When you write to the workspace as I did you are performing a memory operation from Oracle to the memory held by the .RData file.

THE .RDATA WORKSPACE STORAGE FILE CONTINUED

- In our test this does not vary we tested data sets ranging from 433780 to 24 million rows and the results were the same in each case 10X faster.
- Until saved or on exit from Rstudio the file will remain the same size, when the save.image() command is executed on exit or manually you will see the change in it's size.
- The reason for this is that it uses gzip to compress data.frames copied to it in memory.

BENCHMARKS TESTS THAT I CREATED

- I created a series of simple tests that would query the database read a table and return the results to the client.
- Some would do calculations and return an aggregate result back.
- Others would write a data.frame from RStudio to Oracle.
- Some would read a table from Oracle and write it to the .RData file.
- I tested connecting to both a container and a pluggable database.

BENCHMARK RESULTS FOR .RDATA FILE

- Test results for the yellow_taxi_trip_june table which were written to a csv file, a data.frame in the .RData file and to an Oracle table, the results are below.

	Size	Rows	Read Time	Filename	Compression
Unzip'd	2.3976 GB	12332379		yellow_taxi_trip_june_bk.csv	0
Zip'd	338.798 MB	12332379	16.51 sec	yellow_taxi_trip_june_bk.7z	85.869%
.RData	417.4381 MB	12332379	4 sec	yellow_taxi_trip_june_bk	82.589%
Oracle	3649MB	12332379	30.222 sec	yellow_taxi_trip_june_bk	0

BENCHMARK RESULTS FOR .RDATA FILE

- Test results for the RESTAURANT table which were written also to a csv file, a data.frame in the .RData file and to an Oracle table, the results are below.

	Size	Rows	Read Time	Filename	Compression
Unzip'd	163,841,355 Bytes	433790		RESTAURANT_V7.csv	0
Zip'd	7,856,080 Bytes	433790	27 sec	RESTAURANT_V7.7z	95.205%
.RData	16903.59 KB	433790		RESTAURANT	89.44 %
Oracle	376MB	433790	Sec	RESTAURANT	

CONNECTING TO AN ORACLE DATABASE

LET'S PAUSE

BENCHMARKS PROC.TIME()

- I used the proc.time() function, this function determines how much real CPU time in seconds the currently running R process has already taken.
- The proc.time function returns five elements, but its print method prints a named vector of length 3.
- The first two entries are the total user and system CPU times of the current R process and any child processes on which it has waited, and the third entry is the ‘real’ elapsed time since the process was started.

BENCHMARKS PROC.TIME() CONTINUED

- The proc.time function is used as shown below:

```
> ptm <- proc.time()
> rs <- dbSendQuery(con, "SELECT EXTRACT(MONTH FROM TPEP_PICKUP_DATETIME ) AS TMONTH, COUNT(*) FROM SEARSTGI_ADMIN.YELLOW_TAXI_TRIP GROUP BY EXTRACT(MONTH FROM TPEP_PICKUP_DATETIME )")
> proc.time() - ptm
   user  system elapsed
   0.01    0.02    0.04
```

CONNECTING TO AN ORACLE DATABASE

- This is how you can connect to an Oracle pluggable database.
- library('ROracle')
- drv <- dbDriver("Oracle")
- con <- dbConnect(drv, "searstgi_admin", “PASSWORD", dbname='//localhost:1521/pdt16tst', prefetch = FALSE, bulk_read = 1000L, bulk_write = 1000L, stmt_cache = 0L, external_credentials = FALSE, sysdba = FALSE)

CONNECTING TO AN ORACLE DATABASE

- This is how you can connect to an Oracle container database.

```
library('ROracle')
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "system", "PASSWORD", dbname='data16pr')
```

BENCHMARK RESULTS CONTINUED

- This query returns an aggregate result of six rows which is a row count by month.
- The statement in RStudio is: rs <- dbSendQuery(con, "SELECT EXTRACT(MONTH FROM TPEP_PICKUP_DATETIME) AS TMONTH, COUNT(*) FROM SEARSTGI_ADMIN.YELLOW_TAXI_TRIP GROUP BY EXTRACT(MONTH FROM TPEP_PICKUP_DATETIME)")
- Oracle returned 'All Rows Fetched: 6 in 220.094 seconds'.
- RStudio returned all the rows in 198.38 seconds.

BENCHMARK RESULTS CONTINUED

- The query returns a count of how many rows are in the table.
- The statement in SQL Developer is:

SELECT COUNT(*) FROM SEARSTGI_ADMIN.YELLOW_TAXI_JUNE;

All Rows Fetched: 1 in 56.187 seconds

24664760

RStudio returned # rows in 53.92 seconds.

BENCHMARK RESULTS CONTINUED

- This query returns a count of how many rows are in the table .
- The statement in RStudio is:

```
>rs <- dbSendQuery(con, "SELECT COUNT(*) FROM SEARSTGI_ADMIN.YELLOW_TAXI_JUNE")
> data <- fetch(rs)    ## extract all rows
user  system elapsed
   0.00    0.00   53.48
COUNT(*)
1 24664760
```

BENCHMARK RESULTS CONTINUED

- This query creates a table on the Oracle server.
- The statement in SQL Developer is:

```
CREATE TABLE RESTAURANT_CP3
AS
SELECT * FROM SEARSTGI_ADMIN.RESTAURANT;
```

Task completed in 60.545 seconds

BENCHMARK RESULTS CONTINUED

- This query creates a table on the Oracle server. The statement in Rstudio is:

```
> ptm <- proc.time()
> dbWriteTable(con, "YELLOW_TAXI_JUNE", allrecs, overwrite = FALSE, append=TRUE, row.names = F, schema="SEARSTGI_ADMIN")
[1] TRUE
> proc.time() - ptm
   user  system elapsed
 51.35    9.53  816.10
```

BENCHMARK RESULTS CONTINUED

- These statements return a count of how many rows are in the table.
- Statement in Rstudio:

```
rs <- dbSendQuery(con, "select * from searstgi_admin.restaurant")
 Data8 <- fetch(rs) ## extract all rows
proc.time() - ptm
  user  system elapsed
  2.14    0.30    4.20
```

BENCHMARK TESTS CONCLUSIONS

- My conclusions are these:
  - When a query extracts data from the Oracle database to .Rdata workspace that operation will be 10X faster than the reading from an Oracle table and writing to another one or a CTAS operation.
  - When performing operations that just return a result the execution times are very similar as you saw in the various examples.
  - The RStudio/ROracle interface gives the R users a fast connection to the database and equal performance with Oracle tools.

USEFUL WEB SITES

- Web site for Plot.ly Analyze and visualize https://plot.ly/r/bubble-maps/
- Web site for R http://stat.ethz.ch
- Web site for statistics https://www.gapminder.org/data/
- Web site for Shiny package https://shiny.rstudio.com/
- Web site for R blogging https://www.r-bloggers.com
- Web site for world data http://data.worldbank.org/indicator/NY.GDP.PCAP.PP.CD
- NYC Open Data web site https://data.cityofnewyork.us/Transportation/2015-Yellow-Taxi-Trip-Data/ba8s-jw6u

HOW TO REACH ME

- My email: jdearce1@gmail.com
- The data sets used are from the NYC Open Data web site if you wish I have descriptions and URLs for each, just email me and I will send them to you.
- For the listings of SQL and R statements, just email me and I will send them to you also.