Paper 198-2010

Data Extraction from ONCORE® Using SAS

Wing-Keung Chiu

UNC's Lineberger Comprehensive Cancer Center, Chapel Hill, NC

Abstract

This paper will present a method to extract data from the Oracle based ONCORE database system using SAS. We will examine in detail the necessary SAS code to execute such a procedure. Pros and Cons will be discussed. This paper is of interest to any data analysts (SAS programmers, statisticians, etc.) who are interested in the timely and accurate retrieval of important data stored in ONCORE. This may be particularly important for data that may not be accessible through the BioStat Console, and therefore must be retrieved from ONCORE's 'back end'. Readers are expected to have an intermediate working knowledge in SAS and SQL languages.

Introduction

The use of Online Collaborative Research Environment (ONCORE) as a database and clinical research system has been gaining in popularity in academic clinical and translation cancer research centers in recent years. This software application provides a common working platform for all important members of a research team; from the principle investigators, through clinical research nurses and associates, statisticians, and data management personnel, including vital personnel from regulatory and finance departments, who also need to work with important study data in this 'on-line' environment.

Data analysts in particular have the frequent need to extract data that when analyzed, provides important information that principle investigators may need to report. For example, toxicity summary tables may be needed for treatment team meetings, or to report to a Data and Safety Monitoring Committee. However, some data, with the way it is currently stored in ONCORE may not be either easily found or correctly formatted. Therefore it may not be easily extractable from the Biostat Console export facility. Some important data may not even be available for extraction from the Biostat Console. This exporting feature in ONCORE may not be the best tool for data analysts who may frequently need to extract data and provide results. And while the Biostat Console may deliver a 'user-friendly' interface, the format data that is extracted from this console is restricted to either Excel® spreadsheets, or comma-separated text files. This method then requires an extra step to import this file from the Biostat Console into a SAS dataset. This, of course, means that there is another opportunity to introduce errors or other problems.

Our solution is to use SAS code to directly access the Oracle based back end, to get directly to the data of interest, and extract it directly into SAS software for data management and analysis, thereby greatly reducing the possible errors by multiple conversions, done in a 'point and click fashion, which can be both time consuming and prone to error. In this paper we will show a method to easily extract data of interest from the back end of the Oracle based ONCORE system using SAS.

Data export in ONCORE

The Biostat Console in ONCORE provides a data export interface for data analyst to export clinical data (appendix 1). The desired protocol can be chosen followed by a selection of electronic Case Report Forms (eCRFs) in a checkbox menu. A dropdown selection box is available to pick the preferred data format either in Excel or comma-separated text file. When the export button is entered, multiple eCRFs may be exported in a single (often zipped or compressed) file.

The design of a menu driven interface and checkbox selection makes this process simple and user-friendly. However, data analysts are often more interested in having a quick, simple, accurate, and 'programmatic one step' method to extract their data directly into their statistical analysis software package. Of course, we choose SAS!

The SAS approach

The SAS/ACCESS package provides an interface to work with various Database Management Systems (DBMS). It allows data access and interaction between SAS and third-party relational databases. It enables data to be read, written to and updated, regardless of its native database platform. Common DBMSs that are supported by SAS/ACCESS include DB2, Informix, Microsoft SQL, MySQL, ODBC, OLE-DB, Oracle, Sysbase and Teradata.

Depending on your license conditions, this package may or may not come with your SAS installation. A separate license may be required to execute SAS/ACCESS successfully. You can run the PROC SETINIT procedure to examine the current license condition. A portion of output which appears in SAS log is as follow.

```
PROC SETINIT; RUN;
```

```
Product expiration dates:
--Base Product                        31DEC2010
--SAS/STAT                            31DEC2010
--SAS/GRAPH                           31DEC2010
--SAS/ACCESS Interface to DB2         31DEC2010
--SAS/ACCESS Interface to ORACLE      31DEC2010
--SAS/ACCESS Interface to SYBASE      31DEC2010
--SAS/ACCESS Interface to PC Files    31DEC2010
--SAS/ACCESS Interface to ODBC        31DEC2010
--SAS/ACCESS Interface to OLE DB      31DEC2010
--SAS/ACCESS Interface to Teradata    31DEC2010
--SAS/ACCESS Interface to MYSQL       31DEC2010
--SAS/ACCESS Interface to Netezza     31DEC2010
--SAS/ACCESS Reserved Slot 551        31DEC2010
```

There are at least two ways to execute SAS/ACCESS connecting to an Oracle DMBS. They are:

1. Oracle interface in LIBNAME statement
2. Pass-through in SQL

Oracle interface in LIBNAME statement

The LIBNAME statement is generally used to associate a directory to an explicit directory that is different from the default 'Work' library. An alias is designated by the user to reference the path location. Once assigned, it can be used as a library name for the target location.

Beginning in SAS version 8, the Oracle interface was introduced in the LIBNAME statement. To activate the Oracle engine, you first specify an alias name for the DBMS, followed by a DBMS keyword, plus additional connection settings that include login ID, password, path name, and name of schema. Some generic code is shown below:

```
LIBNAME mylib ORACLE
              USER     = user_id
              PASSWORD = user_key
              Path     = 'oracle_name'
              SCHEMA   = schema_name;
```

Notes:

mylib	:- alias of the Oracle database
ORACLE	:- keyword to invoke Oracle
user_id	:- user login
user_key	:- password
oracle_name	:- reference name of the Oracle drive
schema_name	:- name of public schema

When the above syntax is executed, a connection to Oracle is immediately established. The log below prompts a successful connection via the LIBNAME statement.

```
NOTE: SAS initialization used:
      real time           2.48 seconds
      cpu time            1.34 seconds

1    LIBNAME mylib Oracle schema=schema_name
     user=user_id password=XXXXXXXXXXXXX
     path="oracle_name";

Note: Libref mylib was successfully assigned as
follows:
Engine: Oracle
Physical Name: oracle_name
```

Oracle database files can now be found in the user-defined SAS library (or mylib) and will be treated like a regular SAS dataset file. A two-level name (i.e. mylib.filename) is used to store and retrieve data in the Data step and SAS procedures. For example:

```
PROC CONTENTS data=mylib.toxicity;
RUN;
```

You can add SAS system options to get a better understanding of the information being passed to Oracle.

```
OPTIONS SASTRACE=',,,d' SASTRACELOC=saslog
NOSTSUFFIX;
```

The SASTRACE= option is useful to trace the information to and from the DBMS. The ',,,d' syntax specifies that all the SQL commands sent to Oracle be written to the log. In PC SAS, you must also include SASTRACELOC= option when using SASTRACE=. These records can be outputted to a SAS log by SASTRACELOC=SASLOG or to a log file by specifying SASTRACELOC=file'C:\SAS\SASlog.log'. You can make the log more readable by including the NOSTSUFFIX options.

To end the Oracle DBMS connection, use the CLEAR option in the LIBNAME statement with the pre-defined alias.

```
LIBNAME mylib CLEAR;
```

Pass-through in SQL

The pass-through facility (pass-thru) in PROC SQL enables you to send specific SQL statements directly to Oracle for execution. With SAS/ACCESS, the pass-thru interface creates a connection to Oracle that allows you to access and retrieve DBMS data files in a single procedure step.

The three major components that execute the pass-thru facilities are illustrated below, they consist of a CONNECT statement, a CONNECTTION TO statement and a DISCONNECT statement.

The CONNECT statement lets you specify values for arguments, including login ID, password and path name, connecting to the Oracle database. The CONNECTTION TO module in the FROM clause of the SELECT statement helps to retrieve data from the database. A DBMS specific query is sent to Oracle by another SELECT statement following the CONNECTION TO module. This specific query code can take native Oracle languages that execute directly in the DBMS and return the results to SAS. Lastly, the Oracle connection is terminated by a DISCONNECT statement at the end of the procedure.

To capture an error message log, you can use the automatic macro variables, SQLXRC and SQLXMSG, with the %PUT macro. The former one returns code that identifies the data source error, whereas the later one writes out descriptive information about the error.

```
PROC SQL;
CONNECT TO ORACLE (user = user_id
                   pass = user_key
                   path = 'oracle_name' );
CREATE TABLE mytable AS

SELECT * FROM CONNECTION TO ORACLE
 (SELECT * FROM schema_name.table_name);

%PUT &SQLXMSG;

DISCONNECT FROM ORACLE;
QUIT;
```

Notes:

ORACLE	:- keyword to invoke Oracle interface
user_id	:- user login
user_key	:- password
oracle_name	:- reference name of the Oracle drive
schema_name	:- name of public schema

A valuable function of using the pass-thru facilities is the capability to execute a comprehensive syntax in PROC SQL. General data manipulation techniques like sorting, formatting, sub-setting, etc., can be easily integrated into the code. In addition, the SQL procedure supports more sophisticated features such as sub-querying, summary functions, and tables joining. These all can be embedded into the programming code to generate a well-defined dataset or summary table in a few procedural steps.

PROC SQL code is demonstrated below to: (1) establish a connection to the Oracle back end view in ONCORE, (2) create a view of the toxicity forms and (3) query the subjects that have toxicity grades greater than or equal to three in Neutropenia.

```
PROC SQL;
CONNECT TO ORACLE
            (user = user_id
             pass = user_key
             path = 'oracle_name');

CREATE VIEW myview AS

SELECT
  SeqNum as SN 'ID',
  Onset as OSD 'Onset Date' format mmddyy10.,
  AdvEvent as AE 'Adverse Event',
  Toxicity as Tox 'Toxicity',
  ToxGrade as Grade 'Tox. Grade'

FROM CONNECTION TO ORACLE
 (SELECT * FROM mylib.toxicity
  WHERE protocol_no='LCCC1234');

%PUT &SQLXMSG;

DISCONNECT FROM ORACLE;

SELECT DISTINCT Ae, Sn, Osd, Grade
FROM myview
WHERE (Grade GE 3) and (Ae='Neutropenia')
GROUP BY Sn;
QUIT;
```

The Oracle connection is first configured, followed by the selection of variables of interest, simultaneously renaming the variables and attaching the desired formats and labels. Since toxicity forms can comprise records from various protocol studies, the table is 'sub-setted' to only include the protocol of interest. Lastly, a query statement is issued to list the subjects that have grade three (or higher) toxicities in Neutropenia.

Evaluation

Oracle back end views in ONCORE provide the data analyst the ability to access data directly by using SAS and SAS/ACCESS. Two ways of implementing SAS/ACCESS were discussed above. The method using the Oracle engine in the LIBNAME statement is considered a simple and straight-forward method to create a connection linkage. On the other hand, the pass-thru facilities in PROC SQL exhibit a greater flexibility to condense complex coding in the procedure steps.

We consider the strength of establishing a direct connection through SAS to retrieve data in ONCORE as crucial. Using the Biostat Console to extract data in an Excel spreadsheet or a text file data, and then using something like PROC IMPORT to move this data again, may lend itself to errors and can be considered not optimal. Our direct connection method allows the data analyst to save a substantial amount of time and effort by not adding an unnecessary (and possibly error-prone) step. The connection itself is rapid, safe and secure. Also, quick and easy tables produced by SAS allow study data to be easily reviewed by others in the research team for quality control and other related issues.

Depending on your site license status, you may need to purchase an additional license in order to activate the SAS/ACCESS package. An initial, but one-time, setup is required on each computer to make sure that the Windows® environment is ready for the DMBS connection.

A successful connection to ONCORE not only relies on SAS/ACCESS and the right SAS code, but also on the configuration of your computer. For example the Oracle® client software has to be properly installed and the ODBC driver connections must have the right settings. You will need to check with your ONCORE administrator (or technical support) to get a valid login ID, password, and other needed Oracle settings (for PATH= and SCHEMA= parameters in SAS coding) to make it work for you.

Conclusion

ONCORE is a powerful clinical database platform that is designed to meet the needs of the multiple members of a research team. Making SAS 'talk directly' with ONCORE may be the optimal method of extracting data for the data analyst.

References

SAS® Institute Inc., 'SAS/ACCESS® 9.2 for Relational Database: References', Cary, NC: SAS® Institute., 2009

Chapman, Sridharma (2005), 'Using the Oracle Libname Engine to reduce the time it takes to extract data from Oracle database', Proceedings of the Eighteen Annual NorthEast SAS® User Group conference

Levin (2004), 'Methods of Storing SAS Data into Oracle Tables', Proceedings of the Twenty-Nine Annual SAS® User Group International conference, 29.

Prattter (2002), 'Access to Relational Database using SAS', Proceedings of the Annual Pharmaceutical Industry SAS® User Group conference.

Acknowledgements

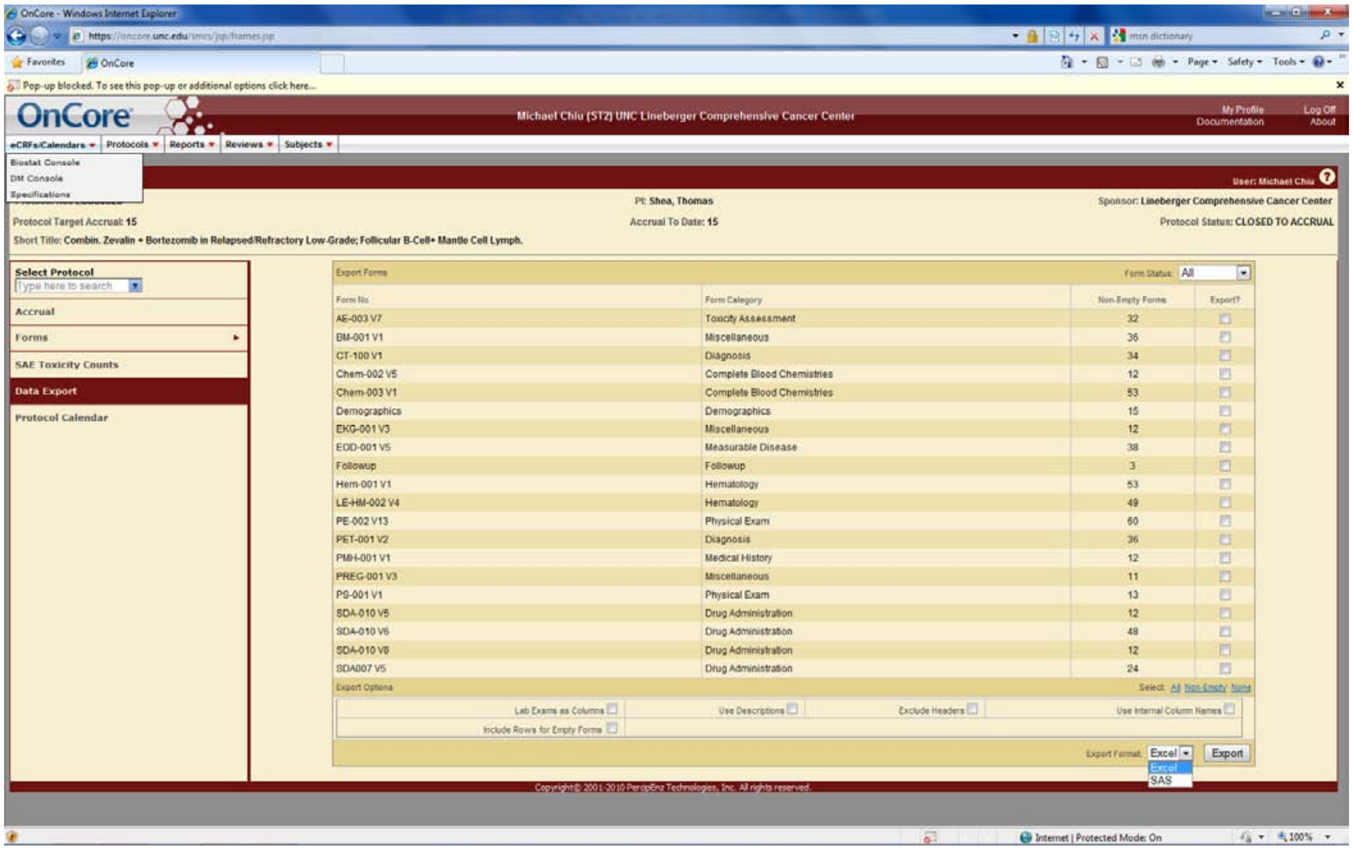
I want to thank Dominic Moore and Paul Jones for their input and review of this paper.

Contact Information

Wing-Keung Chiu
UNC Lineberger Comprehensive Cancer Center
20-024 Administrative Tower, CB#7295
Chapel Hill, NC 27599-7295
Phone: (919) 843-1534
Email: chiumich@med.unc.edu

Trademarks

Appendix 1: ONOCRE provides data export interface under the Biostat Console



PI: Shea, Thomas

Sponsor: Lineberger Comprehensive Cancer Center

Protocol Target Accrual: 15

Accrual To Date: 15

Protocol Status: CLOSED TO ACCRUAL

Short Title: Combin. Zevalin + Bortezomib in Relapsed/Refractory Low-Grade; Follicular B-Cell+ Mantle Cell Lymph.

Export Forms — Form Status: All

Form No	Form Category	Non-Empty Forms	Export?
AE-003 V7	Toxicity Assessment	32	☐
BM-001 V1	Miscellaneous	36	☐
CT-100 V1	Diagnosis	34	☐
Chem-002 V5	Complete Blood Chemistries	12	☐
Chem-003 V1	Complete Blood Chemistries	53	☐
Demographics	Demographics	15	☐
EKG-001 V3	Miscellaneous	12	☐
EOD-001 V5	Measurable Disease	38	☐
Followup	Followup	3	☐
Hem-001 V1	Hematology	53	☐
LE-HM-002 V4	Hematology	49	☐
PE-002 V13	Physical Exam	60	☐
PET-001 V2	Diagnosis	36	☐
PMH-001 V1	Medical History	12	☐
PREG-001 V3	Miscellaneous	11	☐
PS-001 V1	Physical Exam	13	☐
SDA-010 V5	Drug Administration	12	☐
SDA-010 V6	Drug Administration	48	☐
SDA-010 V8	Drug Administration	12	☐
SDA007 V5	Drug Administration	24	☐

Export Options — Select: All Non-Empty None

Lab Exams as Columns ☐ Use Descriptions ☐ Exclude Headers ☐ Use Internal Column Names ☐

Include Rows for Empty Forms ☐

Export Format: Excel (Excel, SAS) Export

Copyright© 2001-2010 PercipEnz Technologies, Inc. All rights reserved.