irutils: An R Package for Institutional Research

Jason M. Bryer
jbryer@expcelsior.edu
Excelsior College

October 28, 2011

Abstract

This document specifies the functions available in the irutils package. This package is designed to accomony the *Introduction to R and LaTeX* document. The package provides many functions useful for typical reporting requirements of Institutional Research offices.

Keywords: *institutional research, database, higher education, ipeds*

1 Introduction

Note: Sections in italics represent portions of the document that will only be available to Excelsior College staff.

Load the ecir package from the O drive. Using the lib.loc option will load the latest version directly from the O drive without the need of installing it first.

```
> library(ecir, lib.loc = "O:/Resources/R/library")
```

2 Database Access

For many Institutional Research offices the institutions student information system (SIS) is the most common source of data. Since virtually all SIS systems are backed by a database, extracting data requires extracting data using queries. Typically the language used to extract data is called structured query language (SQL) regardless if the database is provided by Oracle, Microsoft, or an open source options such as MySQL and PostgreSQL. There are a number of functions in this package that will faciliate extracting data from these databases directly into R.

The database access functions provide an interface to a directory of SQL scripts. SQL scripts are simply a plain text file containing the query. The directory containing these files can be determined or set using the getSQLRepos and setSQLRepos functions, repsectively.

```
> getSQLRepos()

[1] "O:/Resources/R Packages/ecir/data"
```

```
> setSQLRepos("O:/Resources/R Packages/ecir/data")
```

By convention, all SQL files must use a `.sql` file extension. The `getQueries` function will return a list of all the queries available in the current reposistory.

```
> getQueries()

 [1] "CourseCancelations"                "EnrolledStudents"
 [3] "graduates"                         "GraduatesWithinRange"
 [5] "PendingGrades"                     "TransferCredit"
 [7] "WarehouseData"                     "WarehouseDataWithCredits"
 [9] "WarehouseDataWithCreditsNoINLCCS" "WarehouseDataWithDemographics"
[11] "warehouseSummary"
```

The `getQueryDesc` and `getParameters` functions will provide some details about the query in question. In particular, the latter will return the parameters that are required for the query to execute.

```
> getQueryDesc("GraduatesWithinRange")

[1] " This function returns all graduates in the given date range."

> getParameters("GraduatesWithinRange")

[1] "endDate"   "startDate"
```

There are two functions available for executing the query. The `execQuery` will execute the query and return a data frame. The `cacheQuery` however, will first look in the specified directory (by default the `dir` parameter is set to `getwd()`) for a CSV file that matches the currently request query. That is, the file name (which is returned when this function is executed) is built using a combination of the query name and parameters to uniquely identify it. This is useful when using Sweave and LaTeXfor document preparation where the function may be executed multiple times but the data does not change. It is considerably faster to read data from a flat file then it is to query the database each time.

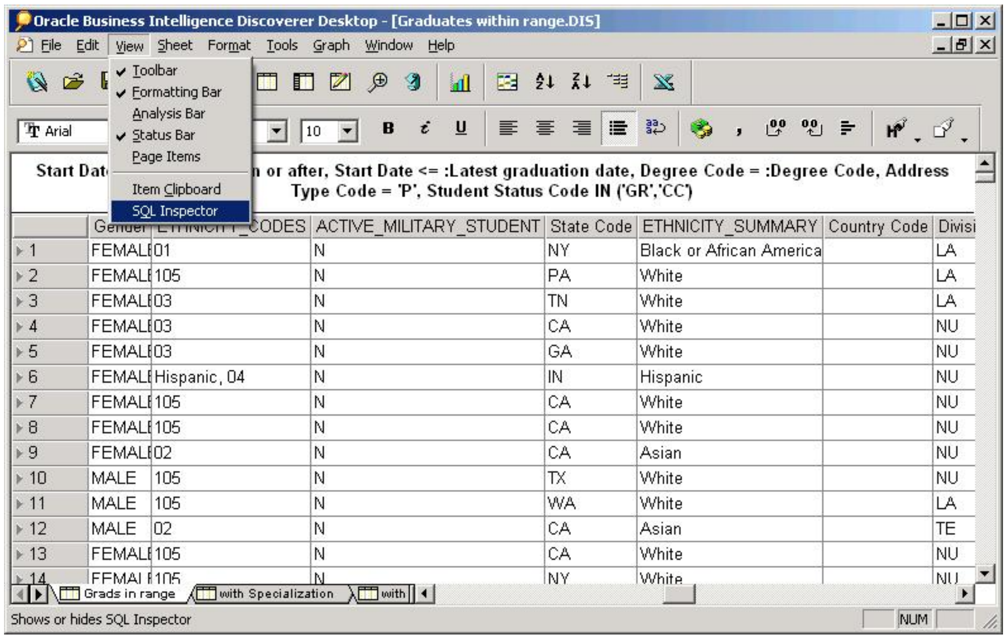
```
> graduates = execQuery("GraduatesWithinRange", startDate = "01-JUL-2010",
+     endDate = "30-JUN-2011")

> graduates = cacheQuery("GraduatesWithinRange", startDate = "01-JUL-2010",
+     endDate = "30-JUN-2011")

[1] "Cahced query file: C:/Temp/GraduatesWithinRange.endDate.30-JUN-2011.startDate.01-JUL-2010.csv
```

2.1 Creating Your Own Query

To create your own query, simply place the SQL statement in its own text file ending with `.sql`. Comments can be placed in the file using the `#` symbol. Placing informative comments at the beginning of the file will be useful since the `getQueryDesc` function will return these comments to the user. Parameters can be placed anywhere in the file and must be enclosed


Oracle Business Intelligence Discoverer Desktop - [Graduates within range.DIS]
File Edit View Sheet Format Tools Graph Window Help
✓ Toolbar
✓ Formatting Bar
Analysis Bar
✓ Status Bar
Page Items
Item Clipboard
SQL Inspector
Arial
10
Grads in range
with Specialization
with
Shows or hides SQL Inspector
NUM

Start Dat… n or after, Start Date <= :Latest graduation date, Degree Code = :Degree Code, Address Type Code = 'P', Student Status Code IN ('GR','CC')

	Gender	ETHNICITY_CODES	ACTIVE_MILITARY_STUDENT	State Code	ETHNICITY_SUMMARY	Country Code	Divisi
1	FEMAL	01	N	NY	Black or African America		LA
2	FEMAL	105	N	PA	White		LA
3	FEMAL	03	N	TN	White		LA
4	FEMAL	03	N	CA	White		NU
5	FEMAL	03	N	GA	White		NU
6	FEMAL	Hispanic, 04	N	IN	Hispanic		NU
7	FEMAL	105	N	CA	White		NU
8	FEMAL	105	N	CA	White		NU
9	FEMAL	02	N	CA	Asian		NU
10	MALE	105	N	TX	White		NU
11	MALE	105	N	WA	White		LA
12	MALE	02	N	CA	Asian		TE
13	FEMAL	105	N	CA	White		NU
14	FEMAL	105	N	NY	White		NU

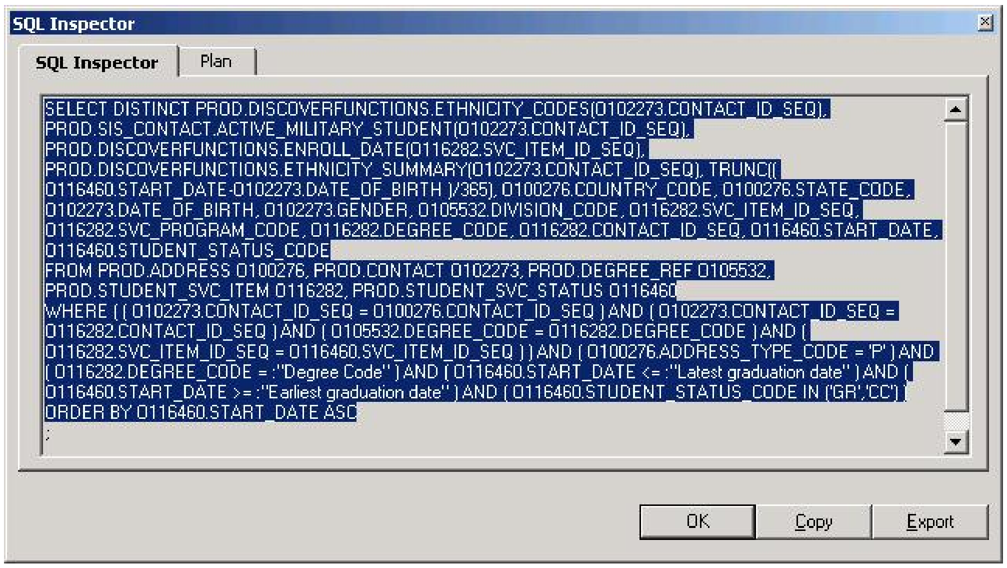
Figure 1: Beginning 72-Month Completion Rates by Military Status

within colons (i.e. the : character). Additionally, parameter names must begin with a letter, contain only letters and numbers, and cannot have any spaces.

At Excelsior College, Oracle Discoverer is the most common approach to extracting data from SIS. Once a desired report is created using Discover, the SQL statement can be viewed using the "View" menu, then "SQL Inspector" (see Figure 1). Figure 2 represents the resulting SQL Inspector view. Copy and paste the query (excluding the trailing semicolon) to a new text document and save it with an appropriate name. Parameters will need to be changed according to the rules described above. It is useful if a comment is placed at the top of the file describing the query and table names be changed from the numeric representation in the FROM claus to more meaningful names. It should also be noted that Oracle functions will not be automatically aliased. That is, the full function name will be used as the variable name. The final query (GraduatesWithinRange) represented in Figure 2 is provided below.

```
> query = getQuery("GraduatesWithinRange")
> strwrap(query, width = 80, exdent = 5)
```

```
 [1] "SELECT DISTINCT PROD.DISCOVERFUNCTIONS.ETHNICITY_CODES(contact.CONTACT_ID_SEQ)"
 [2] "     ETHNICITY_CODES,"
 [3] "     PROD.SIS_CONTACT.ACTIVE_MILITARY_STUDENT(contact.CONTACT_ID_SEQ)"
 [4] "     ACTIVE_MILITARY_STUDENT,"
 [5] "     PROD.DISCOVERFUNCTIONS.ENROLL_DATE(studentSvcItem.SVC_ITEM_ID_SEQ)"
```



```
SELECT DISTINCT PROD.DISCOVERFUNCTIONS.ETHNICITY_CODES(O102273.CONTACT_ID_SEQ),
PROD.SIS_CONTACT.ACTIVE_MILITARY_STUDENT(O102273.CONTACT_ID_SEQ),
PROD.DISCOVERFUNCTIONS.ENROLL_DATE(O116282.SVC_ITEM_ID_SEQ),
PROD.DISCOVERFUNCTIONS.ETHNICITY_SUMMARY(O102273.CONTACT_ID_SEQ), TRUNC((
O116460.START_DATE-O102273.DATE_OF_BIRTH )/365), O100276.COUNTRY_CODE, O100276.STATE_CODE,
O102273.DATE_OF_BIRTH, O102273.GENDER, O105532.DIVISION_CODE, O116282.SVC_ITEM_ID_SEQ,
O116282.SVC_PROGRAM_CODE, O116282.DEGREE_CODE, O116282.CONTACT_ID_SEQ, O116460.START_DATE,
O116460.STUDENT_STATUS_CODE
FROM PROD.ADDRESS O100276, PROD.CONTACT O102273, PROD.DEGREE_REF O105532,
PROD.STUDENT_SVC_ITEM O116282, PROD.STUDENT_SVC_STATUS O116460
WHERE ( ( O102273.CONTACT_ID_SEQ = O100276.CONTACT_ID_SEQ ) AND ( O102273.CONTACT_ID_SEQ =
O116282.CONTACT_ID_SEQ ) AND ( O105532.DEGREE_CODE = O116282.DEGREE_CODE ) AND (
O116282.SVC_ITEM_ID_SEQ = O116460.SVC_ITEM_ID_SEQ ) ) AND ( O100276.ADDRESS_TYPE_CODE = 'P' ) AND
( O116282.DEGREE_CODE = :"Degree Code" ) AND ( O116460.START_DATE <= :"Latest graduation date" ) AND (
O116460.START_DATE >= :"Earliest graduation date" ) AND ( O116460.STUDENT_STATUS_CODE IN ('GR','CC') )
ORDER BY O116460.START_DATE ASC
;
```

Figure 2: SQL Inspector

```
 [6] "      ENROLL_DATE,"
 [7] "      PROD.DISCOVERFUNCTIONS.ETHNICITY_SUMMARY(contact.CONTACT_ID_SEQ)"
 [8] "      ETHNICITY_SUMMARY, TRUNC(("
 [9] "      studentSvcStatus.START_DATE-contact.DATE_OF_BIRTH )/365)"
[10] "      AGE_AT_GRADUATION, address.COUNTRY_CODE, address.STATE_CODE,"
[11] "      contact.DATE_OF_BIRTH, contact.GENDER, degreeRef.DIVISION_CODE,"
[12] "      studentSvcItem.SVC_ITEM_ID_SEQ, studentSvcItem.SVC_PROGRAM_CODE,"
[13] "      studentSvcItem.DEGREE_CODE, studentSvcItem.CONTACT_ID_SEQ,"
[14] "      studentSvcStatus.START_DATE, studentSvcStatus.STUDENT_STATUS_CODE FROM"
[15] "      PROD.ADDRESS address, PROD.CONTACT contact, PROD.DEGREE_REF degreeRef,"
[16] "      PROD.STUDENT_SVC_ITEM studentSvcItem, PROD.STUDENT_SVC_STATUS"
[17] "      studentSvcStatus WHERE ( ( contact.CONTACT_ID_SEQ = address.CONTACT_ID_SEQ"
[18] "      ) AND ( contact.CONTACT_ID_SEQ = studentSvcItem.CONTACT_ID_SEQ ) AND ("
[19] "      degreeRef.DEGREE_CODE = studentSvcItem.DEGREE_CODE ) AND ("
[20] "      studentSvcItem.SVC_ITEM_ID_SEQ = studentSvcStatus.SVC_ITEM_ID_SEQ ) ) AND"
[21] "      ( address.ADDRESS_TYPE_CODE = 'P' ) AND ( studentSvcStatus.START_DATE <="
[22] "      ':endDate:' ) AND ( studentSvcStatus.START_DATE >= ':startDate:' ) AND ("
[23] "      studentSvcStatus.STUDENT_STATUS_CODE IN ('GR','CC') ) ORDER BY"
[24] "      studentSvcStatus.START_DATE ASC"
```