

irutils: An R Package for Institutional Research

Jason M. Bryer
jbryer@expcelsior.edu
Excelsior College

November 14, 2011

Abstract

This document specifies the functions available in the `irutils` package. This package is designed to accomony the *Introduction to R and L^AT_EX* 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')
```

The `irutils` package is hosted at <http://github.com/jbryer/irutils>. The `devtools` (Wickham, 2011) package provides functions to assist in R package development including a function to install directly from github. The `install_github` will install the latest development version of the `irutils` package.

```
> library(devtools)
> install_github('devtools', 'jbryer')
```

The currently available functions are:

```
> library(irutils)
> ls('package:irutils')
```

[1] "cacheQuery"	"cohortRetention"
[3] "cohortSummary"	"cohortSummaryLaTeX"
[5] "createLocalRepos"	"execQuery"
[7] "fySummary"	"fySummaryLaTeX"

[9] "getAge"	"getAgeGroups"
[11] "getAgeGroupsIPEDS"	"getParameters"
[13] "getQueries"	"getQuery"
[15] "getQueryDesc"	"getSQLRepos"
[17] "local.available.packages"	"local.install.packages"
[19] "plotBarchartTable"	"plotCohortRetention"
[21] "plotHeatmapTable"	"plotPersistence"
[23] "plotRetention"	"plotRetentionOverall"
[25] "retention"	"retentionSummary"
[27] "retentionSummaryLaTeX"	"saveIPEDStoDB"
[29] "setSQLRepos"	"updateLocalRepos"

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 facilitate 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, respectively.

```
> getSQLRepos()
```

```
[1] "C:/Program Files/R/R-2.14.0/library/irutils/data"
```

```
> setSQLRepos("0:/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 repository.

```
> getQueries()
```

[1] "Alumni"	"CourseCancelations"
[3] "EnrolledStudents"	"graduates"
[5] "GraduatesWithinRange"	"PendingGrades"
[7] "TransferCredit"	"TransferCreditDetail"
[9] "WarehouseData"	"WarehouseDataWithCredits"
[11] "WarehouseDataWithCreditsNoINLCCS"	"WarehouseDataWithDemographics"
[13] "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 L^AT_EX for 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] "Cached query file: C:/Dropbox/My Dropbox/Projects/irutils/inst/doc/GraduatesWithinRange.endDate"
```

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 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 clause 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)"
```

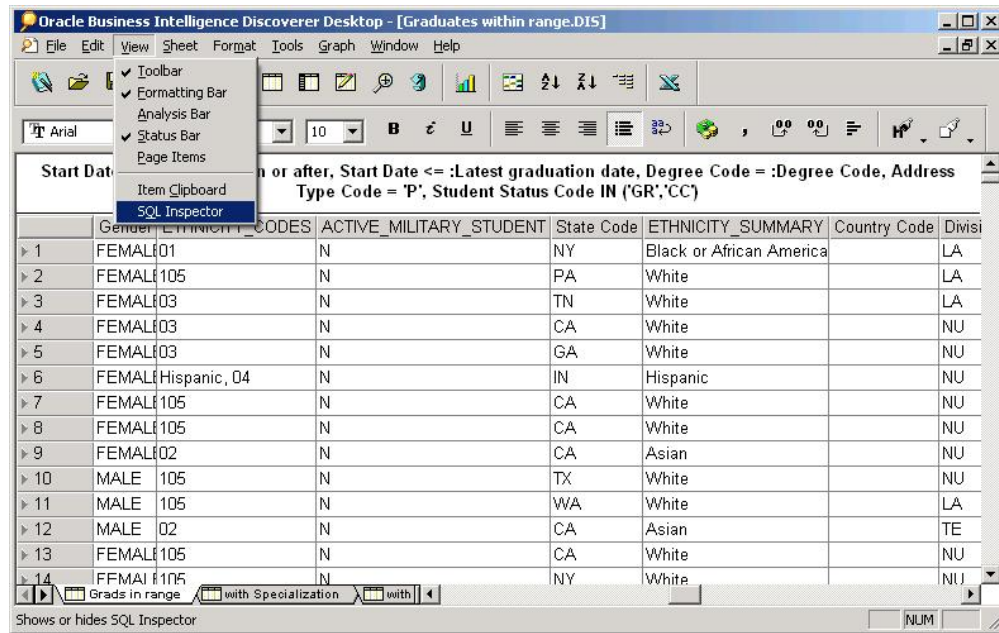


Figure 1: Beginning 72-Month Completion Rates by Military Status

```

[4] "    ACTIVE_MILITARY_STUDENT, "
[5] "    PROD.DISCOVERFUNCTIONS.ENROLL_DATE(studentSvcItem.SVC_ITEM_ID_SEQ)"
[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"

```

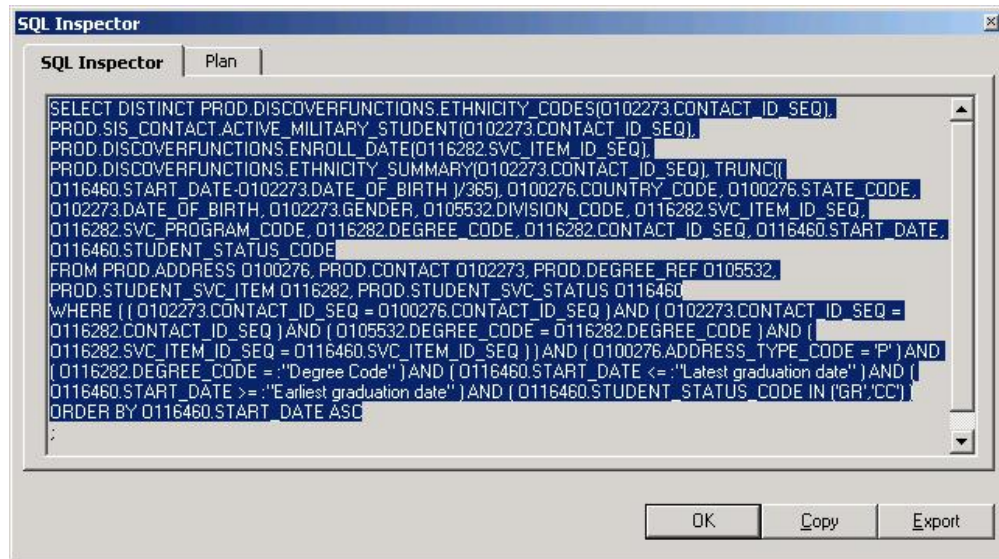


Figure 2: SQL Inspector

[24] " studentSvcStatus.START_DATE ASC"

3 Visualizing Likert Response Items

```
> data(pisa)
> items28 = pisa[,substr(names(pisa), 1,5) == 'ST24Q']
> names(items28) = c("I read only if I have to.",
+   "Reading is one of my favorite hobbies.",
+   "I like talking about books with other people.",
+   "I find it hard to finish books.",
+   "I feel happy if I receive a book as a present.",
+   "For me, reading is a waste of time.",
+   "I enjoy going to a bookstore or a library.",
+   "I read only to get information that I need.",
+   "I cannot sit still and read for more than a few minutes.",
+   "I like to express my opinions about books I have read.",
+   "I like to exchange books with my friends")
> for(i in 1:ncol(items28)) {
+   items28[,i] = factor(items28[,i], levels=1:4,
+     labels=c('Strongly disagree', 'Disagree', 'Agree', 'Strongly Agree'),
+     ordered=TRUE)
+ }
```

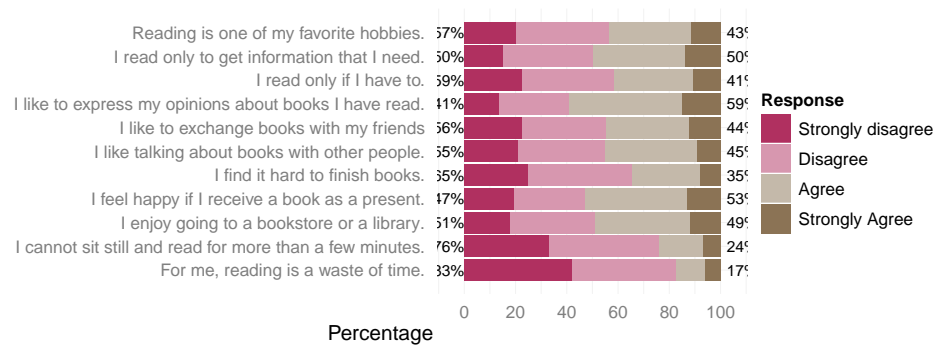


Figure 3: Barchart Table Graphic

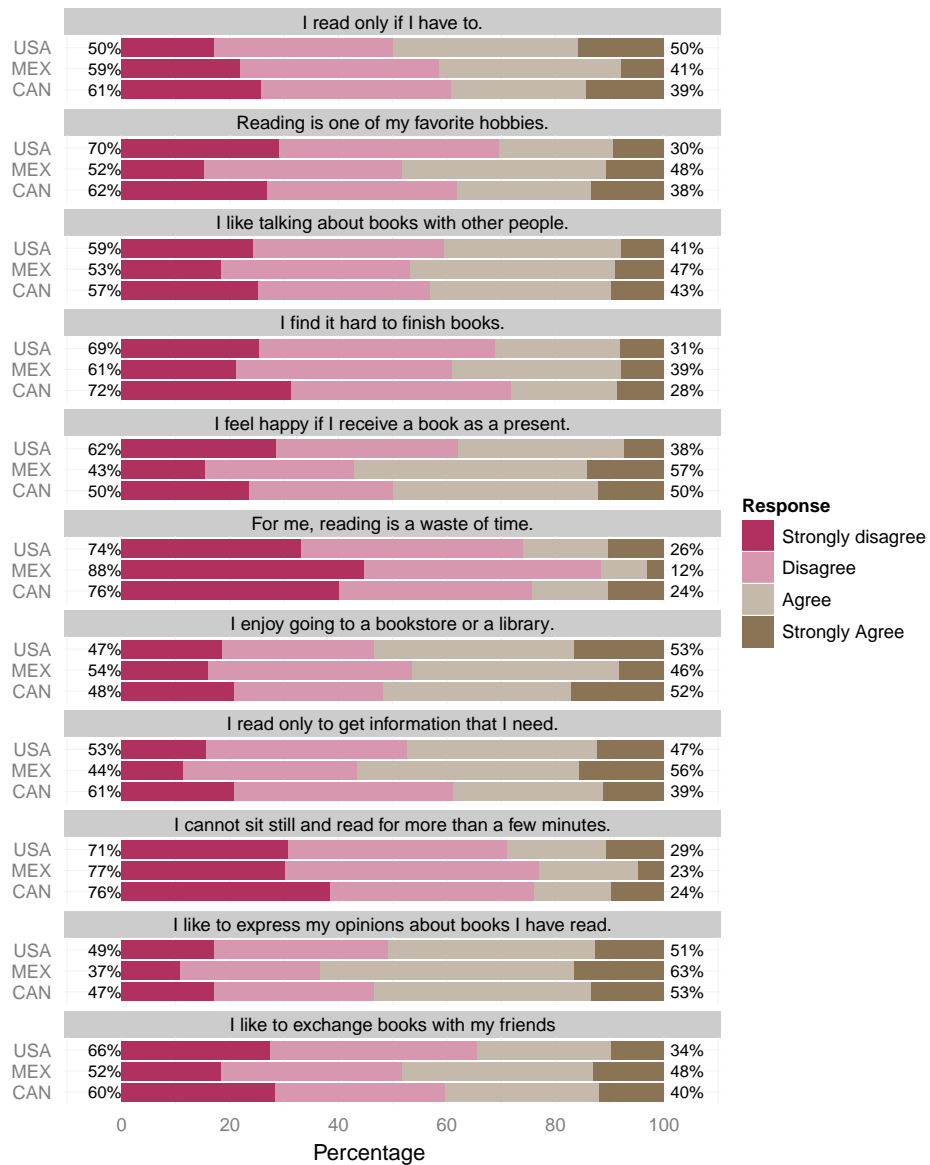


Figure 4: Barchart Table Graphic Grouped by Country

References

Wickham, H. (2011). devtools: Tools to make developing r code easier [Computer software manual]. Available from <http://CRAN.R-project.org/package=devtools> (R package version 0.4)