The R Package for Jackknife Confidentiality Protection Final Report for ESSNet

Jobst Heitzig, December 2009

Introduction

In this part of the project, we developed an open source software package named “condense” for the likewise open source R statistical computing environment (see http://www.r-project.org).

The “condense” package is a reference implementation of basic methods of statistical analysis with included Jackknife confidentiality protection (see J. Heitzig, “The 'Jackknife' method: Confidentiality protection for complex statistical analyses”, Work session on statistical data confidentiality, UNECE, Geneva (2005), http://www.unece.org/stats/documents/ece/ces/ge.46/2005/wp.39.e.pdf). It covers the most important univariate analysis methods such as frequencies, sums, other moment- and quantile-based statistics, Student's t test, robust measures of location, scale and shape, etc., putting some focus on explorative methods.

This work is also related to the more general open source software development project “ConDEnSE” (see J. Heitzig, “ConDEnSE: towards an open source remote analysis system with jackknife confidentiality protection”, 56th Session of the International Statistical Institute, Lisboa (2007), http://condense.sourceforge.net/stcpm02_heitzig.pdf) which plans to use the developed R package in its back-end. In particular, we also use the ConDEnSE code repository for versioning (see http://condense.svn.sourceforge.net/viewvc/condense/)

Quick start guide

Installation

- Install the open source R statistical computing environment with version 2.9.2 or later from http://www.r-project.org
- Download the “condense” package tarball (currently condense_1.0_0.tar.gz) from http://condense.svn.sourceforge.net/viewvc/condense/tags/condense_1.0_0/condense_1.0-0.tar.gz
- On the command line, cd to the location with the downloaded tarball, start R and call `install.packages("condense_1.0_0.tar.gz")`

Test

- Create on localhost a MySQL database named “condense”, a MySQL user named “condense” with the password “condense”, and give the user full permissions for the database.
- In R, call `require(condense)`, then `demo(condense2)`
- If no local MySQL is available, you can use any other MySQL database for which you have enough permissions by connecting to it via
  `mysql.connection <- dbConnect(dbDriver("MySQL"), host="...", user="...", password="...", dbname="...")`
  and then calling `demo(condense2)`. See the documentation for the standard R package

RMySQL for details about dbConnect().

Online help

In R, you get the online help by calling `help(condense)` or `help(jk.means)`.

Concept

The computational statistics community largely uses the open source R statistical computing environment to develop, test, and share new statistical analysis methods and algorithms for its powerful, well-structured and transparent programming language and its interfaces to most common data storage systems (see, e.g., Rizzi, Alfredo and Vichi, Maurizio (Eds.): "COMPSTAT 2006 - Proceedings in Computational Statistics", 17th Symposium Held in Rome, Italy (2006), http://www.springer.com/statistics/computational/book/978-3-7908-1708-9).

In this project, R was chosen as a development framework not only because of these reasons but also to make sure that the resulting software will be available freely to everyone and in the hope to stimulate some further cooperation with members of the computational statistics community.

As R is a functional language, the package consists of a top-level function jk.means and of a number of bottom-level functions containing basic functionality common to this and other analyses that might be implemented later as top-level functions, and for performing the various steps of Jackknife confidentiality protection. While the top-level functions build the interface for both the users of interactive R sessions and for graphical front-ends such as the ConDEnse front-end, the latter set of functions is designed for maximal extensibility to allow contributers of new statistical analysis methods to develop "safe" versions of their methods by combining them with the packages helper functions.

However, functionality and calling syntax of the top-level function were also designed in a certain analogy to the common procedures MEANS and UNIVARIATE of the SAS(R) language, since it was felt that these SAS(R) procedures more or less cover those univariate methods most important for end-users, and since for some them and other SAS(R) procedures, already a prototypical implementation of the Jackknife method had been developed in form of SAS(R) macros.

The basic calling syntax of the top-level functions is exemplified by a the following call to the function "jk.means":

```
jk.means (
        connection = some.mysql.database.connection
        input.table = "person",
        analyse = list (
                list ( name = "age", quartiles = c (0,20,40,60,Inf) ),
                list ( name = "income", quartiles = c (0,1e3,1e4,1e5,1e7) )
        ),
        group.by = list (
                list ( name = "gender", label = "gender", groups = list (
                        list ( level = 1, label = "all",        condition = "TRUE" ),
                        list ( level = 2, label = "female",     condition = "gender='f'" ),
                        list ( level = 2, label = "male",       condition = "gender='m'" )
                ) )
        ),
        where = "income>0",
        secret = "some constant string kept secret from users"
        return.format = "ds",
```

```
        mu0 = 0.0,
        alpha.notch = 0.025
    )
```

All function arguments are given by keyword parameters and might be nested structures using the common R constructs list() and c(). All parameters except the last two are designed to be common with other top-level functions that might be implemented later.

The remaining parameters are specific to one this top-level function.

The return value of a top-level function is usually a multi-dimensional result table, returned as either a nested list of headers and a stream of data cells ("mudit" format), or as a linearized data frame containing a row for each cell of the multi-dimensional result table ("df" format). In the above example, these two return formats would return a four-dimensional result table with 59*2*2*3=708 cells like this:

"mudit" format:

```
    list ( list (
        type = "mudit",
        dims = c (
            "statistics",
            "analysed column",
            "quality",
            "gender"
        ),
        groups = list (
            list (  list ( level = 1, label = "no. of non-null values" ),
                    … [57 other statistics],
                    list ( level = 1, label = "Crow-Siddiqui's kurtosis" ) ),
            list (  list ( level = 1, label = "age" ),
                    list ( level = 1, label = "income" ) ),
            list (  list ( level = 1, label = "approx. value" ),
                    list ( level = 1, label = "approx. protection error" ) ),
            list (  list ( level = 1, label = "all" ),
                    list ( level = 2, label = "female" ),
                    list ( level = 2, label = "male" ) )
        ),
        stream = c ( 123.4, 62.3, 61.2, … [702 more values], 0.0234, 0.0123, 0.0135)
    ) )
```

"df" format:

stat	**label**	**column**	**quality**	**gender**	**result**
n	no. of non-null values	age	approx. value	all	123.4
n	no. of non-null values	age	approx. value	female	62.3
n	no. of non-null values	age	approx. value	male	61.2
… [702 more rows]					
cskurt	Crow-Siddiqui's kurtosis	income	approx. protection error	all	0.0234
cskurt	Crow-Siddiqui's kurtosis	income	approx. protection error	female	0.0123

cskurt	Crow-Siddiqui's kurtosis	income	approx. protection error	male	0.0135

Implementation details

Construction of replacement values

For each observation and each analysis variable *x*, two independent pseudo-random replacement values *r1,r2* from a distribution determined by the specified protection quartiles *q0,q1,q2,q3,q4* are computed as deterministic functions of

- the "secret" specified as a parameter to the top-level function call and
- the protection quartiles *q0,q1,q2,q3,q4* specified on the top-level function call.

In particular, *r1,r2* disclose nothing about any true values of *x*. This is done in the following way:

- For each observation, the MD5 hash of the concatenation of the secret and the value of *x* is split into four 8-digit hexadecimal numbers, giving four independently uniformly distributed pseudo-random integers *u1a,u1b,u2a,u2b* in $[0, 16^8–1]$.
- Using the Box-Muller formula, (*u1a,u1b*) and (*u2a,u2b*) are transformed into two independent standard normal variates *n1,n2*.
- Depending on *q0,q1,q2,q3,q4*, those are further transformed into independent variates *r1,r2*, where the constants *a,b,c,d* are determined so that the resulting quartiles match the prescribed ones:
  - if *q0* = –Inf and *q4* = Inf, then the normal bell shape is skewed by putting $r_i = a + b*n_i + c*\text{sqrt}(1 + n_i^2)$
  - if *q0* > –Inf but *q4* = Inf, then r_i is shifted log-normal: $r_i = q0 + \exp(a + b*n_i)$, where *q1* is ignored
  - if *q0* = –Inf but *q4* < Inf, then $–r_i$ is shifted log-normal: $r_i = q4 – \exp(a + b*n_i)$, where *q3* is ignored
  - if *q0* > –Inf and *q4* < Inf, then r_i is shifted "tan-normal": $r_i = a + b*\text{atan}(c + d*n_i)$, where either *q1* or *q3* is ignored depending on whether *q2* is nearer to *q0* or to *q4*.

These replacement values *r1,r2* can be used to compute variants of the value of the requested statistics *f* that arise when one value of *x* is replaced by either *r1* or *r2*.

Computation of true results and extreme variants

The computation of the true results of all statistics *f* and of the extreme variants of *f* that might arise by removing one value of *x* or replacing it with some replacement value from the protection distribution is done in the following steps in jk.means:

- The true power sums n,sum,uss,us3,us4 for all groups and all *x* are computed from those observations in the data that meet the where condition, and the true values of the other purely moment-based statistics (such as mean, stddev, etc.) are computed from these power sums.
- For each observation, the values of the above statistics *f* are computed that arise when only this observation was removed, and the minimum and maximum values *f_l,f_u* of those variants are determined for each group.

- All order statistics *x(i)* needed for the estimation of the needed quantiles p1,p2,...,p98,p99 are determined as well as their lower and upper neighbours. For each needed quantile *a* (=0.01, 0.02, …, 0.99), find that integer *b* and that *c* in [0,1) such that *a*(n – 1) = b + c*, and
  - estimate the *a*-quantile by *q = (1 – c)*x(b) + c*x(b + 1)*,
  - determine its minimum variant *q_l = (1 – c)*x(b – 1) + c*x(b)* and
  - its maximum variant *q_u = (1 – c)*x(b + 1) + c*x(b + 2)*.

  (Note that the removal or replacement of a single value can move *q* only inside [*q_l,q_u*]. For the estimation of min and max, we use *b* = 2 and *b = n – 1* instead of *b* = 1 and *b* = n, respectively, in order to be able to compute their minimum and maximum variants. For this reason, these two statistics are called min2 and max2 and are reported as "2nd smallest" and "2nd largest" value instead of "minimum" and "maximum")
- From the statistics *f* and their extreme variants *f_l,f_u* determined so far, now the true values of other derived statistics (such as trimean, range, etc.) and bounds for their extreme variants are computed by plugging in *f_l* and *f_u*. For example:
  - range = max2 – min2
  - range_l = min(max2_l – min2, max2 – min2_u)
  - range_u = max(max2_u – min2, max2 – min2_l)

Construction of reported approximate values and approximate protection errors

Given the true result *f* and the extreme variants *f_l* and *f_u* of some statistics f for some analysis column *x* in some group *g*, put *d* = max(| *f_u – f* |, | *f – f_l* |). The

- reported approximate value *a* of *f* and the
- reported approximate protection error *e*

are then computed as *a = f + d*n1* and *e = d*w*, where *n1* is a pseudo-random number from the standard normal distribution and *w* is a pseudo-random number of the Weibull(4,1.37) distribution. Because |*n1*| < *w* with 75.89% probability, |*n1*| < 2**w* with 95.77% probability, and |*n1*| < 3**w* with at least 99% probability, the user can construct from the reported values *a* and *e*

- a 75% confidence interval [*a – e, a + e*] for *f,*
- a 95% confidence interval [*a – 2*e, a + 2*e*] for *f,* and
- a 99% confidence interval [*a – 3*e, a + 3*e*] for *f.*

In this sense, *e* serves the same purpose for the sample value of *f* as the standard error serves for the population value of *f.*

In order to protect confidentiality from certain disclosure attempts, *n1* and *w* are computed as a deterministic functions of

- the "secret" specified as a parameter to the top-level function call,
- the values of *x* in *g,*
- the replacement values *r1, r2* of *x* in *g,*
- the name of the statistics *f* .

A disclosure attempt which averages the results of repeated identical queries or queries with only infinitesimally changed where conditions will get each time the same result and will thus not be able to reduce the protection error. A disclosure attempt which "scans" the real line for observations

by submitting a series of queries which only differ in a varying where condition of the form “x < c” for varying c will notice changes in the result whenever c crosses a true value of x but also whenever c crosses a replacement value for x; such an attempt would thus at most find a set of 3n values among which the n true values are, but would not be able to identify those true values.

n1 and *w* are computed in the following way:

- As above, for each observation, the MD5 hash of the concatenation of the secret and the value of *x* is split into four 8-digit hexadecimal numbers, giving four independently uniformly distributed pseudo-random integers *u1a,u1b,u2a,u2b* in $[0, 16^8–1]$.
- Let *s* be the sum of $u1a – 16^8/2$ for all observations that meet the where condition, plus the sum of all $u1b – 16^8/2$ for all observations that meet the where condition with *r1* instead of *x*, plus the sum of all $u1b – 16^8/2$ for all observations that meet the where condition with *r2* instead of *x*.
- Compute from the MD5 hash of the concatenation of *s* with the name of *f* four independently uniformly distributed pseudo-random integers *v1,v1,v2,v2* in $[0, 16^8–1]$.
- Use the Box-Muller formula to compute *n1* from *v1* and *v2*, and put $w = 1.37 * (22.180709777918 – \log(v3))^{1/4}$.