Análisis de conglomerados: Un caso práctico aplicado a la protección de datos
Curso 2015-16

Resumen

En este documento planteamos una solución indicativa de la segunda prueba de evaluación continua de la asignatura de Análisis Multivariante de Datos.

El objetivo del documento es dar un conjunto de guías e indicaciones sobre como se debe abordar el problema, pero no aporta una solución detallada, ya que no existe una única solución al problema y cada estudiante puede haber escogido opciones diversas e igualmente válidas.

1. Introducción

En la segunda parte del curso hemos trabajado conceptos básicos de Análisis de Componentes Principales, Escalado Multidimensional, Análisis de Correspondencias y Análisis de Conglomerados. En esta PEC se pide la aplicación de los conocimientos adquiridos durante la segunda parte del curso. En especial nos centramos en el análisis de conglomerados, que aplicaremos a un problema de protección datos.

1.1. Protección de datos

Existen muchos métodos que tienen por objetivo proteger datos de carácter personal provenientes de individuos concretos (i.e. microdatos) con el fin de poder cederlos a centros de investigación y universidades para su uso secundario.

El conjunto CENSUS, es un conjunto de microdatos y el objetivo de esta PEC será protegerlo. Los dos métodos más comunes para la protección de datos para uso secundario son:

- Adición de ruido Gaussiano: El método de adición de ruido Gaussiano se basa en perturbar los datos reales con ruido resultado de muestrear una distribución Gaussiana (habitualmente con media nula). Al añadir este ruido a los datos se busca evitar que un atacante pueda re-identificar a los individuos cuyos datos están siendo publicados.

- Microagregación: El método de microagregación persigue el mismo objetivo que el de ruido Gaussiano (*i.e.* proteger a los individuos de la re-identificación por parte de un atacante) pero sigue un proceso distinto basado en agrupación. El procedimiento de microagregación de un conjunto de datos puede resumirse en los siguientes pasos:

    1. Agrupación del conjunto original: Se aplican técnicas de análisis de conglomerados para clusterizar/agrupar los distintos registros (1080 en el caso de CENSUS) en subconjuntos disjuntos. Los subconjuntos pueden tener una cardinalidad[1] constante k (habitualmente k =3, 4, o 5) o variable.
    2. Para cada subconjunto/agrupación i de registros obtenido en el paso anterior, se calcula su vector de medias V_i.

[1]La cardinalidad hace referencia al número de elementos de cada conjunto.

3. Se substituye cada registro j del conjunto original por el vector de medias V_i del subconjunto/agrupación i en el que se ha clasificado el registro j. Con esta substitución se consigue un nuevo conjunto de datos formado únicamente por vectores de medias (repetidos al menos k veces).

Ambos métodos (*i.e.* adición de ruido y microagregación) persiguen el objetivo de modificar los datos de forma suficiente como para proteger a los individuos frente a la re-identificación pero a la vez maximizando la utilidad de los datos.

Para el caso de la microagregación, el criterio de protección se fija mediante la cardinalidad de los subconjuntos (como hemos dicho en el punto 1 del algoritmo). Por regla general, a mayor k se obtiene mayor protección y mayor pérdida de información. El objetivo es, dada una determinada k, encontrar los subconjuntos que minimizan la pérdida de información.

Para calcular la perdida de información se acostumbra a usar la suma del error cuadrado (SSE), es decir, se calcula la diferencia valor-a-valor entre el conjunto de datos original y el conjunto de datos modificado, se eleva al cuadrado dicha diferencia y se acumula. Fijada una cardinalidad k, cuando menor sea el valor de SSE mejor se considera el método de microagregación.

2. Objetivos

A partir del conjunto de datos CENSUS se pide:

1. Aplicar análisis de conglomerados: Los estudiantes pueden usar los métodos, conjuntos de métodos y/o modificaciones propias de los métodos estudiados u otros con el objetivo de:
   a) Determinar subconjuntos de cardinalidad constante k que reducen la pérdida de información (SSE).
   b) Determinar los subconjuntos de cardinalidad variable que minimizan la pérdida de información.
2. Estudiar la pérdida de información asociada a los métodos estudiados
3. Analizar críticamente los resultados obtenidos

3. Descripción del conjunto de datos

El conjunto de datos multivariante estudiado en esta prueba se conoce con el nombre de CENSUS. Este conjunto de datos ha sido usado en el proyecto

CASC (`http://neon.vb.cbs.nl/casc/index.htm`) como conjunto de test para estudiar y proponer técnicas de protección de datos en el campo del control de la revelación estadística (en Inglés: Statistical Disclosure Control).

El conjunto CENSUS contiene 1080 registros (filas) y 13 atributos (columnas) con datos extraídos de la base de datos del U.S. Bureau of the Cesus. La información de CENSUS hace referencia a salarios, impuestos, beneficios y sueldos pagados por empresas y particulares Americanos durante 1995. Para una descripción más completa puede consultarse (`http://neon.vb.cbs.nl/casc/CASCrefmicrodata.pdf`). El conjunto de datos CENSUS se encuentra públicamente disponible en: `http://neon.vb.cbs.nl/casc/CASCrefmicrodata.zip`.

4. Aplicar análisis de conglomerados

En esta sección describimos algunos de los diversos métodos que pueden usarse para agrupar elementos en clústers de cardinalidad constante k y de cardinalidad variable $p \in (k, 2k-1)$.

Como paso previo a la agrupación de elementos es necesario cargar el conjunto de datos CENSUS en R, para ello podemos usar el siguiente comando, que nos permite escoger el conjunto de datos mediante una ventana emergente (y nos ahorra definir la ruta y el nombre del fichero en el propio comando).

```
> census<-read.csv(file.choose(),header=TRUE,sep=";")
> census
```

Obtenemos el resultado siguiente:

```
  AFNLWGT AGI   EMCONTRB FEDTAX PTOTVAL STATETAX TAXINC POTHVAL INTVAL PEARNVAL FICA WSALVAL ERNVAL
1 270914  45554 4173     4621   45527   1428     30809  27      27     45500    3480 45500   45500
2 250802  57610 2639     6045   42008   1902     39234  1008    808    41000    3136 41000   41000
3 299391  56606 3315     4765   56485   1903     31767  485     485    56000    4284 56000   56000
...
...
```

Una vez cargados los datos resulta conveniente estandarizarlos dividiendo cada variable por su desviación típica y restándole su media. Para ello podemos usar el siguiente comando de R:

```
> std_census<-scale(census,center=TRUE, scale=TRUE)
```

Mediante la estandarización conseguimos eliminar el efecto de las unidades en las variables (i.e. una variable expresada en metros no tiene mayor efecto que una expresada en kilómetros).

4.1. Clusterización aleatoria por distancia mínima

Una primera aproximación sencilla consiste en calcular la matriz de distancias entre los elementos, e ir creando clusters de elementos partiendo de un elemento aleatorio al que se añaden sus $k-1$ elementos más próximos.

Para calcular la matriz de distancia usamos:

```
> distancia <- dist(std_census, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
> distancia <- as.matrix(distancia)
```

Podemos comprobar que las dimensiones de la matriz son las correctas usando el comando *dim*. En nuestro caso 1080×1080.

Con las siguientes funciones calculamos los clusters:

```
crea_cluster <-function(x, k, n, lista)
{
   ## Seleccionamos al azar un elemento de la lista
   semilla <- sample(lista, 1)

   ## Añadimos el elemento semilla al cluster
   cluster <- as.vector(semilla)

   ## Para evitar que devuelva como elemento más próximo a él mismo
   x[semilla,semilla]<-10000

   ## Eliminamos el elemento de la lista
   lista <-lista[lista!=semilla]

   for (i in 1:(k-1)){

      ## Buscamos el elemento más próximo al elemento semilla
      proximo <- which.min (x[semilla,])

      cluster <-as.vector(cluster)
      cluster <-c(as.vector(proximo), as.vector(cluster))

      ## Cancelamos el elemento
      for (j in 1:n)
      {
         x[j,proximo]<-10000
         x[proximo,j]<-10000
      }
      ## Eliminamos el elemento de la lista
      lista <-lista[lista!=proximo]
   }

   ## Cancelamos el elemento semilla
   for (j in 1:n)
   {
      x[j,semilla]<-10000
      x[semilla,j]<-10000
   }

   if (length(lista)>=k)
   {
      cluster<-rbind(crea_cluster(x,k, n, lista),as.vector(cluster))
   }
   else
   {
      as.vector(cluster)
   }
}

random_clusters <- function(x, k, n){
   ## Creamos la lista con n elementos (indices)
   lista <- c(1:n)

   clusters<-crea_cluster(x, k, n, lista)

   clusters<-as.matrix(clusters)
}
```

Llamando a la función *random-clusters* obtenemos una matriz donde cada fila representa los índices de los elementos que pertenecen al mismo cluster:

```
> clu<-random_clusters(distancia, 4, 1080)
```

Con el comando anterior estamos indicando que queremos clusters de cardinalidad 4 usando la matriz de distancia *"distancia"* e indicamos que Census tiene 1080 elementos.

El resultado *"clu"* es:

```
> clu
       [,1] [,2] [,3] [,4]
  [1,]  615  499  514  587
  [2,]  356  121  481  351
  [3,]  159  145   99  859
  [4,]  204  593   41  672
....
....

[268,]  732  307 1053 1061
[269,]  123  126  662  890
[270,]  929  523   77   76
```

Si repetimos la ejecución obtendremos resultados distintos puesto que la selección de las semillas es aleatoria y cambia con cada ejecución del algoritmo.

Ahora tenemos los elementos agrupados y necesitamos calcular el vector media de cada grupo que substituirá a los elementos originales con el fin de protegerlos. Para ello usamos el siguiente código:

```
## x:  matriz de datos originales
## mc: matriz de clusters donde cada fila contiene los índices de
##     los elementos que pertenecen al mismo cluster
## n:  número de elementos en x
## k:  cardinalidad de los clusters

crea_datos_agregados <-function(x, mc, n, k){

   x_protegida <- x

   numero_clusters <- n/k

   for(i in 1:numero_clusters)
   {
      ## Calculamos el vector agregado - El representante de cada cluster
      vector_protegido <- x[mc[i,1],]
      for(j in 2:k)
      {
         vector_protegido <- vector_protegido + x[mc[i,j],]
      }
      vector_protegido <- vector_protegido/k

      ## Una vez calculado lo guardamos en la matriz protegida
      for(j in 1:k)
      {
         x_protegida[mc[i,j],] <- vector_protegido
      }
   }
   x_protegida
}
```

Llamando a la función *crea_datos_agregados* obtenemos una nueva matriz microagregada con los datos protegidos

```
> census_protegido<-crea_datos_agregados(census, clu, 1080, 4)
```

Obtenemos en este caso:

```
      AFNLWGT      AGI EMCONTRB   FEDTAX  PTOTVAL STATETAX   TAXINC  POTHVAL   INTVAL PEARNVAL     FICA  WSALVAL   ERNVAL
1   262381.00 45173.50  3677.50  5045.75 41456.00  1393.25 31197.50   405.50   143.25 41050.50  3140.00 41050.00 39800.00
2   215240.25 47230.50  2160.50  5699.00 44681.50  1470.75 32686.25  2806.50   553.75 41875.00  3662.00 35875.00 26750.00
3   266941.25 62438.25  3162.50  7139.50 58762.50  2078.25 40648.25  1541.50  1484.00 57221.00  4377.00 57221.00 57221.00
4   146212.50 39877.25  1916.25  3967.50 19268.25  1245.50 26452.75  1798.25   828.50 17470.00  1336.25 17470.00 16970.00
```

```
...
1077 194967.75 86472.50  3073.75 17210.25 82951.50  4056.50 70390.50  8451.50  6151.50 74500.00 4837.25 74500.00 72500.00
1078 347911.75 27564.00  3523.25  1951.75 32227.25   836.25 14341.25  5977.25   364.75 26250.00 2007.75 26250.00 25250.00
1079  93554.50 47556.00  1662.50  6664.75 45402.50  2654.00 34365.00   976.50   651.50 44426.00 3398.50 44426.00 44426.00
1080 437894.25 36981.25  3264.75  3552.50 26634.25   870.25 23686.25   738.25   193.00 25896.00 1980.50 25896.00 21660.50
```

Una vez obtenida la matriz de datos protegida ya podemos calcular su pérdida de información respecto de la original (dejamos esto para la sección 5).

Debe observarse que el procedimiento descrito en este punto genera agrupaciones de cardinalidad constante k. Si quisiéramos generar grupos de cardinalidad variable podríamos usar diversas aproximaciones. Apuntamos algunas de ellas:

- Aproximación aleatoria: Se trata de la aproximación más sencilla y consiste en seleccionar la cardinalidad de cada conjunto de forma aleatoria con probabilidad uniforme en el intervalo $[k, 2k-1]$. Para cada iteración del algoritmo de creación de clusters tendremos una cardinalidad distinta en el susodicho intervalo consiguiendo así una agregación de cardinalidad variable. Se pueden usar distribuciones no uniformes para hacer prevalecer una cardinalidad sobre otra.

- Aproximación con umbral: En este caso, una vez generado un cluster de cardinalidad k se decide si se siguen añadiendo elementos al cluster en función de criterios de distancia. Algunos ejemplos son:

  - Distancia Mínima: Dado un conjunto C formado por los elementos $e_1, \cdots, e_k$ se determina el elemento $e_{candidato}$ más próximo a C y llamamos a su distancia con C, d^m_{in}. A continuación, se determina la distancia mínima d^m_{out} de $e_{candidato}$ a cualquiera de los elementos restantes (fuera de C). Si $d^m_{in} < d^m_{out}$, el elemento $e_{candidato}$ se añade a C, alternativamente el proceso termina y se pasa a la siguiente iteración del algoritmo de creación de clusters.

  - Distancia Media: Dado un conjunto C formado por los elementos $e_1, \cdots, e_k$ se determina el elemento $e_{candidato}$ más próximo al centroide (centro de masa o valor medio) de C y llamamos a su distancia con C, d^m_{in}. A continuación, se determina la distancia mínima d^m_{out} de $e_{candidato}$ a cualquiera de los elementos restantes (fuera de C). Si $d^m_{in} < d^m_{out}$, el elemento $e_{candidato}$ se añade a C, alternativamente el proceso termina y se pasa a la siguiente iteración del algoritmo de creación de clusters.

  - Distancia Minima-Máxima: Dado un conjunto C formado por los elementos $e_1, \cdots, e_k$ se determinan los elementos $e^1_{candidato}, \cdots, e^k_{candidato}$

más próximos a cada uno de los elementos de C. De entre todos estos elementos de selecciona aquel elemento $e_{candidato}$ que se encuentre a distancia máxima d_{in}^{m}. A continuación, se determina la distancia mínima d_{out}^{m} de $e_{candidato}$ a cualquiera de los elementos restantes (fuera de C). Si $d_{in}^{m} < d_{out}^{m}$, el elemento $e_{candidato}$ se añade a C, alternativamente el proceso termina y se pasa a la siguiente iteración del algoritmo de creación de clusters.

4.2. Clusterización mediante k-means

En el apartado anterior hemos descrito un método sencillo para generar agrupaciones de elementos de cardinalidad constante k, y hemos indicado algunas aproximaciones que permitirían extenderlo para generar agrupaciones de cardinalidad variable.

Un método alternativo para generar agrupaciones de cardinalidad variable es usar el algoritmo de las k-medias (k-means) para generar un número k de clusters con cardinalidad variable.

Nota: Es muy importante distinguir entre la k del algoritmo de k-means y la k que se usa como parámetro de cardinalidad. En el caso del algoritmo k-means, k hace referencia al número de conjuntos/grupos/clusters que se generar, mientras que en el caso de la microagregación, la k hace referencia a la cardinalidad de los conjuntos que se generan y no al número de conjuntos.

Usar el algoritmo k-means en R es muy sencillo puesto que ya viene implementado en la librería (stats). Con la siguiente instrucción, estaremos generando 3 clusters de cardinalidad variable sobre el conjunto Census:

```
km<-kmeans(census,3)
```

El contenido de km es:

```
K-means clustering with 3 clusters of sizes 422, 165, 493

Cluster means:
   AFNLWGT      AGI EMCONTRB   FEDTAX  PTOTVAL STATETAX   TAXINC  POTHVAL   INTVAL PEARNVAL     FICA  WSALVAL   ERNVAL
1 106372.8 58428.82 3120.448 8054.851 46933.36 3052.175 41744.89 5515.509 1582.841 41417.85 3060.647 40770.27 39556.12
2 382460.4 57762.58 3274.315 7758.661 47133.82 2541.455 40904.02 4991.164 1025.673 42142.66 3109.242 41995.13 40747.90
3 210401.0 53819.05 3184.371 7036.314 43136.60 2226.371 37575.01 4917.081 1415.677 38219.52 2829.694 37628.80 36722.18

Clustering vector:
   [1] 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 1 3 1 1 1 3 3 3 3 3 3 3 2 3 3 3 2 3 3 3 3 3 3 3 3
  [64] 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 1 1 3 1 3 3 3 1 3 1 3 3 3 1 3 1 1 3 3 1
 [127] 1 1 1 3 1 1 3 1 1 1 1 1 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 1 1 3 3 3 3 3 1 3 3 1 2 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3
 [190] 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2
 [253] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3
 [316] 3 2 3 3 3 2 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 2 3 1 1 1 1 1 1 1 1 1 1 1 2 2 2
 [379] 1 2 1 1 1 1 2 2 1 1 1 3 1 1 1 1 1 2 2 2 3 3 3 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [442] 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3
 [505] 3 3 3 1 3 3 3 3 3 3 3 3 3 2 2 3 2 3 3 2 2 2 3 2 3 3 1 1 1 3 1 3 1 1 3 3 3 3 3 1 3 1 3 3 3 1 1 1 3 1 1 1 1 1 1 3 1 3 1 3 1 1 3
 [568] 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 1 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [631] 1 1 3 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1
 [694] 3 3 3 1 1 1 3 1 3 3 1 3 1 3 3 1 3 3 1 1 1 1 3 3 1 3 1 1 3 2 3 3 3 2 3 2 3 3 3 3 3 3 3 3 2 3 3 3 1 3 3 3 1 3 3 1 2 2 2 2 2 2 1
 [757] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 3 1 3 1 1 1 1 3 1 1 3 1 1 1 1 1 1 1 1 1 1 2 1 1
 [820] 1 1 1 1 1 3 3 3 3 3 1 3 3 1 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 1 1 1 3 3 3 3 3 3 3 3 3 1 3 3 1 3 3 1 3 3 3 3 3 3 3
 [883] 3 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 3 3
 [946] 3 3 3 1 3 3 3 3 1 3 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3
[1009] 2 2 2 2 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 2 2 1 1 1 1 1 1 1 3 2 3 3 3 3 3 3 2 3 3 2 3 3 2 3 3 3 3 1 1 2 3 3 3 3 2 3 1 1 1 3 2 2
[1072] 2 1 1 1 1 1 2 1 2
```

En el resultado se observa que se han generado 3 clusters de cardinalidad variable 422, 165 y 493 con sus respectivos centroides (listados bajo "cluster means"). El vector de clustering indica a que cluster (1, 2, o 3) pertenece cada uno de los elementos del conjunto Census. La función k-means aporta más información que no necesitamos para nuestro propósito y dejamos al lector interesado profundizar en su estudio.

Debe observarse que nuestro objetivo es generar clusters de cardinalidad entre p y $2p - 1$ [2], para ello podemos indicar al algoritmo que queremos un número de clusters k tal que $k * p = 1080$ (donde 1080 es el número de elementos del conjunto Census). Por ejemplo, si queremos clusters de cardinalidad 4 podríamos crear 270 clusters:

```
km<-kmeans(census,270)
```

Al ejecutar la instrucción anterior obtenemos 270 clusters con cardinalidades 2, 4, 3, 3, 2, 2, 5, 3, 5, 1, 4, 4, 6, 6, 4, 2, 5, 7, 4, 4, 2, 5, 9, 3, 3, 5, 2, 5, 2, 3, 2, 6, 6, 4, 4, 3, 1, 1, 4, 5, 5, 5, 3, 7, 3, 4, 4, 2, 3, 5, 4, 2, 2, 7, 4, 8, 4, 5, 3, 1, 5, 1, 5, 4, 4, 5, 5, 6, 6, 4, 4, 2, 3, 8, 4, 6, 5, 3, 5, 7, 6, 4, 2, 5, 3, 4, 7, 6, 5, 6, 6, 5, 5, 2, 6, 6, 1, 5, 4, 1, 4, 3, 2, 3, 3, 3, 4, 2, 4, 1, 3, 4, 2, 7, 6, 4, 8, 5, 7, 5, 2, 7, 3, 5, 2, 7, 4, 3, 1, 3, 5, 1, 5, 3, 7, 2, 3, 6, 8, 4, 3, 3, 1, 5, 2, 3, 3, 3, 3, 1, 6, 7, 1, 4, 4, 10, 1, 5, 2, 2, 3, 3, 6, 5, 4, 4, 3, 5, 3, 3, 2, 6, 3, 5, 1, 5, 4, 5, 5, 4, 7, 3, 5, 2, 4, 6, 4, 4, 4, 6, 5, 4, 2, 6, 6, 2, 9, 6, 4, 1, 5, 4, 11, 1, 4, 2, 2, 2, 5, 3, 3, 4, 7, 3, 3, 4, 4, 3, 3, 4, 3, 5, 3, 5, 6, 3, 4, 4, 8, 3, 4, 2, 2, 3, 4, 5, 7, 2, 1, 4, 5, 3, 4, 2, 6, 4, 2, 5, 5, 5, 3, 2, 1, 1, 9, 3, 7, 1, 3, 3, 4, 2, 9, 1, 8, 3, 1, 7, 2 y 2 respectivamente.

Con esta información podemos generar la matriz de datos protegida/agregada usando el siguiente código R:

```
## x:  matriz de datos originales
## km: resultado de la ejecucion de kmeans
## n:  número de elementos en x
crea_datos_agregados_kmeans <-function(x, km, n){
```

[2]Llamaremos p a la cardinalidad de los conjuntos para no confundirla con la k del número de conjuntos del algoritmo k-means

```
   x_protegida <- x
   for(i in 1:n)
   {
    x_protegida[i,]<- km$centers[km$cluster[i],]
    }
    x_protegida
}
```

Una vez creada la matriz de datos agregados ya podemos calcular su pérdida de información (lo cual dejamos para el apartado 5). Sin embargo, es importante destacar que el algoritmo k-means NO garantiza que los clusters tengan una cardinalidad entre p y $2p-1$, de modo que sería necesario aplicar un algoritmo de postprocesado que lo garantice (dejamos este punto para el lector).

4.3. Otras aproximaciones

En esta solución indicativa tansolo hemos descrito dos posibles soluciones. Sin embargo existen muchas aproximaciones distintas que el estudiante podría haber usado con éxito. La siguiente es una breve lista (no completa) de posibles soluciones alternativas:

- Usar el paquete SDC micro: El paquete SDC-Micro de R incorpora una serie de algoritmos de protección de microdatos entre los que se cuentan la adición de ruido, la microagregación, el pram, etc.. Dejamos para el lector interesado el estudio de esta librería. Hemos añadido las 4 primeras páginas de descripción de la librería en el apéndice A.

- Implementar el algoritmo V-MDAV: El algoritmo V-MDAV es una evolución del algoritmo MDAV (Miximum Distance to Average Vector) inspirado en el método de Ward que permite generar clústers de cardinalidad variable. El lector interesado encontrará la descripción del algoritmo en el Apéndice B.

- Usar algoritmos evolutivos: Dado que el problema de protección de microdatos no se puede resolver de forma óptima en tiempo polinómico (es un problema NP) se ha sugerido el uso de algoritmos evolutivos y otras técnicas de inteligencia artificial basada en agregación. El lector interesado puede encontrar un ejemplo en el Apéndice C.

5. Estudio de la pérdida de información y análisis

Para estudiar la pérdida de información producida por el algoritmo de protección se acostumbra a usar la suma del error cuadrado (SSE)[3]. Resulta habitual normalizar el valor de SSE por el del SST que se obtiene calculando el SSE sobre una microagregación de cardinalidad $k = n$ donde n es el número de elementos del conjunto original.

El siguiente código hace los cálculos de SSE y SST:

```
## Funcion que calcula el SSE de dos matrices valor a valor
## a: matriz 1 (por ejemplo la original)
## b: matriz 2 (por ejemplo la perturbada)
## n: numero de filas
## m: numero de columnas

SSE <- function(a,b,n,m){

   total <- 0;

   for(i in 1:n){
      for(j in 1:m){
         error <- (a[i,j]-b[i,j])
         error_cuadrado <- error*error
         total <- total + error_cuadrado
      }
    }
   total
}

##Funcion que calcula el SST de una matriz considerando su
## vector media
## a: matriz original
## n: numero de elementos
## m: numero de dimensiones (columnas)
SST <- function(a,n,m){

   vector_media <- a[1,]

   for(i in 2:n){
      vector_media <- vector_media + a[i,]
   }
   vector_media <- vector_media/n

   error <- 0
```

[3]Del Inglés Sum of Squared Errors

```
    total <- c(0)
    for(i in 1:n){
        for(j in 1:m){
            error <- (a[i,j]-vector_media[j])
            error_cuadrado <- error*error
            total <- total + error_cuadrado
        }
    }
    as.numeric(total)
}
```

Mediante el uso de las funciones anteriores podemos calcular la pérdida de información normalizada ($\frac{SSE}{SST} \times 100$) para distintos valores de k.

Los resultados que hemos obtenido con los dos algoritmos descritos en la sección 4.1 y 4.2 son los siguientes:

5.1. Resultados para el algoritmo aleatorio

Dado que el algoritmo basado en distancia mínima tiene una componente aleatoria, lo hemos ejecutado 10 veces y hemos calculado su resultado medio.

	k=3	k=4	k=5	k=10
1	10.54670	16.90013	16.55097	24.52994
2	10.80634	13.34749	16.34394	24.62827
3	10.96795	14.74327	17.61700	26.18878
4	11.62616	14.93601	17.46716	25.58967
5	11.24932	15.37628	16.48567	26.96298
6	12.86896	13.74675	18.07996	26.99309
7	11.07508	14.80755	15.45247	24.41784
8	10.07873	13.84617	16.90570	25.65330
9	12.24756	14.19388	16.28544	26.46798
10	10.96171	13.42706	19.42736	27.02375
Media	11.24285	14.53246	17.06157	25.84556
Desviación	0.817429	1.07651	1.126958	1.044153

Cuadro 1: Resultados para el algoritmo de distancia mínima

Puede observarse claramente que el aumento en la cardinalidad de los clusters se traduce en un aumento de la pérdida de información.

5.2. Resultados para el algoritmo k-means

Hemos ejecutado el algoritmo k-means para $k = 360$, $k = 270$, $k = 216$ y $k = 108$ para obtener clusters de cardinalidad cercana a 3, 4, 5 y 10. Sin embargo, esta cardinalidad no está garantizada al 100 % por el algoritmo implementado.

Como en el caso anterior, el algoritmo k-means presenta una componente aleatoria y hemos repetido su ejecución 10 veces para tener un resultado estadísticamente más preciso.

N	k=360	k=270	k=216	k=108
1	0.8321792	1.123968	1.45008	2.73437
2	0.9086227	1.139451	1.459619	2.86669
3	0.8121587	1.148357	1.5280210	2.956615
4	1.000095	1.16147	1.48798	2.984955
5	0.7842453	1.156666	1.435719	2.969123
6	0.8289067	1.127556	1.443031	2.955235
7	0.828971	1.154142	1.45151	2.819843
8	0.7986362	1.115812	1.41609	3.218586
9	0.8226696	1.130614	1.663494	2.740492
10	0.8287398	1.223954	1.391102	2.97852
Media	0.84452245	1.1481991	1.4726656	2.9224433
Desviación	0.063650624	0.030677696	0.076694994	0.141926174

Cuadro 2: Resultados para el algoritmo k-means

Se observa, como en el caso anterior, un aumento de la pérdida de información con la cardinalidad de los clusters.

5.3. Análisis críticos de los resultados

El uso de algoritmos de agrupación para la protección de microdatos no es nuevo. Existen numerosas aproximaciones que se aplican con éxito. En este documento, hemos analizado dos propuestas de fácil aplicación por parte de los estudiantes. Los resultados obtenidos son consistentes con el actual estado del arte.

Podemos observar que el algoritmo aleatorio introduce una alta pérdida de información. Ello se debe a que la selección de los puntos iniciales de generación de los clusters es totalmente aleatoria y no hay ningún proceso de refinado (adaptación de los grupos) y, en consecuencia, los clusters resultantes no se adaptan a los datos y pueden generar malas agrupaciones. Sin embargo, los resultados son consistentes y muestran una tendencia natural (y conocida) a aumentar la pérdida de información al aumentar la cardinalidad de los clusters. En relación con el estado del arte, este método es de 2 a 3 veces peor que los métodos usados en la práctica.

En contraposición a los pobres resultados del algoritmo aleatorio encontramos los del método basado en k-means. Si analizamos los resultados vemos que la pérdida de información es muy pequeña, en parte gracias a la buena agrupación proporcionada por k-means y su capacidad de generar grupos de cardinalidad variable. Sorprendentemente funciona mejor que los métodos usados en la literatura.

Sin embargo, un estudio más reposado indica que ello se debe a que el algoritmo no garantiza la cardinalidad mínima requerida para la protección y existe un elevado numero de clusters de cardinalidad 1 que no introducen pérdida de información. En conclusión, a pesar de que el algoritmo propuesto es un buen ejercicio para aplicar los conocimientos de la asignatura de Análisis Multivariante de Datos, debería modificarse convenientemente para satisfacer las condiciones de cardinalidad para poder ser usado en protección de datos (tal y como se ha indicado en secciones anteriores).

A. Apéndice A

Package ‘sdcMicro’

May 4, 2015

Type Package

Title Statistical Disclosure Control Methods for Anonymization of Microdata and Risk Estimation

Version 4.5.0

Date 2015-04-30

Author Matthias Templ, Alexander Kowarik, Bernhard Meindl

Maintainer Matthias Templ <`matthias.templ@gmail.com`>

Description Data from statistical agencies and other institutions are mostly confidential. This package can be used for the generation of anonymized (micro)data, i.e. for the creation of public- and scientific-use files.
In addition, various risk estimation methods are included.
Note that the package 'sdcMicroGUI' includes a graphical user interface for various methods in this package.

LazyData TRUE

ByteCompile TRUE

LinkingTo Rcpp

Depends R (>= 2.10), brew, knitr,data.table,xtable

Suggests laeken

Imports car, robustbase, cluster, MASS, e1071, tools, Rcpp, methods,sets

License GPL-2

URL `https://github.com/alexkowa/sdcMicro`

Collate '0classes.r' 'addNoise.r' 'aux_functions.r' 'dataGen.r' 'dRisk.R' 'dRiskRMD.R' 'dUtility.R' 'freqCalc.r' 'globalRecode.R' 'GUIfunctions.R' 'indivRisk.R' 'LLmodGlobalRisk.R' 'LocalRecProg.R' 'localSupp.R' 'localSupp2.R' 'localSupp2Wrapper.R' 'localSuppression.R' 'mdav.R' 'measure_risk.R' 'methods.r' 'microaggregation.R' 'plot.localSuppression.R' 'plotMicro.R' 'pram.R' 'print.freqCalc.R' 'print.indivRisk.R' 'print.localSuppression.R' 'print.micro.R' 'rankSwap.R'

'report.R' 'shuffle.R' 'suda2.R' 'summary.freqCalc.R'
'summary.micro.R' 'summary.pram.r' 'swappNum.R'
'timeEstimation.R' 'topBotCoding.R' 'valTable.R' 'zzz.R'
'printFunctions.R' 'mafast.R' 'maG.R' 'show_sdcMicroObj.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-05-04 15:33:28

R topics documented:

`sdcMicro-package` *Statistical Disclosure Control (SDC) for the generation of protected microdata for researchers and for public use.*

Description

This package includes all methods of the popular software mu-Argus plus several new methods. In comparison with mu-Argus the advantages of this package are that the results are fully reproducible even with the included GUI, that the package can be used in batch-mode from other software, that the functions can be used in a very flexible way, that everybody could look at the source code and that there are no time-consuming meta-data management is necessary. However, the user should have a detailed knowledge about SDC when applying the methods on data.

The package is programmed using S4-classes and it comes with a well-defined class structure.

The implemented graphical user interface (GUI) for microdata protection serves as an easy-to-handle tool for users who want to use the sdcMicro package for statistical disclosure control but are not used to the native R command line interface. In addition to that, interactions between objects which results from the anonymization process are provided within the GUI. This allows an automated recalculation and displaying information of the frequency counts, individual risk, information loss and data utility after each anonymization step. In addition to that, the code for every anonymization step carried out within the GUI is saved in a script which can then be easily modified and reloaded.

Details

Package:	sdcMicro
Type:	Package
Version:	2.5.9
Date:	2009-07-22
License:	GPL 2.0

Author(s)

Matthias Templ, Alexander Kowarik, Bernhard Meindl

Maintainer: Matthias Templ <templ@statistik.tuwien.ac.at>

References

Templ, M. and Meindl, B. *Practical Applications in Statistical Disclosure Control Using R*, Privacy and Anonymity in Information Management Systems, Bookchapter, Springer London, pp. 31-62, 2010

Kowarik, A. and Templ, M. and Meindl, B. and Fonteneau, F. and Prantner, B.: *Testing of IHSN Cpp Code and Inclusion of New Methods into sdcMicro*, in: Lecture Notes in Computer Science, J. Domingo-Ferrer, I. Tinnirello (editors.); Springer, Berlin, 2012, ISBN: 978-3-642-33626-3, pp. 63-77.

Templ, M. *Statistical Disclosure Control for Microdata Using the R-Package sdcMicro*, Transactions on Data Privacy, vol. 1, number 2, pp. 67-85, 2008. http://www.tdp.cat/issues/abs.a004a08.php

Templ, M. *New Developments in Statistical Disclosure Control and Imputation: Robust Statistics Applied to Official Statistics*, Suedwestdeutscher Verlag fuer Hochschulschriften, 2009, ISBN: 3838108280, 264 pages.

Examples

```
## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
f$fk
f$Fk
## with missings:
x <- francdat
x[3,5] <- NA
x[4,2] <- x[4,4] <- NA
x[5,6]  <- NA
x[6,2]  <- NA
f2 <- freqCalc(x,  keyVars=c(2,4,5,6),w=8)
f2$Fk
## individual risk calculation:
```

B. Apéndice B

V-MDAV: A Multivariate Microaggregation With Variable Group Size

Agusti Solanas and Antoni Martínez-Ballesté

CRISES Research Group. Department of Computer Engineering and Mathematics. Rovira i Virgili University. Av.Països Catalans 26. 43007 Tarragona. Catalonia. Spain. {agusti.solanas,antoni.martinez}@urv.net

Summary. Microaggregation is a clustering problem with minimum size constraints on the resulting clusters or groups; the number of groups is unconstrained and the within-group homogeneity should be maximized. In the context of privacy in statistical databases, microaggregation is a well-known approach to obtaining anonymized versions of confidential microdata. Optimally solving microaggregation on multivariate data sets is known to be difficult (NP-hard). Therefore, heuristic methods are used in practice. This paper presents a new heuristic approach to multivariate microaggregation, which provides variable-sized groups (and thus higher within-group homogeneity) with a computational cost similar to the one of fixed-size microaggregation heuristics.

1 Introduction

Many among the usual transformations performed on data sets (data mining, knowledge discovery, statistical disclosure control for database privacy, etc.) can be viewed as clustering processes with different kinds of constraints [3, 6, 13, 18]. In this article we address the problem of microaggregation, a special kind of clustering problem where there are constraints on the minimum size of clusters or groups, but not on their number, and the within-groups homogeneity should be maximized. Microaggregation is a problem appearing in statistical disclosure control (SDC), where it is used to cluster a set of records in groups of at least k records, with k being a user-definable parameter. The collection of groups is called a k-partition of the data set. The microaggregated data set is built by replacing each original record by the centroid of the group it belongs to. The microaggregated data set can be released without jeopardizing the privacy of the individuals which form the original data set: records within a group are indistinguishable in the released data set. The higher the within-group homogeneity in the original data set, the lower the information loss incurred when replacing records in a group by their centroid; therefore within-group homegeneity is inversely related to information loss caused by microaggregation. The inverse of the within-groups sum of squares SSE is the most usual measure for clustering homogeneity [7, 9, 10, 14, 19]. SSE can be computed as

$$SSE = \sum_{i=1}^{s} \sum_{j=1}^{n_i} (\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)'(\mathbf{x}_{ij} - \bar{\mathbf{x}}_i) \quad (1)$$

where s is the number of generated sets, n_i is the number of records in the i-th set, $\mathbf{x}_{ij}$ is the j-th record in the i-th set and $\bar{\mathbf{x}}_i$ is the centroid of the i-th set. In terms of SSE, the microaggregation problem consists of finding a k-partition with minimum SSE.

Microaggregation has been used for several years in different countries: it started at Eurostat [2] in the early nineties, and has since then been used in Germany [16] and several other countries [8]. These years of experience and practice devoted to microaggregation have bequeathed us a variety of approaches, which we next briefly summarize.

- Optimal methods:
  - Univariate case: In [11] a polynomial algorithm for optimal univariate microaggregation was presented. But optimal multivariate microaggregation was shown to be NP-hard in [15].
  - Multivariate case: Optimal multivariate microaggregation was shown to be NP-hard in [15]. So the only practical multivariate microaggregation methods are heuristic.
- Heuristic methods:
  - Fixed-size heuristics: The best-known example of this class is Maximum Distance to Average Vector (MDAV) [4, 6, 12]. MDAV produces groups of fixed cardinality k and, when the number of records is not divisible by k, one group with a cardinality between k and $2k-1$. MDAV has proven to be the best performer in terms of time and one of the best regarding the homogeneity of the resulting groups.
  - Variable-size heuristics: These yield k-partitions with group sizes varying between k and $2k-1$. Such a flexibility can be exploited to achieve higher within-group homogeneity. Variable-size methods include the genetic-inspired approach for small data sets in [18] and also [3, 13, 5].

To sum up, the challenge in microaggregation is to design good heuristics for the multivariate case, where "goodness" refers to combining high group homogeneity and computational efficiency. This paper is about a new method along this line. Since optimal microaggregation via exhaustive search is not a feasible benchmark for data sets of more than 15 records, we take as benchmark heuristic to compare with our new proposal the MDAV.

MDAV is, to the best of our knowledge, one of the best heuristic methods for multivariate microaggregation. However, being a fixed-size heuristic, there are situations in which it yields a k-partition far from the optimal one. This is illustrated by the next toy example.

Example 1. Let be D our data set composed by 13 records having 2 attributes.

$$D = \{(2.4, 3), (1.68, 4.9), (3.18, 5.54), (5.32, 3.6), (18.68, 11.49), (20.14, 9.56), (19.85, 12.33), (13.67, 18.9), (17.11, 21), (16.07, 19.23), (21.28, 18.9), (22, 21), (23, 18.5)\}$$

Considering $k = 3$, Figure 1 illustrates the output of the MDAV algorithm when applied over our toy-example. The figure depicts the records in D *(circles)* and the global centroid *(triangle)*. The group marked in red is very scattered and causes the overall 3-partition to be poor. This example shows that the fixed-size nature of MDAV may fail to suitably adapt the resulting k-partition to the particular data set.

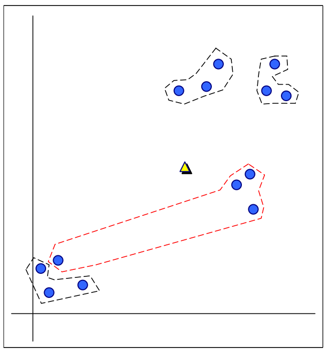

Fig. 1. Output of the MDAV algorithm on the toy example

1.1 Contribution and plan of the paper

In this paper we propose a new heuristic method for multivariate microaggregation called V-MDAV. V-MDAV stands for Variable-size Maximum Distance to Average Vector. It improves on the well-known MDAV method in terms of lower SSE while maintaining an equivalent computational cost.

The rest of the paper is organized as follows. In Section 2 we describe the proposed algorithm. Section 3 shows the experimental results obtained on several data sets. In Section 4 an assessment of the method's computational complexity is given. Finally, Section 5 is a conclusion.

2 Description of V-MDAV

As mentioned above, MDAV generates groups of fixed-size k and, thus, it lacks flexibility for adapting the group size to the distribution of the records in the data set, which may result in poor within-group homogeneity. V-MDAV is a new algorithm that intends to overcome this limitation by computing a variable size k-partition with a computational cost similar to the MDAV cost. Next, the main steps of V-MDAV are described in detail.

2.1 Group Generation

This is the first step of V-MDAV and it follows a similar strategy to MDAV, this is, firstly a matrix of distances containing the distances between any two records is built *(line 2 in the Algorithm)*. Next the global centroid is computed and the most

Algorithm 1 Variable group size algorithm

```
01) function V-MDAV in:DataSet $D$, Integer $k$; out:microaggregatedSet $M$ is
02)   Compute_Distances_Matrix($D$);
03)   $C$ = ComputeCentroidOfDataSet($D$);
04)   while (ThereAreMoreThan[$k-1$]RecordsToAssign) do
05)     $e$ = SelectTheMostDistantRecordToCentroid ($D$,$C$);
06)     $g_i$ = BuildGroupFromRecord($e$,$D$,$k$);
07)     $g_i$ = ExtendTheGroup($g_i$,$D$,$k$);
08)   end while
09)   $g_1 \dots g_s$ = AssignRemainingUnassignedRecords($D$, $g_1 \dots g_s$);
10)   $M$ = BuildMicroaggregatedDataSet($D$, $g_1 \dots g_s$);
11)   return $M$
12) end function
```

distant record from it is searched *(lines 3 and 5)*. Once this record is found, a group of k records is formed by selecting the $k-1$ records closest to the initial one *(line 6)*. At this point MDAV would apply this strategy until all records in the data set are assigned to a group; in V-MDAV there is an additional step *(line 7)* that allows it to adapt to the data set distribution and generate variable-size groups. As to complexity, the most important differences between MDAV and V-MDAV are:

- MDAV computes a centroid in each iteration. V-MDAV only computes the data set centroid at the beginning. This results in a computational time improvement.
- MDAV does not build a matrix of distances; on the contrary, it computes distances as many times as needed. Thus, V-MDAV is faster.

2.2 Extension of the group

Given a group g with p records, the record e_{min} among unassigned records outside g nearest to g and the minimum distance d_{in} between e_{min} and g are defined by Equation (2) and (3):

$$d_{in} = \min_{j \in [1, N_{un}]} [(d(e_i^g, e_j)], \forall i \in [1, p]] \tag{2}$$

$$e_{min} = \arg \min_{j \in [1, N_{un}]} [(d(e_i^g, e_j)], \forall i \in [1, p]] \tag{3}$$

where e_i^g denotes the i-th record in group g, e_j means the j-th record in the unassigned set of records and N_{un} is the number of unassigned records, this is, the number of records which have not yet been assigned to any group. If Equation (3) is satisfied by more than one record, one of them is randomly selected as e_{min}. Next, the minimum distance d_{out} from the selected record e_{min} to any of the remaining unassigned records is found using Equation (4):

$$d_{out} = \min_{j \in [1, N_{un}], e_{min} \neq e_j} [(d(e_{min}, e_j)] \tag{4}$$

Finally, in order to decide on the inclusion of e_{min} into group g, we compare its distance d_{in} to g with its distance d_{out} to the closest unassigned neighbor. Expression (5) gives the decision criterion:

$$ADD\_RECORD = \begin{cases} \text{YES if } \; d_{in} \; < \; \gamma \, d_{out} \\ \\ \text{NO} \quad \text{otherwise} \end{cases} \tag{5}$$

where γ is a *gain factor* that has to be tuned in order to improve the adaptability of V-MDAV. How to determine the best values of γ is not straightforward and due to space limitations it will not be discussed in this paper. We will expand a little more on this aspect in Section 3. The extension process is repeated until the group size equals $2k - 1$ or the condition in Expression (5) is not satisfied, because it was shown in [4] that, in an optimal k-partition, each group includes between k and $2k - 1$ records.

2.3 Addition of the last records

Similarly to MDAV, the proposed method can leave some records unassigned at the end of the main loop. Thus, it is necessary to assign these records to a group before ending the algorithm. The remaining records are assigned to their closest group *(line 9)*. At the end of all these steps, a microaggregated data set M is built from the resulting k-partition represented in the Algorithm 1 as $(g_1 \ldots g_s)$ *(line 10)*.

3 Experimental results

The experiments which have been carried out use very different data sets ranging from real to synthetic data up to 20 dimensions and from 400 records to almost 6000. Specifically, we have used the *Tarragona*, *Census* and *EIA* data sets [1], because they have become the usual reference data sets to test multivariate microaggregation [4, 5, 13]. Additionally, three new clustered data sets [1] have been synthetically created, as in [3], in order to show how MDAV and V-MDAV behave on this kind of data sets.

Our experiments demonstrate that V-MDAV is the best performer when applied to clustered data sets, because it can adapt to the structure of data. However, the distinction between *clustered* and *scattered* data does not only depend on the data but on the value of k. This variation in the outlook of the data set is similar to the effect of looking at a picture from the near or far distance. As an example let us consider the effect that a "zoom in" produces on an image. When we are very close to an image we are able to distinguish a greater number of details and some previously hidden clusters could become visible. On the other hand, a "zoom out" can produce a similar effect because, in some situations, in which we are very close to the image, the trees hide the wood. These "zoom in" and "zoom out" effects are equivalent to the ones produced by the variation of k.

For the above reasons, deciding whether a data set is scattered or clustered is not an easy task. In [17] a study on the type of data sets is presented and seems to indicate the existence of a clear dichotomy between the behaviors of clustered and scattered data sets. This research line is still open and we take some of its ideas to determine the data set type (Table 1, left-hand side). The determination of the best value of γ for a given data set is not straightforward. However, there is empirical evidence that values of γ close to zero are effective when the data are scattered because a low γ causes the algorithm to expand groups very conservatively, that is,

[1] We say that a data set is *clustered* when its records form natural clusters. Otherwise, we call the data set *scattered*: no natural clusters are apparent.

towards very close records only. In fact, for $\gamma = 0$ V-MDAV is equivalent to MDAV. On the contrary, when the data set is clustered the best values for γ are usually close to one. Then, V-MDAV takes more risks in expanding a group. In our experiments we have selected a $\gamma = 0.2$ for scattered data sets and a $\gamma = 1.1$ for clustered data sets. Table 1 (right-hand side) shows the information loss caused by each method measured using the following expression

$$I_{loss} = \frac{SSE}{SST} \cdot 100 \tag{6}$$

where SST is the total sum of squares (sum of squared Euclidean distances from all records to the data set centroid). The lower I_{loss}, the better. From Table 1, it can

Table 1. Results of the experiments. *Left:* Classification of the data sets into clustered or scattered according to k. *Right:* Information loss caused by MDAV and V-MDAV for different data sets and values of k.

Dataset	$k=3$	$k=4$	$k=5$	$k=10$
"Census"	S	S	S	S
"Tarragona"	S	S	S	S
"EIA"	S	S	C	C
"Synthetic 1"	C	S	S	S
"Synthetic 2"	C	S	S	S
"Synthetic 3"	S	C	C	S

Dataset	Method	$k=3$	$k=4$	$k=5$	$k=10$
"Census"	MDAV	5.66	7.51	9.01	14.07
	V-MDAV	5.69	7.52	8.98	14.07
"Tarragona"	MDAV	16.96	19.70	22.88	33.26
	V-MDAV	16.96	19.70	22.88	33.26
"EIA"	MDAV	0.49	0.67	1.78	3.54
	V-MDAV	0.53	0.75	1.30	2.82
"Synthetic 1"	MDAV	7.64	8.84	12.20	30.21
	V-MDAV	0.94	5.64	11.83	30.03
"Synthetic 2"	MDAV	9.54	13.35	17.34	39.28
	V-MDAV	1.63	8.21	16.49	38.91
"Synthetic 3"	MDAV	5.95	7.66	9.03	22.36
	V-MDAV	2.17	1.85	5.21	21.41

be seen that the results obtained by MDAV and V-MDAV are very similar when working with scattered data sets: V-MDAV outperforms or matches MDAV, except for "Census" and "EIA" with $k = 3$ and $k = 4$, where it is *slightly* outperformed (by at most 0.08%). This is not a bad result, because MDAV is known to perform very very well on scattered data sets. On the other hand, V-MDAV clearly improves the result of MDAV when working on clustered data sets. More specifically, a spectacular improvement by almost 8% information loss reduction is achieved for the *Synthetic 2* data set with $k = 3$; the improvement is almost 7% for *Synthetic 1* with $k = 3$, etc. After analyzing these results, we can conclude that V-MDAV behaves as well as MDAV on scattered data and clearly outperforms it on clustered data.

4 Computational analysis

Let n be the number of records in the data set. The dimension of the data set will not be considered in the analysis because it is usually much smaller than n and it only has a slight computational impact on distance computations. The breakdown of complexity is as follows:

1. Computing the distances matrix can be done with $O(n^2)$ (proportional to the number of distances to be computed).
2. Computing the centroid has a cost of $O(n)$.
3. Computing the distance from each point to the centroid requires n distance computations. Thus, it has $O(n)$ cost.
4. *Main loop.* In each iteration, between k and $2k-1$ records are microaggregated. Since on average $(3k-1)/2$ records are grouped, about $2n/(3k-1)$ iterations are needed. Each iteration consists of:
   a) Selecting the most distant record from the centroid, which costs $O(n)$.
   b) Building a group of size k. Since $k-1$ records are to be added and $n/2$ records, on average, remain unassigned, $((k-1)n)/2$ comparisons are required. Thus, the cost is $O(n)$.
   c) Extending the group. Up to $k-1$ records can be added to the current group, say $(k-1)/2$ on average. This requires performing $((k-1)n)/4$ comparisons, with a cost $O(n)$.

   Therefore the computational cost of the main loop is
$$O(\frac{2n}{3k-1}(n+n+n)) = O(\frac{6n^2}{3k-1}) = O(n^2)$$
5. Finally, as some records may remain unassigned, a final step is performed. For each unassigned record, n distances are computed to determine the group to which it will be added. This results in a $O(n)$ cost.

Thus the overall complexity of V-MDAV is $O(n^2)$.

5 Conclusion and further work

V-MDAV, a new method for multivariate microaggregation has been presented. V-MDAV overcomes the fixed group size constraint of previous heuristics with a similar computational cost. Therefore, V-MDAV offers increased flexibility without computational over cost. Moreover, the way in which V-MDAV expands the groups can be tuned by using the γ *gain factor*. Experimental results show substantial performance improvement by V-MDAV when working on clustered data sets. A number of research issues remain open and will be addressed in future work: *a)* Analyze the behavior of the γ *gain factor* in detail and determine the optimal value of γ for a given data set. *b)* Study the dichotomy between clustered and scattered data sets.

Acknowledgement. The authors wish to thank Prof. Josep Domingo-Ferrer and Dr. Josep M. Mateo-Sanz for their insightful comments and kind suggestions. Partial support by the Government of Catalonia under grant 2005 SGR 00446 and by the Spanish Ministry of Education and Science under project SEG2004-04352-C04-01 "PROPRIETAS" is also gratefully acknowledged.

References

1. R. Brand, J. Domingo-Ferrer, and J. M. Mateo-Sanz. Reference data sets to test and compare sdc methods for protection of numerical microdata, 2002. European Project IST-2000-25069 CASC, http://neon.vb.cbs.nl/casc.

2. D. Defays and P. Nanopoulos. Panels of enterprises and confidentiality: the small aggregates method. In *Proc. of 92 Symposium on Design and Analysis of Longitudinal Surveys*, pages 195–204, Ottawa, 1993. Statistics Canada.
3. J. Domingo-Ferrer, A. Martínez-Ballesté, and J. M. Mateo-Sanz. Efficient multivariate data-oriented microaggregation. *Manuscript*, 2005.
4. J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.
5. J. Domingo-Ferrer, F. Sebé, and A. Solanas. A polynomial-time approximation to optimal multivariate microaggregation. *Manuscript*, 2005.
6. J. Domingo-Ferrer and V. Torra. Ordinal, continuous and heterogenerous k-anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11(2), 2005.
7. A. W. F. Edwards and L. L. Cavalli-Sforza. A method for cluster analysis. *Biometrics*, 21:362–375, 1965.
8. Economic Commission for Europe. Statistical data confidentiality in the transition countries: 2000/2001 winter survey. In *Joint ECE/Eurostat Work Session on Statistical Data Confidentiality*, 2001. Invited paper n.43.
9. A. D. Gordon and J. T. Henderson. An algorithm for euclidean sum of squares classification. *Biometrics*, 33:355–362, 1977.
10. P. Hansen, B. Jaumard, and N. Mladenovic. Minimum sum of squares clustering in a low dimensional space. *Journal of Classification*, 15:37–55, 1998.
11. S. L. Hansen and S. Mukherjee. A polynomial algorithm for optimal univariate microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1043–1044, July-August 2003.
12. A. Hundepool, A. Van de Wetering, R. Ramaswamy, L. Franconi, A. Capobianchi, P.-P. DeWolf, J. Domingo-Ferrer, V. Torra, R. Brand, and S. Giessing. *μ-ARGUS version 4.0 Software and User's Manual*. Statistics Netherlands, Voorburg NL, may 2005. http://neon.vb.cbs.nl/casc.
13. M. Laszlo and S. Mukherjee. Minimum spanning tree partitioning algorithm for microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):902–911, 2005.
14. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.
15. A. Oganian and J. Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal of the United Nations Economic Comission for Europe*, 18(4):345–354, 2001.
16. M. Rosemann. Erste ergebnisse von vergleichenden untersuchungen mit anonymisierten und nicht anonymisierten einzeldaten amb beispiel der kostenstrukturerhebung und der umsatzsteuerstatistik. In *G. Ronning and R. Gnoss (editors) Anonymisierung wirtschaftsstatistischer Einzeldaten, Wiesbaden: Statistisches Bundesamt*, pages 154–183, 2003.
17. Agusti Solanas. On the type of data dets: Clustered vs. scattered. *Manuscript*, 2006.
18. Agusti Solanas, Antoni Martínez-Ballesté, Josep M. Mateo-Sanz, and Josep Domingo-Ferrer. Multivariate microaggregation based on a genetic algorithm. *Manuscript*, 2006.
19. J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.

C. Apéndice C

A Genetic Approach to Multivariate Microaggregation for Database Privacy

Antoni Martínez-Ballesté, Agusti Solanas, Josep Domingo-Ferrer and Josep M. Mateo-Sanz
Dept. of Computer Engineering and Maths,
Universitat Rovira i Virgili,
Av. Països Catalans 26,
E-43007 Tarragona, Catalonia
e-mail {antoni.martinez,agusti.solanas}@urv.cat
e-mail {josep.domingo,josepmaria.mateo}@urv.cat

Abstract

*Microaggregation is a technique used to protect privacy in databases and location-based services. We propose a new hybrid technique for multivariate microaggregation. Our technique combines a heuristic yielding fixed-size groups and a genetic algorithm yielding variable-sized groups. Fixed-size heuristics are fast and able to deal with large data sets, but they sometimes are far from optimal in terms of the information loss inflicted. On the other hand, the genetic algorithm obtains very good results (*i.e. *optimal or near optimal), but it can only cope with very small data sets.*

Our technique leverages the advantages of both types of heuristics and avoids their shortcomings. First, it partitions the data set into a number of groups by using a fixed-size heuristic. Then, it optimizes the partitions by means of the genetic algorithm. As an outcome of this mixture of heuristics, we obtain a technique that improves the results of the fixed-size heuristic in large data sets.

1 Introduction

Microdata, *i.e.* sets of records containing information on individual respondents, are important for planning or analysis purposes in a number of human activities: healthcare, market analysis, public policies, etc. As a consequence, statistical agencies and large enterprises routinely gather such microdata. However, this information cannot be freely released because the privacy of the respondents would be jeopardized. Against common belief, suppressing the direct identifiers (names, passport numbers, etc.) is not enough to ensure respondent privacy: *e.g.* if a data set contains civil status, age and sensitive information (salary, diseases, etc.) for a number of people, a 17-year old widow is easy to re-identify even if names have been suppressed in the data set.

A plethora of methods for statistical database privacy collectively known as statistical disclosure control (SDC) methods have been developed, which include on-line database query control [19] and data perturbation. In the latter case the aim is to anonymize the collected information so that the privacy of respondents is preserved and the anonymized data are still useful. Microaggregation is a family of SDC methods achieving anonymization through data perturbation. Quite recently, microaggregation has been shown to be also applicable to privacy problems different from statistical databases, namely location privacy [3].

The microaggregation problem can be stated as a clustering problem with restrictions on the size of clusters. Specifically, given a security parameter k and a data set $\mathbf{X}$ consisting of n records with p attributes each, the microaggregation problem consists in obtaining a partition of $\mathbf{X}$, such that each group in the partition has at least k records and the within-groups homogeneity is maximized. The resulting partition is called k-partition [5]. Once the k-partition is built, a microaggregated data set $\mathbf{X}'$ can be obtained by replacing each record in $\mathbf{X}$ by the centroid (*i.e.* the average vector) of the group to which it belongs. This clearly anonymizes original records, because all microaggregated records within a group are the same. The requirement of maximum within-groups homogeneity in the original data set is intended to minimize the information loss caused by microaggregation, that is, by the replacement of the original records in $\mathbf{X}$ by the centroids of their groups.

There are several group homogeneity measures based on different distance definitions (*e.g.* Euclidean distance, Chebyshev distance, Minkowski distance, etc.). The most common homogeneity measure for clustering is the within-groups sum of square errors (SSE) [8, 22, 10, 11, 16]. This is the sum of squared Euclidean distances from the centroid of each group to every element in the group. For a given k-partition, the SSE is computed as:

$$SSE = \sum_{i=1}^{s} \sum_{j=1}^{n_i} (\mathbf{x}_{ij} - \bar{\mathbf{x}}_i)'(\mathbf{x}_{ij} - \bar{\mathbf{x}}_i) \qquad (1)$$

where s is the number of groups in the k-partition, n_i is the number of elements in the i-th group, $\mathbf{x}_{ij}$ is the j-th element in the i-th group and $\bar{\mathbf{x}}_i$ is the centroid of the i-th group.

Given a data set, the optimal k-partition is the one with minimum SSE. The optimal k-partition of a univariate data set can be found in polynomial time [12]. However, finding the optimal k-partition for a multivariate data set is NP-hard [17]. Thus, multivariate microaggregation —*i.e.* aiming at minimizing the SSE of multivariate data— must be heuristically tackled. However, it is known that groups in an optimal k-partition contain between k and $2k-1$ records [5] and heuristics make use of this important knowledge. Multivariate microaggregation heuristics in the literature fall into two categories:

Fixed-size All groups in the resulting k-partition are of size k except perhaps one, which is of size between k and $2k-1$. Heuristics in this class can be found in [2, 5, 14, 7, 9].

Variable-size All groups can have sizes varying between k and $2k-1$, depending on the natural grouping of the data. Heuristics in this class include [5, 18, 15, 9, 4, 20, 21]. They are normally slower than fixed-size heuristics.

1.1 Contribution and plan of this paper

In this paper we show how to construct a hybrid heuristic for multivariate microaggregation offering the lower information loss caused by variable-size microaggregation and taking advantage of the speed of fixed-size microaggregation. The idea is to combine a fixed-size heuristic with the genetic microaggregation algorithm in [21] by these authors.

Our proposal can be divided in two main steps, namely group generation and group refinement. During the first step we apply the fixed-size heuristic to the original data set in order to obtain a number of groups with a cardinality not less than k. In the second step, we use the genetic algorithm on each group to refine it and we finally obtain a k-partition of the original data set.

Thanks to the genetic algorithm, our technique improves the results of the fixed-size heuristic in terms of within-groups homogeneity. Although our construction works with any fixed-size heuristic, empirical results in this paper have been computed for the specific case of MDAV [14, 7], arguably the most broadly used heuristic of this class.

The rest of the article is organized as follows. Section 2 recalls the MDAV heuristic and the main idea of our approach to solving microaggregation with a genetic algorithm. Section 3 describes how to combine the genetic algorithm with a fixed-size heuristic to obtain better results. Section 4 reports some experimental results for the specific case of MDAV. Finally Section 5 contains some concluding remarks and lines for future work.

2 Background

In this section we recall the MDAV heuristic and our approach to solving microaggregation using a genetic algorithm.

2.1 The MDAV heuristic

MDAV is a multivariate microaggregation method that was implemented as part of the μ-Argus [14] package for statistical disclosure control; its description can be found in [7]. Algorithm 1 is a short description of MDAV.

The goal of any microaggregation method is to generate a microaggregated data set protecting the privacy of the respondents. To do so, it is necessary to find a k-partition which maximizes within-groups homogeneity. MDAV builds a k-partition as follows. First, a square matrix of distances between all records is computed (line 1 of the algorithm). Two main approaches can be adopted to perform these distance calculations. The first approach is to compute and store the distances at the beginning of the microaggregation process. The second approach consists in computing the distances on the fly when they are needed. The first approach is computationally cheaper but it requires too much memory when the number of records in the data set is large. In this paper, we have used the first approach. However, the second one could be used as well without affecting the proposed technique nor the reported results.

After computing the matrix of distances, MDAV iterates (lines 4-13) and builds two groups at each iteration. In order to build these groups, the centroid c —*i.e.* the average vector— of the remaining records —those which have not yet been assigned to any group— is computed at the beginning of each iteration (line 5). Then the most distant record r from c is taken (line 6) and a group of k records is built around r (line 8). The group of k records around r is formed by r and the $k-1$ closest records to r. Next, the most distant record s from r is taken (line 7) and a group of k records is built around s (line 10). The generation of groups continues until the number of remaining records (RR) is less than $2k$. When this condition is met, two cases are possible, namely $RR < k$ or $RR \geq k$. In the first

Algorithm 1: Maximum Distance to Average Vector (MDAV) algorithm

```
   Data: Dataset $\mathbf{X}$, Integer k, Integer n.
   Result: $k$-partition.
1  ComputeDistanceMatrix($\mathbf{X}$);
2  $RR$ = $n$; //Remaining Records
3  $i$ = 0;
4  while $RR < 2k - 1$ do
5      $c$ = ComputeNewCentroid($\mathbf{X}$);
6      $r$ = GetFarthestRecordFromCentroid($\mathbf{X}$,$c$);
7      $s$ = GetFarthestRecordFromRecord ($\mathbf{X}$,$r$);
8      $g_i$ = BuildGroupAroundRecord($r$,$\mathbf{X}$,$k$);
9      $i = i + 1$;
10     $g_i$ = BuildGroupAroundRecord($s$,$\mathbf{X}$,$k$);
11     $i = i + 1$;
12     $RR = RR - 2k$;
13 end
14 $g_1 \ldots g_s$ = AssignRemainingRec($\mathbf{X}$, $g_1 \ldots g_s$);
15 return $g_1 \ldots g_s$
```

case, the remaining records are assigned to their closest group. In the second case, a new group is built with all the remaining records. Note that all groups have k elements except perhaps the last one. Thus, the generated k-partition is a candidate to optimality because it fulfills the necessary condition that the cardinality of all its groups is between k and $2k - 1$.

Finally, given the k-partition obtained by MDAV, a microaggregated data set is computed by replacing each record in the original data set by the centroid of the group to which it belongs.

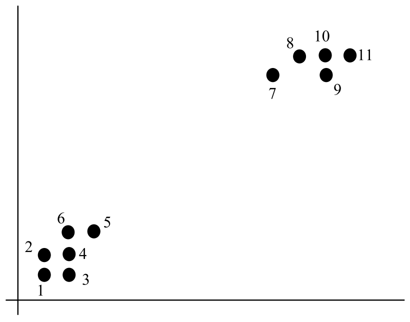

Figure 1. Example bivariate data set to illustrate the problem of MDAV.

Although the k-partition output by MDAV meets the necessary optimality condition on group cardinalities, it can be sometimes quite far from the actual optimum. In Figure 1 an example illustrating this problem is shown. For the given example, MDAV returns the 3-partition $\{\{1,2,3\},\{4,5,6,7,8\},\{9,10,11\}\}$. Clearly, the resulting 3-partition is not optimal in terms of within-groups homogeneity: an optimal k-partition would be $\{\{7,8,9,10,11\},\{4,5,6\},\{1,2,3\}\}$. It becomes apparent that the lack of flexibility in the size of the groups splits "natural" groups. As a result, the obtained k-partition is far from optimal.

2.2 A genetic algorithm for multivariate microaggregation

Our approach to microaggregation corresponds to a common Genetic Algorithm (GA) scheme [13] with some modifications to adapt it to the microaggregation task on multivariate data sets. Further details on those adaptations can be found in [21]. Specifically, our fitness function is

$$F = \frac{1}{SSE + 1} \quad (2)$$

where SSE is computed using Expression 1. Our GA uses the roulette wheel selection algorithm, which is based on weighing the probability of selecting a chromosome proportionally to its fitness. We have used the most common genetic operators (*i.e.* simple one-point crossover and mutation).

Following the recommendations of [21], we use the values of Table 1 for the main parameters of the GA.

Table 1. Values for the main GA parameters

Chromosomes	Mut.Rate	Cross.Rate	Iterations
10	0.1	0.5	10^4

3 Our method

As shown in [21], the GA outperforms fixed-size heuristics in terms of SSE when applied to very small data sets. Thus, our method consists of: i) partitioning the original data set using a fixed-size heuristic; ii) applying the GA on the groups output by the fixed-size heuristic. It is not straightforward to determine how to partition the original data set in order to obtain groups that are suitable for GA-based optimization. Next, we analyze two different partition strategies.

3.1 One-step partitioning

A simple approach for partitioning a data set with n elements is as follows:

1. Let k be a small value, (*e.g.* $k = 3$).
2. Let K be larger than k and small enough to be suitable for our GA with a reasonable number of iterations (*e.g.* $K = 20$).
3. Use a fixed-size heuristic to build a K-partition.

Finally, the GA is applied to each group of the K-partition in order to obtain an optimal or near optimal k-partition.

One-step partitioning has an important shortcoming, though. Running a fixed-size heuristic to get groups of cardinality K may split natural groups which would be better preserved if a lower cardinality parameter k was used. Figure 2 shows, on the left, the groups obtained using MDAV with $K = 24$. It can be observed that some nearby records are assigned to different groups. Hence, using one-step partitioning prevents the GA from finding solutions with a better SSE.

3.2 Two-step partitioning

With the aim of averting the shortcomings of the one-step partitioning method, we propose an alternative method that uses the fixed-size heuristic twice rather than once. With the proposed two-step partition method, the number of split natural groups is significantly reduced.

The proposed method can be summarized as follows:

1. Let k be a small value (*e.g.* $k = 3$).
2. Let K be larger than k and divisible by k, small enough to be suitable for our GA (*e.g.* $K = 21$).
3. Use a fixed-size heuristic to build a k-partition.
4. Taking as input records the average vectors obtained in the previous step, apply the fixed-size heuristic to build *macrogroups* (*i.e.* sets of average vectors) of size K/k.
5. For each given *macrogroup*, replace the average vectors by the k original records to obtain a K-partition.

Finally, apply the GA to each macrogroup in the K-partition in order to generate an optimal or near optimal k-partition of the macrogroup. The composition of the k-partitions of all macrogroups yields a k-partition for the entire data set. An important issue when using the GA on a macrogroup is to include the fixed-size k-partition of the macrogroup induced by the k-partition of the entire data set computed in Step 3 above as one of the chromosomes of the initial population fed to the GA.

Figure 2 shows, on the right, the groups built using the two-step partitioning method with $K = 24$ and $k = 3$. It can be observed that the resulting groups are better than those obtained with the one-step partitioning method. The following result holds:

Lemma 1 *The SSE of the k-partition obtained using a two-step partition based on a fixed-size heuristic followed by a GA is not greater than the SSE of the k-partition output by the fixed-size heuristic alone.*

Proof: By assumption, the fixed-size k-partition of each macrogroup induced by the fixed-size k-partition of the data set obtained at Step 3 is one of the chromosomes of the initial population fed to the GA. Therefore, for each macrogroup, the k-partition output by the GA is at least as good as the fixed-size k-partition in terms of the fitness function. This implies that the SSE of the k-partition of the entire data set obtained by composition of the macrogroup k-partitions output by the GA is not greater than the SSE of the k-partition of Step 3. $\Box$.

4 Experimental results

In this section we summarize the empirical results obtained with our hybrid technique. MDAV has been used as a fixed-size heuristic. The two partitioning methods described above and the GA have been tested on four different data sets, each of them presenting particular features in terms of data dispersion and natural grouping.

The data sets used are next described:

Synthetic data sets "Scattered" and "Clustered" are data sets that contain $n = 1000$ records with $p = 2$ attributes. The former is *scattered* in the sense that no natural clusters are apparent. Attribute values were drawn from the $[-10000, 10000]$ range by simple random sampling. The second data set is considered to be *clustered* because its records are grouped in natural clusters. The attribute values were uniformly drawn from $[-10000, 10000]$ as in the previous case, but records were then groupwise shifted by a random amount in order to form clusters of size between 3 and 5 records each; more details on the generation of this data set can be found in [4].

Real data sets "Census" is a data set that contains 1080 records with 13 numerical attributes. "EIA" contains 4092 records with 11 numerical attributes. These data sets were proposed as reference microdata sets during the CASC project [1] and have been widely used [5, 6, 7, 15].

Table 2 shows the results of running the one-step and two-step partitioning methods on the aforementioned data sets. Microaggregation has been performed with $k = 3$,

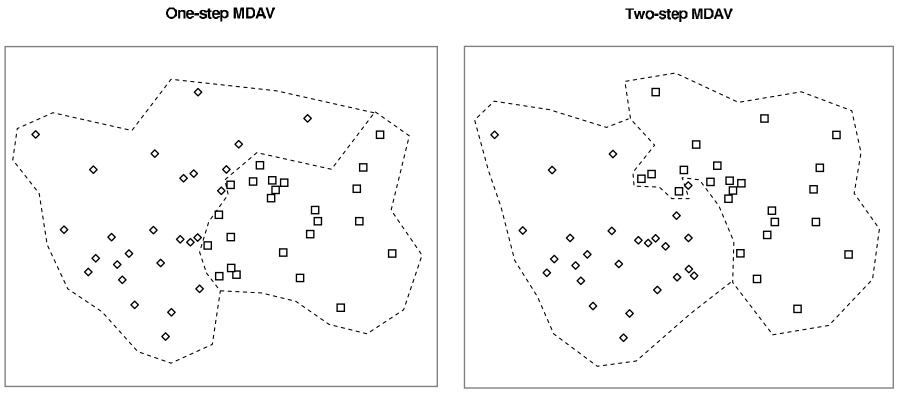

Figure 2. Left, one-step MDAV partitioning of the data ($K = 24$). Right, two-step MDAV partitioning of the data ($K = 24$, $k = 3$).

which is a common choice. Different values of K have been tested. Results comprise the number of macrogroups generated as well as the SSE of the microaggregated data sets using both MDAV on the whole set and GA on the macrogroups. We have used the GA parameters given in Table 1. Using these parameters, we have run our C++ GA implementation on a macrogroup of 27 elements, and it takes 2 seconds on a Pentium 4 processor running at 3 GHz under a Linux operating system.

It can be observed in Table 2 that, in general, the results of the one-step partitioning method are not better than the results yielded by the MDAV heuristic alone. Except for the "Scattered" data set with $K = 27$ and the "Clustered" data set, the breaking of several natural clusters by the one-step partitioning approach decreases the within-groups homogeneity of the microaggregated data set (increasing SSE). On the other hand, it can be noticed that in all cases the SSE of the data sets microaggregated using our GA after partitioning the data with the two-step partitioning method are better than the SSE obtained using MDAV only. The improvements achieved range from 1% to 41%.

5 Conclusions and future work

Microaggregation is a technique used to protect privacy in databases and location-based services. We have proposed a new hybrid technique for multivariate microaggregation that combines a heuristic yielding fixed-size groups and a genetic algorithm yielding variable-sized groups. This mixture blends the low information loss (high within-groups homogeneity) of the genetic approach with the ability to cope with data sets of realistic size (typical of fixed-size heuristics).

The empirical work presented shows that, in the case of the MDAV heuristic, the hybrid approach described can reduce information loss by as much as 41%.

Future work will include tuning the parameters of the hybrid heuristic described to make it fit for applications other than statistical database privacy. Specifically, we plan to adapt it to location privacy. Finally, a comparison of the proposed heuristic with genuine variable-size algorithms mentioned in Section 1 is also under way.

Acknowledgments

This work was partly supported by the Spanish Ministry of Education through project SEG2004-04352-C04-01 "PROPRIETAS" and by the Government of Catalonia under grant 2005 SGR 00446.

References

[1] R. Brand, J. Domingo-Ferrer, and J. M. Mateo-Sanz. Reference data sets to test and compare SDC methods for protection of numerical microdata, 2002. European Project IST-2000-25069 CASC, http://neon.vb.cbs.nl/casc.

[2] D. Defays and N. Anwar. Micro-aggregation: a generic method. In *Proceedings of the 2nd International Symposium on Statistical Confidentiality*, pages 69–78, Luxemburg, 1995. Eurostat.

[3] J. Domingo-Ferrer. Microaggregation for database and location privacy. In *Next Generation Information*

Table 2. Summary of experimental results ($k = 3$)

			One-step MDAV partitioning			Two-step MDAV partitioning		
Data set	K	SSE_{MDAV}	# clus.	SSE	% impr.	# clus.	SSE	% impr.
“Scattered”	12	4.71	83	5.11	-8%	83	4.28	**9%**
	18	4.71	55	4.91	-4%	56	4.33	**8%**
	27	4.71	37	4.71	0%	38	4.58	**3%**
“Clustered”	12	3.57	83	2.26	37%	84	2.84	**20%**
	18	3.57	55	1.83	49%	56	2.14	**40%**
	27	3.57	37	1.18	67%	41	2.09	**41%**
“Census”	12	799	90	865.04	-8%	91	768	**4%**
	18	799	60	879.94	-10%	61	767	**4%**
	27	799	40	919.88	-15%	41	789	**1%**
“EIA”	12	217	341	414.56	-91%	342	189	**13%**
	18	217	227	295.78	-36%	228	186	**14%**
	27	217	151	333.26	-54%	152	197	**9%**

Technologies and Systems-NGITS'2006, volume 4032 of *Lecture Notes in Computer Science*, pages 106–116, Berlin, 2006.

[4] J. Domingo-Ferrer, A. Martínez-Ballesté, J. M. Mateo-Sanz, and F. Sebé. Efficient multivariate data-oriented microaggregation. *The VLDB Journal*, 15(4):355–369, 2006.

[5] J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.

[6] J. Domingo-Ferrer, F. Sebé, and A. Solanas. A polynomial-time approximation to optimal multivariate microaggregation. *Manuscript*, 2005.

[7] J. Domingo-Ferrer and V. Torra. Ordinal, continuous and heterogenerous k-anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11(2):195–212, 2005.

[8] A. W. F. Edwards and L. L. Cavalli-Sforza. A method for cluster analysis. *Biometrics*, 21:362–375, 1965.

[9] E. Fayyoumi and B. J. Oommen. A fixed structure learning automaton micro-aggregation technique. In J. Domingo-Ferrer and L. Franconi, editors, *Privacy in Statistical Databases-PSD 2006*, volume 4302 of *Lecture Notes in Computer Science*, pages 114–128, Berlin, 2006.

[10] A. D. Gordon and J. T. Henderson. An algorithm for euclidean sum of squares classification. *Biometrics*, 33:355–362, 1977.

[11] P. Hansen, B. Jaumard, and N. Mladenovic. Minimum sum of squares clustering in a low dimensional space. *Journal of Classification*, 15:37–55, 1998.

[12] S. L. Hansen and S. Mukherjee. A polynomial algorithm for optimal univariate microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1043–1044, July-August 2003.

[13] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[14] A. Hundepool, A. V. de Wetering, R. Ramaswamy, L. Franconi, A. Capobianchi, P.-P. DeWolf, J. Domingo-Ferrer, V. Torra, R. Brand, and S. Giessing. *μ-ARGUS version 4.0 Software and User's Manual*. Statistics Netherlands, Voorburg NL, may 2005. http://neon.vb.cbs.nl/casc.

[15] M. Laszlo and S. Mukherjee. Minimum spanning tree partitioning algorithm for microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):902–911, 2005.

[16] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[17] A. Oganian and J. Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal of the United Nations Economic Comission for Europe*, 18(4):345–354, 2001.

[18] G. Sande. Exact and approximate methods for data directed microaggregation in one or more dimensions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):459–476, 2002.

[19] J. Schlörer. Disclosure from statistical databases: quantitative aspects of trackers. *ACM Transactions on Database Systems*, 5:467–492, 1980.

[20] A. Solanas and A. Martínez-Ballesté. V-MDAV: Variable group size multivariate microaggregation. In *COMPSTAT'2006*, pages 917–925, Rome, 2006.

[21] A. Solanas, A. Martínez-Ballesté, J. M. Mateo-Sanz, and J. Domingo-Ferrer. Towards microaggregation with genetic algorithms. In *3rd IEEE Conference On Intelligent Systems*, pages 65–70, Westminster, 2006. IEEE Computer Society Press.

[22] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.