Tablas multidimensionales en R

Juan Gómez Duaso

Instituto Galego de Estatística (IGE)

Resumen

En esta ponencia se muestra la experiencia del IGE en la producción de tablas multidimensionales de datos agregados utilizando el lenguaje R.

Se comenta en primer lugar el uso creciente en el IGE de R como plataforma para el desarrollo de soluciones para las distintas fases del proceso estadístico, y las ventajas que presenta respecto a otros lenguajes y entornos.

A continuación se repasa el diseño del banco de datos multidimensionales del Instituto, que representa el grueso de nuestra producción de datos para la difusión. Tradicionalmente las tablas multidimensionales que alberga (*hipertablas*) se venían produciendo de forma artesanal en Excel o, más comúnmente, mediante la plataforma de base de datos OLAP del Instituto (MS-SQLServer). Ahora pueden ser producidas también directamente en R a partir de la tabulación de dataframes de microdatos en un proceso dirigido por los metadatos (jerarquías de clasificación de las distintas dimensiones a cruzar) mediante una función general de cruce que produce resultados multidimensionales, representados en una clase de objetos de R que son una representación de las hipertablas en objetos de R.

La principal ventajas derivada de tener las hipertablas en R es que los datos pueden ser manipulados de diversas formas antes de su grabación en los bancos de datos de difusión. Esto incluye tanto tratamientos de cálculo (tasas, porcentajes, etc.) como tratamientos más orientados a la estructuración de las hipertablas para la presentación (composición de partes, filtrado de elementos, indentaciones jerárquicas etc.).

Se comenta la experiencia en el uso de arrays de R como el tipo de datos que a priori es más natural para representar tablas multidimensionales, sus limitaciones y la solución que se le ha dado a estas. Una parte de la funcionalidad hecha para el manejo de nuestras hipertablas se puede generalizar, de modo que resulten funciones útiles para el manejo de arrays en R, dando un entorno alternativo para programar cálculos.

Palabras clave

Tablas multidimensionales, R, arrays

1 Introducción

En esta ponencia vamos a centrarnos en el sistema de producción de datos, en el nivel instrumental o informático en el que el sistema estadístico se puede ver como una fábrica de datos, transformando la información de partida de censos, encuestas y registros administrativos en los agregados que nutren las tablas y gráficos que difundimos.

Esta cadena de producción se asienta sobre una serie de componentes y divide el proceso en fases que van desde la captura de los microdatos hasta su difusión. El proceso puede ser más o menos común para las distintas operaciones e incluso seguir un modelo formal estándar (SGBPM). No es aun nuestro caso en el IGE, aunque siempre hemos tratado al menos de buscar la unificación, con entornos y piezas comunes para todas las operaciones en la medida de lo posible.

En el Instituto tenemos desde hace tiempo asentada una serie de componentes que configuran la vía principal de producción, y algunas variantes o alternativas a ella pero que tenían un carácter secundario. En sus fases centrales de tratamiento y transformación de datos, esta vía principal está basada en la plataforma de base de datos de Microsoft, con las bases de datos de MS-SQLServer. En un primer escalón la relacional sobre la que los datos individuales son grabados, editados, depurados e imputados. A continuación la multidimensional OLAP (SQLServer Analysis Services) con cubos OLAP encargados de la agregación y los cálculos.

A continuación, nuestra difusión se basa en varios bancos de datos que nutren a la web del instituto *www.ige.eu*. El principal de ellos y que representa el grueso de nuestra producción de datos para la difusión es el banco de tablas multidimensionales, que es el único del que vamos a hablar.

Junto a estos componentes siempre ha estado el software estadístico y matemático (SAS, SPSS, MATLAB) para tareas de análisis.

Más recientemente el IGE ha ido introduciendo y generalizando el uso del lenguaje R para cada vez más fases y tareas del proceso, y no solo analíticas, hasta el punto de que hoy es una vía alternativa al proceso principal en SQLServer y la preferente para los nuevos tratamientos de datos.

Existen razones técnicas a favor de este nuevo entorno, como veremos, pero no ha sido ajena a esta decisión otras dos:

- la crisis económica y la consiguiente necesidad de abaratar costes, en este caso explotando la acepción, que no por errónea es menos real, de *free software* (software gratis) que tiene R.

- la evolución del personal estadístico, desde su formación y cometidos tradicionales hacia la mezcla entre competencia estadística y TIC que hoy se entiende por el rol del científico de datos.

La parte del proceso estadístico que nos interesa en esta ponencia es la del final del tratamiento, donde se producen y procesan los datos agregados, y se dan los retoques necesarios para su difusión. Vamos a describir como estamos empezando a hacer esos tratamientos en R.

Pero antes veremos las razones que nos animaron a hacerlo y después necesitamos conocer algo más como es el banco de datos donde deben acabar los resultados.

OLAP vs R

Los modelos multidimensionales OLAP resultan muy adecuados para la representación de los datos estadísticos. Contemplan el fichero de microdatos con su concepto similar de las tablas de hechos. Las clasificaciones estadísticas, con sus jerarquías, caben en lo que son en OLAP las tablas de dimensiones y sus múltiples niveles. También el tratamiento del tiempo como un eje especial resulta adecuado en nuestro contexto. Los sistemas multidimensionales gestionan la agregación de los datos, quizá aquí no tan sofisticada como se necesita en la estimación que hacemos en estadística, por lo que a veces requiere algún desarrollo adicional a lo que el sistema proporciona directamente. Finalmente los cubos se ofrecen para la consulta y visualización con herramientas diversas y con APIs para construir soluciones de publicación a medida, como hemos hecho en nuestro caso en el IGE.

Quizá la mayor disonancia entre este modelo y nuestras necesidades está en una cuestión de escala. El modelo OLAP nace para la analítica de los datos propios de una empresa. Típicamente van a ser unos pocos cubos, muy elaborados en sus metadatos (niveles, medidas, miembros calculados, fórmulas) y con un alto volumen de datos. En la estadística oficial, tradicionalmente nos enfrentamos a cientos de ficheros estadísticos distintos, no tan grandes y muchas veces no tan detallados. La estructura de la información no siempre podemos determinarla nosotros como si puede hacer la empresa con sus cubos, sino que se adapta a las fuentes y los cambios que se le hagan si es un fichero administrativo externo o a los cambios de diseño de cuestionario derivados de las nuevas demandas de información si se trata de una encuesta propia.

Esto lo hemos experimentado en el IGE a lo largo de los años. Los cubos de SQLServer, resultan capaces de contener los datos y metadatos necesarios de una forma bastante completa, aunque algunos temas como el idioma de las etiquetas o algunos tipos de tablas complejas y cálculos han necesitado software y metadatos adicionales a la base que ofrece el gestor de la base de datos como tal. Las tablas se producen principalmente definiendo consultas sobre los cubos de SQLServer que son procesadas periódicamente por nuestro software de gestión del banco de datos. Las operaciones más estables y voluminosas son las que obtienen mejores resultados, porque justifican la relativa rigidez de los cubos y el esfuerzo inicial de su definición.

R es ante todo una herramienta estadística, con miles de paquetes y funciones para tratamientos estadísticos. Pero también, y especialmente con la introducción de paquetes especializados como *dplyr* (y todos los paquetes de los mismos autores que lo acompañan, formando un conjunto especializado en el tratamiento de la información, en el denominado *tidyverse*[1]), resulta un buen entorno para los procesos de manipulación y transformación de los datos. Las herramientas de consulta multidimensional de SQLServer (MDX) pueden ser más sencillas mientras nos

[1] www.tidyverse.org

mantengamos dentro de los límites para los que están previstas, pero en caso contrario, nos enfrentan a una barrera que solo podemos superar recurriendo a la programación sobre su modelo de objetos que es complicado y en nuestro caso solo al alcance del personal informático. Por el contrario, los scripts en R presentan un nivel de dificultad quizá mayor inicialmente pero es escalable gradualmente hacia mayor complejidad y siempre al alcance del técnico estadístico. Y aunque no aporta todas las capacidades de serie que tienen los cubos OLAP, ofrece un entorno en el que poder emularlas y donde el problema de escala pueda encontrar la solución. Por otra parte, la programación en R también resulta más flexible que en SQL, o que el empleo de los mecanismos de chequeos y restricciones del gestor relacional.

No sabemos aún el resultado final de esta dualidad de entornos para la transformación de datos. Es posible que R se acabe decantando en el papel de ser un complemento del tronco principal que seguirá basado en cubos OLAP, focalizándose en proyectos nuevos o modelos de datos menos complejos o más inestables.

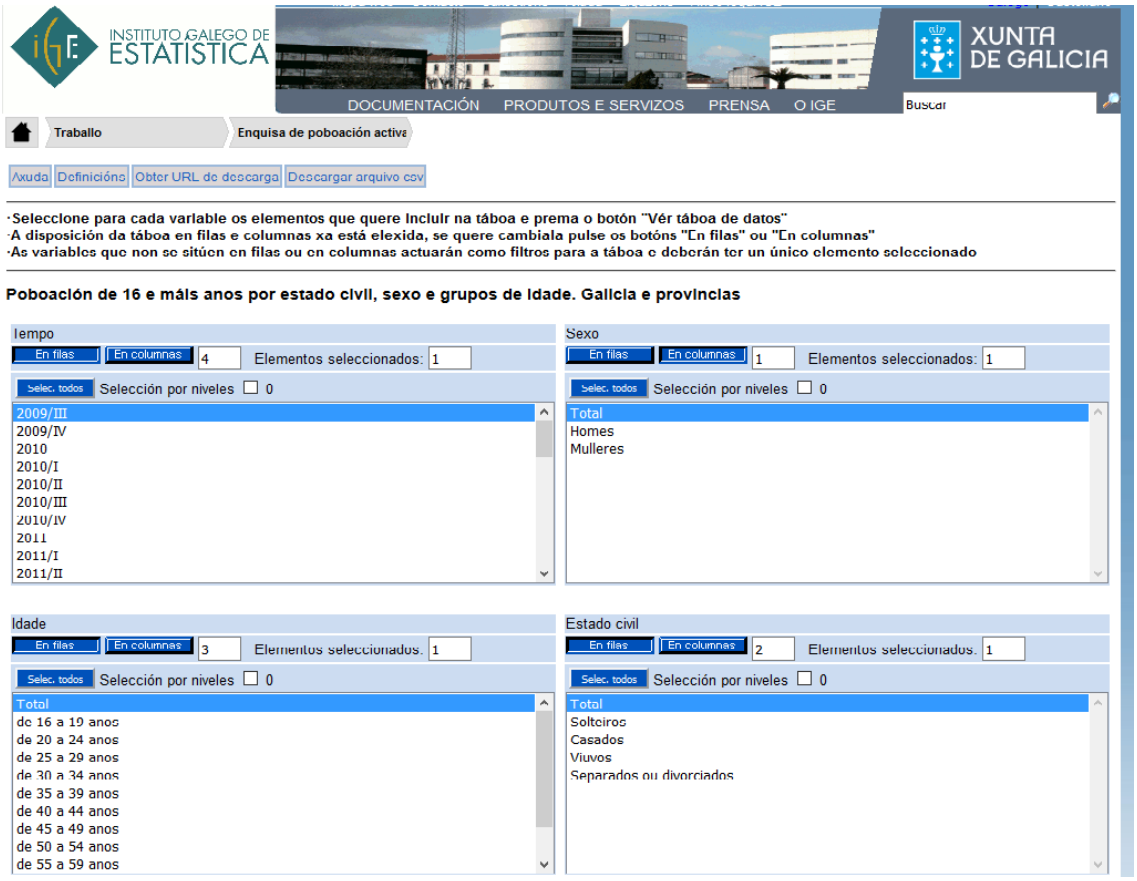
Pero cabe también la posibilidad de que si los resultados que se vayan obteniendo son lo suficientemente alentadores en esos proyectos y si nuestro ritmo de construcción de funciones para enriquecer este nuevo ecosistema es alto, podría llegar a sustituir por completo al modelo precedente. En esa dirección empujaran también dos factores importantes:

- la conveniencia de que siempre es mejor tener un sistema unificado sino hay razones de peso que lo desaconsejen.

- El carácter de software libre de R y el hecho de ser software propio lo que vayamos construyendo sobre él, y la consiguiente independencia y perdurabilidad que esto garantiza. Algo vital cuando hablamos de piezas trocales de nuestro “proceso de negocio”, que deberían sujetarse lo menos posible a los cambios de características e incluso de paradigmas y modelos que acompañan necesariamente al software comercial y sus ciclos de obsolescencia, y del que Microsoft nos ha dado alguna muestra también, aunque no excesiva, a lo largo de los años de uso de sus soluciones OLAP.

El banco de datos de difusión del IGE

Se trata de una base de datos para almacenar las tablas multidimensionales de difusión (que denominamos *hipertablas*) mediante un objeto de diseño propio. Estas tablas de macrodatos son similares a las que son comunes en cualquier organismo de la estadística oficial: tablas numéricas de múltiples ejes o dimensiones cuyos miembros actúan como coordenadas de las celdas de información de las variables multidimensionales. Similares también a los modelos estandarizados en estadística como Pcaxis o Jsonstat.



Quizá la principal peculiaridad de las hipertablas, que las distingue de algunos otros modelos, sea que sus dimensiones admiten una jerarquía entre sus miembros. Por poner un ejemplo, es posible tener una hipertabla cuyo eje de “espacio” sea Galicia, sus provincias y sus municipios. Esto hace a la hipertabla en cierto modo similar a un pequeño cubo multidimensional, en realidad es básicamente un cubo, aunque lo suficientemente agregado para ser publicable. Por eso tiene sentido hacer sobre ella algunas de las operaciones típicas de OLAP. Por ejemplo el *drill down* para profundizar y ver más detalles, tal como tenemos implementado en el interfaz de consulta en la Web. O el *roll up*, la agregación en las jerarquías, del que hemos visto un ejemplo de su utilización en la ponencia de mi compañera Esther en estas JECAS [2].

Las hipertablas tienen sus propias clasificaciones, no las comparten. Esta decisión, quizá no optima, ha resistido sin embargo muy bien el paso de los años.

Solo el espacio es una dimensión compartida y además basada en clasificaciones fijas. El tiempo también tiene una estandarización común de los valores (aunque aquí no se imponen clasificaciones fijas ya que los intervalos de tiempo o las periodicidades de las encuestas son muy variables y no siempre sistemáticos). La ventaja que tiene esta decisión es la flexibilidad: una hipertabla puede representar casi cualquier tabla que se nos ocurra, incluso mezclando variables o concatenando clasificaciones en un mismo eje, sin restricciones.

[2] Esther López Vizcaíno: El reto del control de la calidad de la información estadística difundida mediante técnicas de Big Data

Cruces en R

Al hablar de la agregación podemos referirnos a dos cosas: la agregación por condensación de la tabla de microdatos respecto de algunas de sus variables o la tabulación basada en unas clasificaciones a priori.

En el primer caso entenderíamos la agregación como un proceso para dar lugar a un nuevo conjunto de datos más resumido, en el que las filas con valores iguales en todas ellas se habrán colapsado, agregadas, en una sola. Lo habitual será representarlo como un dataset, de la misma forma que los datos individuales, en tablas SQL, dataframes de R etc; con una (o más) columna numérica para los agregados y otras para las clasificaciones respecto a las que se han tabulado. Esto tiene la ventaja de la homogeneidad, con el conjunto de datos como único contenedor, sin distingos entre los diversos niveles de agregación. Lo que caracteriza a esta forma de agregación es que en este resultado solo aparecerán los cruces de características de clasificación que se den efectivamente en los datos, y lo harán en el orden en que se produzcan al procesar la agregación, sin que este tenga ningún significado. Se trata de una agregación determinada únicamente por la tabla de datos.

En el segundo caso el proceso de agregación se puede entender como la producción del cruce exhaustivo de una serie de dimensiones preestablecidas e independientes del conjunto de datos a agregar, produciendo así unas nuevas variables multidimensionales, definidas sobre el producto cartesiano de las dimensiones. Es una agregación dirigida por los metadatos.

Por supuesto para las necesidades de la difusión en la estadística oficial hemos de aplicar este segundo enfoque ya que nuestro objetivo es producir cruces.

Hay también casos de información multidimensional no proveniente de la agregación, por ejemplo los microdatos de un estudio con un diseño muestral de experimentación en el que se mide sobre el cruce exhaustivo de unos factores varias características. Pero sea de un tipo u otro, en la información multidimensional:

- Se tiene toda la información del cruce exhaustivo de las clasificaciones

- Las clasificaciones tienen un conjunto predefinido y ordenado de elementos

- No hay repeticiones de datos para un mismo cruce de clasificadores

Cuando estamos ante una variable multidimensional, además de almacenarla como un conjunto de datos, tenemos también la posibilidad de recurrir a una estructura multidimensional de algún tipo: arrays, cubos OLAP, modelos específicos para estadística (pcaxis, jsonstat), etc.

El modelo de acceso directo multidimensional es más eficaz:

- En economía de espacio: evitando la redundancia de los mismos clasificadores, innecesarios si se mantienen las celdas en orden y no penalizando demasiado por las, posibles pero probablemente escasas, celdas vacías que debe almacenar

- En velocidad de proceso: por accesos directos a las posiciones fijas de las celdas ordenadas, sin necesidad de búsquedas ni índices.

Estas ventajas podrían ser determinantes si la información es muy voluminosa, pero no lo son, sin embargo, en el caso que nos interesa de la información agregada para difundir, ya que su tamaño limitado hace que tengan prioridad otras consideraciones como la facilidad de trabajo y de expresar los cálculos.

Centrándonos en el caso concreto de R, a partir de un dataframe de microdatos, hay funciones que producen otro dataframe agregado según algunas de sus columnas. El enfoque es claramente de conjunto de datos: los resultados aparecen por el orden de aparición de los elementos de cada clasificación y no se garantiza que aparezcan valores de la clasificación que no estén presentes en los microdatos, o que no aparezcan determinados cruces si no hay caso en ellos. A este grupo pertenecen tanto el operador GROUP BY de SQL como funciones de R base como *aggregate*() o incuso de paquetes recientes como la función *sumarize*() de *dplyr*.

Por otro lado, tenemos la función *tapply* de R base como el ejemplo más claro del modelo multidimensional:

- utiliza columnas de tipo factor para las clasificaciones: se trata de un tipo de datos que garantiza la exhaustividad y el orden de los miembros.

- produce como resultado un array (vector multidimensional)

Por ello y por lo dicho anteriormente, en el IGE hemos hecho nuestra función de cruce, *h.cruzar*(), basada en *tapply*() y con arrays como resultado por ser a priori la opción más natural, y hemos pasado a continuación a evaluar esta forma de trabajar con los agregados y paliar sus posibles inconvenientes.

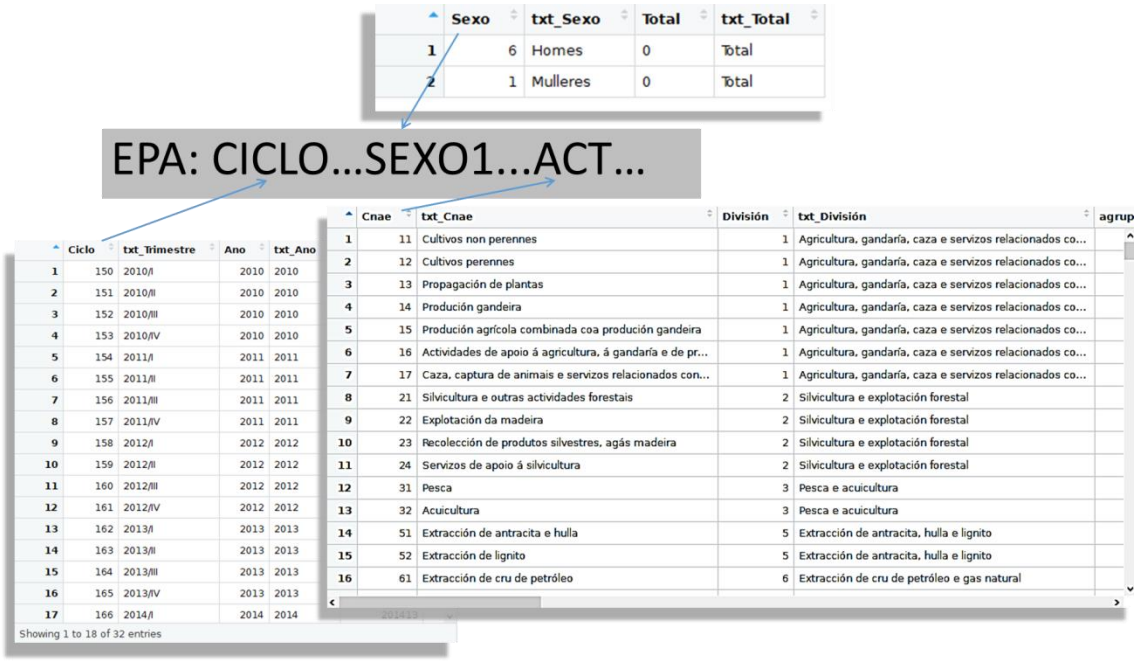
La función *h.cruzar*()

Se trata de una función de propósito general para la agregación. Se basa en los metadatos para guiar el funcionamiento, mediante una estructura en estrella similar al modelo multidimensional relacional (MOLAP). La tabla estadística a cruzar actúa como tabla de hechos y deberemos tener definidos, en torno a ella, una serie de dataframes que cumplen el rol de tablas de dimensiones. Estos siguen cierta convención de nombres y contienen las clasificaciones a varios niveles de detalle que configuran cada dimensión.

La convención de nombres no solo estandariza la estructura de este tipo de dataframes de dimensiones en todo el instituto, sino que además evita la necesidad de tediosas configuraciones. Aprovechamos aquí la gran flexibilidad de R a la hora de operar con los nombres de sus estructuras de datos, que no son sino un atributo de los objetos que podemos alterar en todo momento.

Con esta infraestructura establecida, los cruces se producen de forma declarativa, indicando que ejes, con que dimensiones y con qué niveles de las mismas queremos hacer la agregación. Indicamos además la variable a agregar y la función de agregación, normalmente suma o recuento, pero potencialmente cualquier función aportada por el usuario (medias, medianas, etc.). El resultado son las hipertablas ya completamente finalizadas, con sus múltiples ejes y con la posible jerarquía entre los miembros de los ejes que tengan varios niveles ya establecida.

Así por ejemplo, en la explotación de la EPA podemos tener dimensiones como:



	Sexo	txt_Sexo	Total	txt_Total
1	6	Homes	0	Total
2	1	Mulleres	0	Total

	Ciclo	txt_Trimestre	Ano	txt_Ano
1	150	2010/I	2010	2010
2	151	2010/II	2010	2010
3	152	2010/III	2010	2010
4	153	2010/IV	2010	2010
5	154	2011/I	2011	2011
6	155	2011/II	2011	2011
7	156	2011/III	2011	2011
8	157	2011/IV	2011	2011
9	158	2012/I	2012	2012
10	159	2012/II	2012	2012
11	160	2012/III	2012	2012
12	161	2012/IV	2012	2012
13	162	2013/I	2013	2013
14	163	2013/II	2013	2013
15	164	2013/III	2013	2013
16	165	2013/IV	2013	2013
17	166	2014/I	2014	2014

Showing 1 to 18 of 32 entries

	Cnae	txt_Cnae	División	txt_División	agrup
1	11	Cultivos non perennes	1	Agricultura, gandaría, caza e servizos relacionados co...	
2	12	Cultivos perennes	1	Agricultura, gandaría, caza e servizos relacionados co...	
3	13	Propagación de plantas	1	Agricultura, gandaría, caza e servizos relacionados co...	
4	14	Produción gandeira	1	Agricultura, gandaría, caza e servizos relacionados co...	
5	15	Produción agrícola combinada coa produción gandeira	1	Agricultura, gandaría, caza e servizos relacionados co...	
6	16	Actividades de apoio á agricultura, á gandaría e de pr...	1	Agricultura, gandaría, caza e servizos relacionados co...	
7	17	Caza, captura de animais e servizos relacionados con...	1	Agricultura, gandaría, caza e servizos relacionados co...	
8	21	Silvicultura e outras actividades forestais	2	Silvicultura e explotación forestal	
9	22	Explotación da madeira	2	Silvicultura e explotación forestal	
10	23	Recolección de produtos silvestres, agás madeira	2	Silvicultura e explotación forestal	
11	24	Servizos de apoio á silvicultura	2	Silvicultura e explotación forestal	
12	31	Pesca	3	Pesca e acuicultura	
13	32	Acuicultura	3	Pesca e acuicultura	
14	51	Extracción de antracita e hulla	5	Extracción de antracita, hulla e lignito	
15	52	Extracción de lignito	5	Extracción de antracita, hulla e lignito	
16	61	Extracción de cru de petróleo	6	Extracción de cru de petróleo e gas natural	

Y a partir de ahí, hacer cruces como por ejemplo:

```
tablaResultado=h.cruzar(EPA,
                        ejes=list(SEXO1=c("Sexo", "Total"),
                                  ACT=c("Cnae","División"),
                                  CICLO=c("Trimestre","Ano")),
                         var=EPA$FACTOREL)
```

Los resultados de h.cruzar(), como los de tapply(), producen arrays. Pero no solo eso, ya que en el proceso de cruce tenemos también los clasificadores, las cabeceras de nuestra tabla, provenientes

de los dataframes de dimensiones. Y tenemos también distintos niveles de profundidad en los miembros de los ejes si el cruce se ha hecho con varios niveles de una misma dimensión.

Así por ejemplo en el eje del tiempo (CICLO) tendremos los siguientes elementos y sus profundidades:

```
    TXTELEM PROF
1      2010    0
2    2010/I    1
3   2010/II    1
4  2010/III    1
5   2010/IV    1
6      2011    0
7    2011/I    1
8   2011/II    1
9  2011/III    1
10  2011/IV    1
etc.
```

Veremos a continuación como se guarda todo esto.

Hipertablas en R

Dado que hemos abierto una nueva ruta basada en R para la producción de tablas estadísticas, es necesario reconectarla al final con el circuito principal dotando a R de la capacidad de producir los resultados como hipertablas.

Podríamos habernos limitado a lo necesario: definir funciones para la grabación en la base de datos de difusión de los datos y metadatos producidos por *h.cruzar(*) en los términos admitidos por aquella. Después estos datos y metadatos podrían ser procesados y documentados con las herramientas que ya tenemos disponibles para trabajar con el banco.

Sin embargo se ha preferido llevar el entorno de R un poco más allá dentro del proceso estadístico, dándole ocasión para participar también en la fase de tratamiento de los macrodatos mediante la representación de las hipertablas en R.

Los motivos son dos:

- Explorar las ventajas de la participación de los técnicos estadísticos y su habilidad para la programación en R también en esta fase

- Utilizar tratamientos de cálculos y no solo de documentación o formateo sobre las hipertablas, algo que las aplicaciones del banco de datos contemplaban de una forma limitada y en lo que la programación directa con scripts y las capacidades de cálculo de R pueden ser muy adecuadas.

La base de la representación de las hipertablas serían los arrays de R, a priori el objeto más adecuado y el que nos proporciona directamente la función de agregación.

No es necesaria mucha argumentación para explicar la idoneidad estructural de un array como base de la representación de datos multidimensionales en R. Pero en cambio sí que se puede cuestionar la idoneidad de esta decisión si pensamos que R es en primer lugar un lenguaje de vectores, y después un lenguaje de dataframes y de matrices. En R los arrays resultan comparativamente menos habituales, cuentan con menos relevancia en el entorno estadístico (que es su razón de ser) respecto a otros ámbitos de la computación científica, y por ello tienen también menos funciones asociadas y menos facilidades para su manipulación que en otros lenguajes o entornos.

Quizá el aspecto más útil de esta ponencia sea mostrar la situación que hemos encontrado en el IGE al optar por la utilización de arrays como base de nuestros macrodatos en R.

Vectores reciclaje y arrays

En R no existen los valores atómicos como tales (entendiéndose como vectores de longitud 1), los vectores son la base del sistema de tipos. Las operaciones en R son vectorizadas: al operar dos vectores entre sí, se operan sus miembros correspondientes uno a uno de forma primitiva (y eficiente al realizarse en C y no en R interpretado), sin necesidad de bucles recorriendo su longitud. En caso de que las longitudes de los vectores no sean iguales, en lugar de producir un error, se reutiliza el de menor longitud desde el principio tantas veces como sea necesario hasta completar la longitud del mayor. Solo en el caso de que esta no sea múltiplo de la otra se produce un warning, dado que es posible que la operación no sea intencionada, pero se trata de un mero aviso y la operación se realiza igualmente.

Esta característica de reutilización (array recycling) es muy útil en la práctica. Sobre todo es la forma de operar un vector con un escalar (vector de longitud 1) reciclando el valor único de este para operarlo con todos los del primero. Interesante también, aunque con una utilidad más limitada es que en el caso de enfrentar dos vectores no atómicos, permite algunas construcciones cíclicas sin necesidad de crear previamente un patrón repetitivo. Por ejemplo aplicar unas ponderaciones a los días de la semana en una serie diaria no necesita más que los 7 factores una vez, que ya se repetirán automáticamente.

Sorprende por todo ello que, a la hora de operar con arrays en R, estos no tengan el mismo comportamiento “reciclador”.

Sí lo tienen cuando los operamos con un vector simple, dado que lo que se hace en ese caso es que se operan como vectores, con el reciclaje correspondiente, y se conserva la dimensión del array. Asi vemos que, por ejemplo:

```
> array(1:6,dim=2:3)*c(10,100)
     [,1] [,2] [,3]
[1,]   10   30   50
[2,]  200  400  600
```

nos da lo esperado (teniendo en cuenta que los arrays de R se van rellenando por columnas)

Pero en cambio:

```
> array(1:6,dim=2:3)*array(c(10,100),dim=2)
Error in array(1:6, 2:3) * array(c(10, 100), dim = 1) :
  non-conformable arrays
```

no funciona.

Y en este caso hay una razón, ya que cabe la duda de si la única dimensión del segundo se corresponderá con la primera o con la segunda del primer array (el que una mida 3 y la otra fortuitamente 2 como la del oponente no debería tenerse en cuenta, ya que podrían coincidir las dos).

Pero es que si hacemos:

```
> array(1:6,dim=2:3)*array(c(10,100),dim=2:1)
Error in array(1:6, dim = 2:3) * array(c(10, 100), dim = 2:1) :
  non-conformable arrays
```

O bien:

```
> array(1:6,dim=2:3)*array(c(10,100),dim=1:2)
Error in array(1:6, dim = 2:3) * array(c(10, 100), dim = 1:2) :
  non-conformable arrays
```

Tenemos el mismo error, a pesar de que ahora el número de dimensiones coincide y estamos indicando en que "sentido" (vertical u horizontal respectivamente) se coloca el segundo array.

La utilidad que tendría el reciclaje en los arrays sería la misma que se tiene con los vectores de R. Al menos sería muy útil tener el reciclaje más básico, que se produce al enfrentar dimensiones de longitud N y 1. Así lo han entendido librerías como por ejemplo Numpy, del lenguaje Python, que denominan a esta forma limitada de reutilización de una dimensión degenerada (de un solo elemento) *array broadcasting* [3]

En nuestro caso concreto de uso, las macrotablas de la estadística oficial, sería muy conveniente, por ejemplo, que al tener una hipertabla (Provincias) clasificada por un eje de provincias y por varias otras dimensiones, y otra hipertabla (Galicia) con las mismas dimensiones pero con el espacio dando solamente el total de la C.A., pudiésemos hacer cálculos como:

[3] http://scipy.github.io/old-wiki/pages/EricsBroadcastingDoc

```
PctProv = 100*Provincias/Galicia
```

Podemos así escribir formulas sencillas, sin necesidad de obscurecerlas con bucles anidados recorriendo las dimensiones sobre las que estamos haciendo el cálculo.

Drop

R es un lenguaje pensado para la realización de scripts por un técnico estadístico en su trabajo de análisis. De ahí que se da prioridad a dar facilidades y a la comodidad de manejo frente al rigor que pueden tener otros lenguajes más orientados al desarrollo de aplicaciones. Esto, unido al carácter "secundario" de los arrays en R que comentábamos, se pone de manifiesto en el comportamiento por defecto de perder dimensiones degeneradas (las que tienen un solo miembro) cuando hacemos selecciones.

En varios objetos básicos de R se tiene un conjunto de funciones especiales de selección que admiten una sintaxis especial a base de corchetes (o del signo $ en el caso de listas y dataframes). Cuando hacemos x[3] estamos invocando a la función [con los argumentos x y 3. Lo mismo ocurre con el doble corchete, el $ para seleccionar miembros de una lista e incluso los selectores de asignación que empleamos al hacer por ejemplo x[a>0] <- TRUE son llamadas a la función: ´[<-´

En un array, el corchete selecciona los elementos que se piden en cada uno de sus argumentos separados por comas. Así, una expresión como por ejemplo:

Activos["Lugo", ,2:10 ,a>0]

elige un miembro del primer eje por nombre, todos los del segundo, los miembros 2 a 10 del tercero, y del cuarto según se cumpla una condición.

Este operador "[" admite un parámetro adicional *drop* cuyo valor por defecto es TRUE. En el ejemplo anterior, si no queremos perder el primer eje (y quién sabe si también el cuarto), deberemos poner:

Activos["Galicia", ,2:10 ,a>0, drop=FALSE]

Esto es engorroso, susceptible de errores por olvido y dificulta la legibilidad de las formulas.

La clase hip

Para disponer de nuestras hipertablas en R, hemos definido una clase S3 (llamada *hip*) capaz de representarlas por completo. Los datafames de dimensiones tienen también la posibilidad de incorporar los textos de las cabeceras y no solo sus códigos, de modo que el resultado de un cruce puede ser una hipertabla completa. Y hemos establecido que ese sea el resultado de la función *h.cruzar*().

También hemos acompañado a la clase *hip* de funciones para interactuar con el banco de hipertablas, no solo en escritura sino en lectura. De este modo, R no solo puede nutrir el banco, sino

que podemos explorar las posibilidades de manipulación y tratamiento de las hipertablas mientras están aún en R o incluso de aquellas producidas a partir de los cubos OLAP.

Las operaciones que pretendíamos hacer con ellas incluyen tanto los tratamientos de cálculo (tasas, porcentajes, etc.) como las transformaciones en su estructura más orientadas a la presentación (hipertablas compuestas, filtrado de elementos, edición de los niveles de profundidad jerárquica, etc.).

En la clase hip hemos representado la hipertabla como una lista de R con:

- Un miembro para cada una de las columnas de metadatos asociados a la hipertabla como tal en el banco de datos (fuente, título, etc)
- Un dataframe para la tabla descriptiva de los ejes (sus nombres y algunos otros).
- Una lista de dataframes cada uno de los cuales describe los elementos de un eje (nombres, niveles jerárquicos, códigos, etc.)
- Un vector con los datos (en formato de caracteres, dado que pueden tener también símbolos no numéricos)

Este vector de datos debe entenderse como un array, que en R no es sino un vector homogéneo dotado de unas dimensiones que permiten interpretarlo como multidimensional. En nuestro caso estas vienen dadas por los números de filas que tienen los dataframes de elementos de cada eje (el producto de los cuales debe, obviamente, coincidir con la longitud del vector de datos).

Operaciones en la clase hip

Para solventar los problemas que nos presentaban los arrays, y para lograr poder escribir fórmulas como la anteriormente vista, se han definido las hipertablas como una clase S3 de R y no una mera lista.

Las clases S3 de R, a diferencia de lo que ocurre en los lenguajes orientados a objetos habituales, no tienen métodos asociados. La utilidad de crear el objeto hip como una clase es que permite alterar el funcionamiento de las funciones genéricas cuando uno de sus argumentos es miembro de la clase. De este modo, las funciones genéricas de R tendrán en cuenta los métodos que escribamos para ellas a la hora de tratar con hipertablas.

En concreto, nosotros hemos creado la clase principalmente para cambiar el comportamiento de los operadores aritmético-lógicos, de modo que reciclen cuando alguno de los argumentos sea un miembro de la clase hip.

Esto se hace redefiniendo el grupo genérico de funciones: *Ops()*. En el método que hemos creado para nuestra clase, se interpreta el vector como un array, se pasa a numérico su contenido y hemos hecho además que recicle dimensiones degeneradas (broadcasting). Podría haberse hecho el reciclaje general, para cualquier tamaño, pero no lo hemos visto necesario. Con este cambio es

posible operar cuando una de las hipertablas tiene una dimensión degenerada, en el sentido de que se entiende que el valor único que tiene en ella es válido para enfrentarlo a todos los miembros que tenga el otro operando.

Las hipertablas se operan también con valores atómicos y con arrays que tengan una forma compatible.

Las funciones de transformación asociadas a la clase hip

Como decíamos, hemos acompañado a las hipertablas de un conjunto adicional de funciones para trabajar con ellas, formando una pequeña librería (*libhip*)

Las funciones se pueden clasificar en cinco grupos:

funciones de interacción con el banco de datos

Lectura y escritura completas, incluyendo campos del banco que no tienen ninguna utilidad en R como puede ser por ejemplo el cubo del que se nutren cuando son hechas en SQLServer OLAP services. La idea es poder utilizar R con cualquier hipertabla sin ninguna pérdida de información.

función de cruce

Una única función que produce *hip*s a partir del cruce de variables en un dataframe: La función h.cruzar() ya mencionada

funciones de transformación estructural

Incluyen las reestructuraciones más obvias que se pueden querer hacer sobre las hipertablas, ya que no siempre las tablas publicadas son un simple cruce homogéneo. Hay funciones para:

- filtrar la hipertabla por los miembros de un eje

- pegar a lo largo de un eje dos hipertablas que sean idénticas en todos los demás

- añadir o quitar explícitamente ejes degenerados

- permutar los ejes entre sí

- combinar dos tablas similares pero a distintos niveles de agregación respecto a una de sus dimensiones, dando lugar a una hipertabla con el eje jerarquizado

- realizar agregaciones a lo largo de un eje de la hipertabla llevándolo a un nivel superior

- Realización simplificada de cálculos habituales como los porcentajes

Corchetes

Hemos dotado a la clase hip de métodos para implementar los corchetes simples, dobles y de asignación (función [<-) para que operen con ellas como se espera en analogía con los arrays, pero cuidándonos de no perder dimensiones degeneradas, no solo porque en nuestro caso tienen un significado estadístico, sino porque además llevan metadatos asociados que debemos conservar.

Las funciones de corchete simple producen o editan una hipertabla. El corchete doble sirve para la extracción de datos como un array.

Hemos aprovechado la escritura de estos métodos para introducir una notación alternativa que dé a las formulas un aspecto más legible si se desea, pudiendo optar por referirnos a las dimensiones por su nombre y no necesariamente por posición, evitando así las engorrosas y obscuras listas de comas vacías que ponemos en los arrays.

Por ejemplo, si tenemos una tercera dimensión “mes”, podemos pedir:

Activos[mes=2:10]

como equivalente más claro de:

Activos[, ,2:10,]

Conversión con otros formatos

Resulta vital poder convertir en ambas direcciones respecto a arrays de R base, dataframes y en el futuro otros formatos (jsonstat, pcaxis, etc)

Otras funciones

Específicas de las hipertablas de nuestro banco de datos y sin mayor interés para otros.

Generalización a una clase de arrays extensibles

Una parte de la funcionalidad hecha para el manejo de nuestras hipertablas se puede generalizar, de modo que resulten funciones útiles para el manejo de arrays en R, dando un entorno alternativo para programar cálculos. En situaciones donde existe información de mesodatos o macrodatos numérica y de carácter multidimensional, con unas dimensiones prefijadas, los arrays de R con estas mejoras adicionales que hemos visto pueden ofrecer una alternativa al uso de dataframes en el propio R o del recurso a otros entornos como las hojas de cálculo.

Ejemplos obvios de esto son todos los cálculos de índices, tasas, porcentajes, etc.

Otro ejemplo de este tipo de tratamientos lo hemos encontrado en las tablas de mortalidad. El IGE realiza tablas de mortalidad por edades simples para Galicia y provincias con carácter anual. Los

cálculos involucran una serie de variables: Población, defunciones y nacimientos, con una misma estructura multidimensional, con ejes para: espacio geográfico, sexo, año y edades o grupos de edad. También hay otras variables involucradas en los cálculos que tiene solo algunas de estas dimensiones: Tasa de mortalidad infantil (espacio, sexo, año), índice de masculinidad al nacimiento (espacio, año), factores de separación (edad, sexo), etc. Los resultados obtenidos tienen también esa misma dimensionalidad, por lo que en este ejemplo estamos ante un algoritmo especialmente adecuado para implementarlo con arrays de esas 4 dimensiones. Y más aún si los arrays reciclan las dimensiones faltantes y si no las pierden accidentalmente por defecto.

rarray

Definimos por ello una nueva clase S3 en R, *rarray,* de arrays "reciclables". En ella tenemos:

- Una definición de los operadores con reciclaje en ejes degenerados (*broadcast*)

- Funciones auxiliares de manipulación: Similares a las hechas para la clase hip. Algunas, como por ejemplo aperm() para permutar los ejes, a pesar de estar ya disponibles en R es necesario adaptarlas para garantizar que conserven la clase rarray a medida que transformamos los miembros de la clase.

- Una nueva definición de la función de selección, para arreglar también la pérdida (drop) de dimensiones, como se ha explicado. En nuestra clase hemos alterado el valor por defecto del parámetro drop y la pérdida de dimensiones solo se producirá si se demanda explícitamente.

También se ha incorporado, de cara a mejorar la claridad de las formulas, la posibilidad de referenciar los ejes por nombre como hemos visto con las hipertablas. En los arrays de R puede haber opcionalmente un atributo *dimnames*, con los nombres de los elementos de los ejes. Este, a su vez, puede tener un atributo *names*, nombrando así los propios ejes o dimensiones del array. En caso de estar ambos presentes, nuestro corchete admite librarse del recuento de las comas para saber sobre que dimensión se está actuando, y poner en su lugar parámetros no por posición sino por nombre.

Conclusión

El balance final que podemos hacer de nuestra incursión con R en las fases de agregación y manipulación de macrotablas es positivo. Se han encontrado obstáculos, pero han podido ser solucionados, y con independencia de que las decisiones adoptadas puedan ser discutibles, se ha puesto en evidencia que el lenguaje nos permite adaptarlo con bastante facilidad a lo que se intenta hacer, oponiendo muy pocas barreras o resistencia para que podamos conformar una infraestructura, en este caso de agregación y de cálculos, adecuada a nuestras necesidades.