Propuesta de protocolo de geocodificación

Versión 1.1

Grupo de trabajo de geocodificación. Proyecto MEDEA3

03 agosto, 2017

La versión actual de este protocolo detalla una propuesta de geocodificación de las defunciones para aquellos grupos que deban abordar esta tarea con sus propios medios. Esta propuesta no pretende ser un protocolo de obligado seguimiento para los grupos de investigación, somos conscientes de que varios grupos disponen de procedimientos de geocodificación más refinados vía la colaboración con otras entidades, particularmente institutos estadísticos. En esos casos animamos a que utilicen sus propios medios. Para aquellos que no dispongamos de dicha colaboración el presente protocolo pretende establecer unas líneas homogéneas de geocodificación de forma que los datos de todos los grupos sean comparables y con unos criterios de calidad mínimos, en base a la experiencia que hemos ido acumulando en este sentido.

Las herramientas de geocodificación utilizadas a lo largo del documento son libres y por tanto pueden ser usadas por todos los grupos. Además, estas herramientas pueden ser usadas tanto a través de la web como desde el software estadístico *R*, que automatiza el proceso de llamada a los servicios web que llevan a cabo la geocodificación. Indicaremos las páginas web a las que se debe acceder para proceder a la geocodificación y también se incluirá código de *R* que puede ser usado para tal fin sin necesidad de acceder manualmente a ninguna página web. Recomendamos, y a partir de ahora describimos, el uso de *R* para llevar a cabo estas tares de geocodificación.

Antes de comenzar con la descripción de las herramientas y procesos nos parece importante resaltar que el éxito de la geocodificación depende en gran medida de la calidad de las direcciones disponibles. En este sentido, es importante valorar la posibilidad de mejorar las mismas a partir de fuentes disponibles según la comunidad autónoma (cruce con padrón a partir del DNI, cruce con algún sistema poblacional sanitario a partir del código de identificación sanitario, entre otras). Si no fuera posible realizar dicho cruce el procedimiento descrito seguiría siendo igualmente válido y, en la experiencia de los grupos de la Comunidad Valenciana, daría porcentajes de codificación más que aceptables.

# Herramienta *CartoCiudad*

*CartoCiudad* es un proyecto que depende del *Ministerio de Fomento* y que permite geocodificar información a partir de direcciones postales para todo el ámbito nacional. Podemos encontrar más información en la página web del proyecto accediendo [a este link](http://www.cartociudad.es/portal/).

Los datos disponibles en este proyecto para llevar a cabo la geocodificación son actualizados cada año a nivel municipal en función de su antigüedad o demanda. En [esta página web](http://www.cartociudad.es/portal/web/guest/cobertura-y-actualizacion) se puede consultar la fecha de actualización en cada municipio de cada comunidad autónoma. Se puede observar que la periodicidad de actualización de la información varía bastante de unas comunidades a otras. En concreto, la Comunidad Valenciana, tiene unos datos particularmente actualizados. Esperemos (aunque tampoco lo hemos constatado) que este factor no haya podido influir en la calidad de la geocodificación de los datos de la Comunidad Valenciana y que los datos que manejamos no sean particularmente optimistas.

Accediendo a la sección de cálculos, desde la web de *CartoCiudad* o directamente desde [este link](http://www.cartociudad.es/portal/web/guest/calculos) podemos geocodificar una dirección o un fichero de direcciones, con un límite máximo de 100 direcciones por cada llamada. Debido a este límite en el número de direcciones por fichero resulta más práctico realizar la geocodificación a través de un programa externo que se pueda conectar a este servicio repetidamente e ir enviando las direcciones a la web de forma automática. El programa estadístico *R* permite esta conexión y procedimiento mediante la librería *caRtociudad*. No hay límite diario de solicitudes de geocodificación, el único límite es el tamaño del fichero (nuevamente 100 direcciones), pero es posible programar desde *R* un proceso que envíe las solicitudes de geocodificación sin límite por día.

## *CartoCiudad* desde R: librería *caRtociudad*

La librería *caRtociudad* puede instalarse desde la web de [*GitHub*](https://github.com/carlosvergara/caRtociudad) mediante el siguiente código de *R*:

if(!require(devtools)) install.packages("devtools")  
library(devtools)  
library(httr)  
install\_github("carlosvergara/caRtociudad")

La función *install\_github* puede dar problemas en algunas instituciones por la configuración del proxy. Si así fuera, hemos de configurar *R* para que solvente dichos problemas. En caso contrario los próximos pasos del protocolo pueden ser problemáticos dado que intentarán acceder vía web a *CartoCiudad* y, seguramente, no podrían hacerlo. Recuerda, el siguiente paso será sólo necesario si tienes problemas de conexión web por la configuración de tu red, en un domicilio particular (por ejemplo), sin restricciones de red seguramente no sería necesario. La configuración de *R* se realiza ejecutando la siguiente sentencia:

httr::set\_config(  
 httr::use\_proxy(  
 url = "XXX.XXX.XXX.XX",  
 port = XXXX,  
 username = "XXXX",  
 password = "XXXX"  
))

donde *url* es la dirección IP o dirección web del proxy de vuestra red, *port* es el puerto que vuestra institución utilice para comunicaciones y *username* y *password* las credenciales de acceso, si fueran necesarias. Si no necesitáis identificar usuario y contraseña en vuestra red quitar dichos argumentos de la sentencia anterior. Los valores de nuestra red los hemos enmascarado por seguridad. Las comillas, allá donde aparecen son necesarias. Vuestros servicios de informática (suponemos que) podrán proporcionaros información con más detalle sobre estos valores para vuestras redes informáticas corporativas.

Una vez instalada y cargada la librería (y configurado en *R* el acceso al proxy si fuera necesario) es muy sencillo geocodificar cualquier dirección. A continuación se muestra un ejemplo simple de uso:

library(caRtociudad)  
cartociudad\_geocode("Plaza del Ayuntamiento 1, Valencia, Valencia, 46002")

La salida que proporciona la función es la siguiente:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | province | muni | type | address | postalCode | tip\_via | lat | lng | portalNumber | stateMsg | state |
| 462500003207 | Valencia/València | Valencia | portal | AYUNTAMIENTO | 46002 | PLAZA | 39.46979 | -0.376963 | 1 | Resultado exacto de la búsqueda | 1 |

El resultado contiene información que identifica el lugar o lugares geocodificados, en caso de solicitar la geocodificación para más de una dirección. En el campo **state** contiene información sobre la confianza/precisión de las coordenadas proporcionadas. Esta variable **state** puede tomar como valor:

1. Resultado exacto.
2. Portal par no encontrado, se devuelve el par más cercano. En el campo portalNumber nos reporta qué portal nos ha dado.
3. Portal impar no encontrado, se devuelve el impar más cercano. En el campo portalNumber nos reporta qué portal nos ha dado.
4. Punto kilométrico no encontrado, se devuelve el más próximo.
5. No se encuentran portales o puntos kilométricos asociados al vial.
6. El tipo de vía encontrado no coincide con el solicitado en la consulta.
7. No encuentra resultados, pero devuelve la entidad superior (normalmente será el municipio).

En los campos **lng** y **lat** disponemos de la longitud y la latitud devuelta por *CartoCiudad*. Esta información (longitud, latitud y el resto de campos mencionados) puede ser almacenada en un banco de datos, junto con el resto de información disponible en la base de datos de mortalidad.

## Normalización direcciones para *CartoCiudad*.

En el [documento de referencia de CartoCiudad](http://www.idee.es/resources/presentaciones/JIIDE10/ID449_Servicio_web_de_Geocodificacion_para_Cartociudad.pdf) se comentan algunos detalles interesantes sobre el servicio web de geocodificación con *CartoCiudad*. Entre otros detalles se describe qué formato se recomienda para las direcciones que queramos geocodificar, aunque en todo caso deberá ser una única cadena de texto que pasaremos a la función **geocode**.

El motor de búsqueda de *CartoCiudad* realiza una normalización de las direcciones antes de proceder a la geocodificación. Este proceso realiza transformaciones como eliminar tildes y caracteres raros, pasar todas las letras a minúsculas, eliminar palabras que no aportan información como artículos o pronombres y estandarizar los tipos de vía. Para estandarizar los tipos de vía dispone de unas tablas de equivalencias en las que, por ejemplo, tanto da *AV* como *AVDA* para representar el tipo de vía *AVENIDA*.

A continuación detallamos cómo el buscador de *CartoCiudad* descompone el texto libre para obtener las diferentes partes de la dirección. Conocer este procedimiento nos aporta información útil para organizar de la mejor manera posible la dirección a geocodificar.

En resumen el procedimiento de extracción de información de *CartoCiudad* a partir de cadenas de texto libre sería el siguiente. Este procedimiento se ejecutará secuencialmente siguiendo exactamente el **orden** en el que se describe a continuación:

1. Extracción del **código postal**. Se determina como cualquier número de **5** cifras, en cualquier posición de la cadena. Es decir, si *CartoCiudad* encuentra un número de 5 cifras considera que es el código postal, lo extrae y lo elimina de la cadena. Por este motivo **debemos normalizar el código postal a 5 dígitos (con un cero delante si hiciera falta) y colocarlo en cualquier lugar de la cadena**. Es importante remarcar, que *CartoCiudad* espera el **código postal**, y no el **código INE** del municipio, que también es un código con 5 números. Introducir el **código INE** en la cadena puede perjudicar el comportamiento del algoritmo de geocodificación de *CartoCiudad*.
2. Extracción del **tipo de vía**. *CartoCiudad* busca este valor al **principio** de la cadena de texto si es que lo encuentra en su tabla de equivalencias, con posibles variantes de cada uno de sus valores. Por este motivo es importante que, en la medida de lo posible, esté previamente normalizado a una de las equivalencias más habituales o al nombre directamente para este tipo de vía (*CALLE, AVENIDA o AVDA,...*). Si *CartoCiudad* encuentra alguna de los valores esperables para el tipo de vía lo extrae y lo elimina de la cadena de texto.
3. Extracción del **nombre de la provincia**. Cartociudad espera este valor al **final** de la cadena de texto (búsqueda exacta o aproximada si falla la exacta). Si se encuentra el nombre de provincia *CartoCiudad* lo extrae y elimina de la cadena de texto.
4. Extracción del **nombre del municipio**. *CartoCiudad* espera este valor al **final** de la cadena de texto resultante de los pasos anteriores. Si se hubiera encontrado y extraído la provincia se acotaría la búsqueda (exacta o aproximada de los municipios) a aquellos municipios de la citada provincia. Si no se encuentra el municipio pero sí se tiene la provincia, *CartoCiudad* considera que la provincia es también el municipio. Si se encuentra el municipio se extrae y elimina de la cadena de texto.
5. Extracción del **número de portal o pk**. Se busca este valor al **final** de la cadena de texto. Si se encuentra un valor numérico en dicha posición se extrae y elimina de la cadena de texto.
6. Extracción del **nombre de la vía**. Será la cadena resultante de la sustracciones realizadas en todos los pasos anteriores.

Con este algoritmo *CartoCiudad* extrae las componentes relevantes de la cadena de texto y utiliza dichos valores para geocodificar la dirección correspondiente. Así, en resumen, en base al procedimiento que acabamos de describir recomendamos estructurar la direcciones a geocodificar como cadenas de texto estructuradas en el siguiente orden:

**(TIPO DE VÍA) (NOMBRE DE LA VÍA) (NÚMERO DE PORTAL), (NOMBRE DE MUNICIPIO), (NOMBRE DE PROVINCIA) , (CÓDIGO POSTAL)**

Por ejemplo:

**AVENIDA CORTES VALENCIANAS 18, VALENCIA, VALENCIA, 46015**

En caso de no disponer de alguno de estos datos se puede dejar en blanco ya que, si éstos se introducen en el orden sugerido, *CartoCiudad* los interpretará de forma correcta, aparezcan o no de forma explícita en las cadenas de texto correspondientes.

# Herramienta de geocodificación de *Google*

El portal *Google* es ampliamente conocido y también dispone de herramientas específicas para geocodificar direcciones. *Google* pone a disposición de sus usuarios un servicio web que permite conectarse a través de internet para geocodificar direcciones. El problema de esta herramienta es el límite establecido por *Google* de 2500 geocodificaciones diarias en su versión gratuita.

En la experiencia del grupo de la Comunidad Valenciana *caRtociudad* parece reportar resultados algo más precisos que los de *Google*, aunque creemos que ambas herramientas son útiles y sus resultados bastante acertados. Además, dado al límite diario en el número de direcciones que *Google* permite geocodificar, creemos conveniente considerar esta última herramienta como segundo recurso, usándola sólo para aquellos registros que no han podido ser geocodificados en caRtociudad. En cualquier caso consideramos la herramienta de *Google* como una utilidad paralela a *caRtociudad* ya que el servicio de *Google* reconoce, más allá de direcciones, puntos de interés como "Residencia XXX". Esta opción resulta de gran utilidad para ciertas defunciones cuya dirección ha sido mal cumplimentada en el certificado de defunción.

El programa estadístico *R*, concretamente la librería *ggmap*, permite conectar con el servicio web de *Google* y así automatizar el procedimiento de geocodificación de direcciones con esta herramienta. La función **geocode** de *ggmap* permite geocodificar una o varias direcciones en bloque.

## *Google* Maps desde R: librería *ggmap*

La librería *ggmap* puede instalarse desde el repositorio del programa *R* (*CRAN*) mediante el siguiente código en *R*:

install.packages("ggmap")

Una vez instalada y cargada la librería es muy sencillo geocodificar cualquier dirección. A continuación se muestra un ejemplo simple de uso:

library(ggmap)  
geocode("Plaza del Ayuntamiento 1, Valencia, Valencia, 46002", output = "all", source = "google")

La salida que proporciona esta función contendrá más o menos información en función del valor del parámetro output. Este parámetro puede tomar los valores *latlon*, *latlona*, *more* o *all*. La última de estas opciones devuelve toda la información que *Google* dispone sobre la dirección encontrada en su sistema, permitiéndonos seleccionar aquellas componentes que nos interesen. La salida estará formada por dos objetos, *results* y *status*. En *results* aparecerá la información referente a la geocodificación realizada, entre la que se encuentran la latitud y la longitud (las dos variables de interés principal), o estará vacío si la dirección no ha sido identificada. En *status* aparecerá la información referente al éxito o no de la geocodificación, pudiendo devolver los siguientes valores:

* **OK**: indica que la dirección se pudo identificar correctamente y que se devuelve al menos una geocodificación que corresponde con la dirección de interés.
* **ZERO\_RESULTS**: indica que la dirección se interpretó correctamente pero que no se devuelven resultados, por ejemplo porque para el geocodificador la dirección no existe.
* **OVER\_QUERY\_LIMIT**: indica que se ha excedido el límite de geocodificaciones diarias permitidas o el número de geocodificaciones por segundo permitidas por el servicio.
* **REQUEST\_DENIED**: indica que se rechazó tu solicitud.
* **INVALID\_REQUEST**: suele indicar que la geolocalización no está completa.
* **UNKNOWN\_ERROR**: indica que no se pudo procesar la solicitud por un error en el servidor.

Tanto los resultados que devuelven *OVER\_QUERY\_LIMIT* como los *UNKNOWN\_ERROR* pueden ser geocodificados de forma exitosa en otro momento si se repite la geocodificación. Ambos indican que, por el motivo que sea (exceso sobre el límite o indisposición temporal del servicio) no se ha podido efectuar la geocodificación **en dicho momento**. Los códigos de status más habituales, de largo, son los tres primeros aunque en algún momento el servicio, o la librería *ggmap*, pueden devolver también como resultado un valor perdido *NA* para alguna de las direcciones. Estas direcciones también son susceptibles de ser regeocodificadas en un intento posterior.

En [esta página web](https://developers.google.com/maps/documentation/geocoding/intro#geocoding) se puede ampliar la información aquí resumida que hemos considerado de particular interés para el proyecto.

# Propuesta de procedimiento de Geocodificación con *R* para el proyecto MEDEA.

Una vez introducidas las herramientas de geocodificación que consideramos adecuadas para la geocodificación de la información del proyecto MEDEA presentamos un banco de datos de ejemplo de la Comunidad Valenciana en el que vamos a mostrar cómo se podría geocodificar las direcciones allí contenidas. Puedes acceder al archivo .Rdata que contiene este banco de datos en el subdirectorio "/datos/Ejemplo.RData" de este proyecto. Nuestro banco de datos de ejemplo (con 30 registros) tiene la estructura que se muestra a continuación:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | TVIA | NVIA | NUMEROPOLI | MUNICIPIO | PROVINCIA | CP |
| 1 | CALLE | MANUEL CANDELA | 41 | Valencia | Valencia | 46021 |
| 2 | CALLE | SANTOS JUSTO Y PASTOR | 151 | Valencia | Valencia | 46022 |
| 3 | CALLE | CHIVA | 44 | Valencia | Valencia | 46018 |
| 4 | CALLE | MAESTRA TERESITA NUÑEZ |  | Valencia | Valencia |  |
| 5 | CALLE | AUSIAS MARCH | 40 | Valencia | Valencia |  |
| 6 | PLAZA | VICENTE IBORRA | 8 | Valencia | Valencia |  |
| 7 | CALLE | SANTO DOMINGO SABIO | 6 | Valencia | Valencia |  |
| 8 | CALLE | MARIANO CAVIA | 35 | Valencia | Valencia |  |
| 9 | CALLE | LEONOR JOVENI | 6 | Valencia | Valencia |  |
| 10 | PLAZA | ALFONSO EL MAGNANIMO |  | Valencia | Valencia |  |
| 26 | CALLE | RESIDENCIA (TERCERA EDAD) |  | Forcall | Castellón | 12310 |
| 27 |  | MASIA MUELA |  | Todolella | Castellón |  |
| 28 | CALLE | RESIDENCIA DE LA TERCERA EDAD |  | Forcall | Castellón | 12310 |
| 29 | CALLE | DOCTORA DAMIANA ZARAGOZA | 3 | Villajoyosa | Alicante | 03570 |
| 30 | CALLE | LA NUCIA | 8 | Elche/Elx | Alicante |  |

Tal y como ya hemos anticipado en algún momento optaremos por *CartoCiudad* como primera opción para la geocodificación y el servicio de *Google*, como segunda opción, sólo para aquellas direcciones que no sean geocodificadas de forma satisfactoria por *CartoCiudad*. A continuación describimos ambos procesos para nuestros datos de ejemplo precedidos de una fase de preproceso de los datos.

## Preproceso de los datos

Asumiremos que los datos, tal y como aparecen en la tabla anterior, se encuentran en un *data.frame* de *R*. En nuestro caso dicho *data.frame* se llama *ejemplo2*. En caso de que la información que quisiéramos geocodificar estuviera en una base de datos externa o en algún otro formato externo a *R* ésta debería ser previamente importada y convertida a formato *data.frame* si es que no lo estuviera.

Como proceso previo de los datos, conviene revisar los tipos de vía de todas las direcciones y proceder a la estandarización de su información, pues es posible que algunos de nuestros tipos de vía estén mal escritos y no sean encontrados en la tabla de equivalencias. Así, por ejemplo podemos encontrar para algún registro *CALEL* en lugar de *CALLE*, lo que habría de ser corregido. Según hemos podido comprobar en nuestro caso, este tipo de preproceso mejora los resultados. A tal efecto, hemos creado una función llamada limpiaTVIA en *R*, contenida en el anexo del protocolo, en la que se realiza este proceso sobre el campo **TVIA** según la casuística encontrada en los datos de la C.Valenciana y que podría servir de utilidad para otros grupos. Aconsejamos el uso de dicha función de forma previa a la geocodificación de cualquier banco de datos. Las funciones *normalizaNVIA* y *normalizaCODPOSTAL* realizan operaciones similares para el nombre de vía y código postal respectivamente. Así, por ejemplo, *normalizaNVIA* substituye FDEZ por FERNANDEZ o STA por SANTA en las direcciones lo que mejora su identificación. De la misma forma *normalizaCODPOSTAL* transforma códigos postales del tipo "3400" a "03400", de 5 cifras.

La casuística de las funciones anteriores se ha definido según la experiencia del grupo de Valencia. En cualquier caso dichas funciones pueden ser completadas con la experiencia de otros grupos si es que en vuestros datos encontráis ciertas situaciones, de forma sistemática, susceptibles de ser corregidas. Podéis consultar con detalle las situaciones contempladas en la labor de preproceso a día de hoy en las funciones tal y como aparecen en el anexo al final de este documento. No dudéis en hacer llegar al grupo de Valencia cualquier otra situación que consideréis que ha de ser contemplada en el preprocesamiento de los datos.

A continuación mostramos como se podrían aplicar las funciones anteriores al preproceso de los datos de nuestro ejemplo y como montamos un campo de texto libre que contiene una cadena por cada dirección a geocodificar. El formato de esta cadena es el sugerido en las secciones anteriores de este protocolo.

ejemplo2$TVIA <- sapply(ejemplo2$TVIA, limpiaTVIA)  
ejemplo2$NVIA <- sapply(ejemplo2$NVIA, normalizaNVIA)  
ejemplo2$CP <- sapply(ejemplo2$CP, normalizaCODPOSTAL)  
ejemplo2$DIREC <- with(ejemplo2, paste0(TVIA, " ", NVIA, " ", NUMEROPOLI, ", ",  
 MUNICIPIO, ", ", PROVINCIA, ", ", CP))

## *caRtociudad*

A continuación, las siguientes sentencias mandan a geocodificar cada una de las direcciones de *ejemplo2* y guardan la información original de cada dirección en el data.frame *resultados\_cartociudad*

library(caRtociudad)  
resultados\_cartociudad <- cartociudad\_geocode(ejemplo2$DIREC)  
colnames(resultados\_cartociudad) <- paste0("cartociudad\_", names(resultados\_cartociudad))  
ejemplo2 <- data.frame(ejemplo2, resultados\_cartociudad)

Notar que la función se ejecuta en bucle para cada una de las direcciones disponibles.

Consultando el objeto resultados\_cartociudad$state podemos valorar el éxito de la geocodificación llevada a cabo. En nuestro ejemplo, con 30 registros, 15 de ellos han obtenido *state* igual 1 (resultado exacto), en cuatro se obtuvo *state* de 2 o 3 (portal más próximo), dos obtuvieron state igual a 5 (vial sin portales o pk asociados), y nueve tienen state igual a 10. Daremos por geocodificados los 21 primeros mientras que para los 9 restantes se intentará la geocodificación con la herramienta de *Google*.

## *Google*

De la misma forma que para *CartoCiudad*, en primer lugar mostramos unas sentencias de código que generan una estructura de datos adecuada para contener la información devuelta por la herramienta de *Google*.

library(ggmap)

## Loading required package: ggplot2

# Creamos la estructura para recoger la información que proporciona CartoCiudad  
resultados\_google <- data.frame(matrix(nrow = nrow(ejemplo2), ncol = 5))  
colnames(resultados\_google) <- c("google\_STATUS", "google\_latitude", "google\_longitude",   
 "google\_type", "google\_formatted\_address")  
  
# Anexionamos esta estructura a nuestro banco de datos original  
ejemplo2 <- data.frame(ejemplo2, resultados\_google)

El campo type contiene información acerca de la localización de las coordenadas en relación a la dirección, si corresponde a su portal, al centro del inmueble ... `formated\_address' contiene una cadena con la dirección que se ha geocodificado tal y como la tiene guardada internamente *Google*. Guardamos ambos valores porque consideramos que podrían ser de interés en algún momento. El proceso de geocodificación y almacenaje de la información con la herramienta de *Google* podría llevarse a cabo tal y como se describe en las siguientes sentencias.

# Selección de las geocodificaciones defectuosas en CartoCiudad.  
selec <- which(!ejemplo2$cartociudad\_state %in% paste0(1:6))  
for (i in selec) {  
 direc <- ejemplo2$DIREC[i]  
 # La opción override\_limit parece que permite realizar algunas geocodificaciones extra  
 # por encima del límite diario, a costa de que pueda haber alguna penalización  
 # extra en el límite diario de los días posteriores.  
 aux <- geocode(as.character(direc), output = "all", source = "google", override\_limit = TRUE)  
 if (!any(is.na(aux))) {  
 ejemplo2$google\_STATUS[i] <- aux$status  
 if (length(aux$results) > 0) {  
 ejemplo2$google\_latitude[i] <- aux$results[[1]]$geometry$location$lat  
 ejemplo2$google\_longitude[i] <- aux$results[[1]]$geometry$location$lng  
 if (length(aux$results[[1]]$types) > 0) {  
 ejemplo2$google\_type[i] <- aux$results[[1]]$types[[1]]  
 }  
 ejemplo2$google\_formatted\_address[i] <- aux$results[[1]]$formatted\_address  
 }  
 }  
}

Notar que en las sentencias anteriores el bucle nuevamente geocodifica dirección a dirección aunque se podría hacer por tandas de grupos de direcciones. En cualquier caso, se ha considerado oportuno hacerlo dirección a dirección. También resulta interesante comentar que aunque *Google* devuelve un montón de información sobre la dirección que se ha geocodificado sólo nos guardamos aquella que nos interesa por cuestiones operativas.

Consultando el objeto ejemplo2$google.STATUS podemos valorar el éxito de la geocodificación. Para aquellos registros cuyo valor en este campo sea *OVER\_QUERY\_LIMIT*, *UNKNOWN\_ERROR* o un simple NA es posible obtener una geocodificación exitosa intentándolo de nuevo. Ese proceso se podría llevar a cabo con un conjunto de sentencias de *R* de este tipo:

selec <- which(ejemplo2$google\_STATUS %in% c("OVER\_QUERY\_LIMIT", "UNKNOWN\_ERROR"))  
 # A continuación se repetiría el bucle anterior

Según el procedimiento expuesto, los valores de latitud y longitud de los registros geocodificados se encuentran en las variables carto.latitude y carto.longitude para los registros cuyo valor en cartociudad\_state es 1, 2, 3 y 5, y en los campos google.latitude y google.longitude para los campos en los que cartociudad\_state es 10 y google.status es OK. El resto de registros serían los que no se han podido geocodificar y que deben desestimarse o pasar a un proceso de revisión manual en función de la proporción del total que supongan estos registros y de los recursos del grupo que lleva a cabo el proceso.

## Coordenadas

Por último, tras los procesos de geocodificación llevados a cabo en *CartoCiudad* y *Google*, resulta conveniente crear un campo final único para las coordenadas que contenga las coordenadas de todos los geocodificados independientemente del sistema que las haya generado. Podríamos hacer esta labor, por ejemplo, con este código de *R*:

# Los que no hayan podido ser geocodificados se quedarán con coordenadas (0, 0), ya   
# que los valores perdidos NA hacen que algunos procesos posteriores no funcionen bien.  
ejemplo2$latitude <- rep(0, dim(ejemplo2)[1])  
ejemplo2$longitude <- rep(0, dim(ejemplo2)[1])  
  
# Los geocodificados con caRtociudad  
seleccarto <- which(ejemplo2$cartociudad\_state %in% paste0(1:5))  
ejemplo2$latitude[seleccarto] <- ejemplo2$cartociudad\_lat[seleccarto]  
ejemplo2$longitude[seleccarto] <- ejemplo2$cartociudad\_lng[seleccarto]  
  
# Los geocodificados con \*Google\*  
selecgoogle <- which(!ejemplo2$cartociudad\_state %in% paste0(1:5) & ejemplo2$google\_STATUS == "OK")  
ejemplo2$latitude[selecgoogle] <- ejemplo2$google\_latitude[selecgoogle]  
ejemplo2$longitude[selecgoogle] <- ejemplo2$google\_longitude[selecgoogle]

# Aplicación a la geocodificación de la mortalidad en la Comunidad Valenciana

Para ilustrar el funcionamiento de este procedimiento se comentarán los resultados obtenidos de la aplicación de este procedimiento a los datos de mortalidad de las ciudades de Alicante, Valencia y Castellón de la Comunidad Valenciana del año 2015. Estos resultados se enmarcan dentro de una aplicación más general del algoritmo que nos ha permitido geocodificar toda la mortalidad de la Comunidad Valenciana, independientemente del tamaño de sus municipios, para el periodo 1996-2015. Los resultados obtenidos en este proceso, para las capitales de provincia de nuestra comunidad, fueron contrastados con los utilizados en su día en las fases anteriores de MEDEA con procedimientos completamente independientes (ajenos completamente a *CartoCiudad* y la herramienta de geocodificación de *Google*). Dicha comparación, según se informó en su día en la reunión que mantuvimos en el proyecto sobre geocodificación, arrojó una concordancia muy alta entre los resultados de ambas fuentes, alrededor de un 91%. Teniendo en cuenta que ambos procedimientos podrían tener sus propias fuentes de errores, esta validación otorga un alto grado de fiabilidad a ambos procedimientos.

Se registraron un total de 11800 fallecimientos en el año 2015 en las tres capitales de la Comunidad Valenciana. Aplicando el procedimiento de *CartoCiudad* se consiguieron geocodificar con state final 1, 2, 3 o 5 el 89% aproximadamente de las defunciones. Por tanto, sólo el 11 % del total de defunciones tuvieron que ser mandadas a geocodificar a *Google*. Estas defunciones pueden ser geocodificadas en *Google* en un solo día pese a las restricciones de uso de esta herramienta. Tras la geocodificación en *Google* de los registros que no han podido ser geocodificados en *CartoCiudad* se alcanzó un porcentaje de direcciones encontradas en torno a un 96-97 % para este año y conjunto de ciudades.

# Proyección de las defunciones geocodificadas sobre cartografía para obtener la sección censal correspondiente.

Aunque esta cuestión no pertenece estrictamente al proceso de geocodificación de direcciones, creemos conveniente acabar este protocolo describiendo cómo proyectar las coordenadas geográficas de cada una de las defunciones geocodificadas sobre cierta cartografía, en nuestro caso las secciones censales de las ciudades del proyecto. Obviamente, para ejecutar este proceso necesitaremos disponer de una cartografía por secciones censales de las ciudades de interés. Asumiremos que dicha cartografía esta en formato shapefile (.shp), el formato más común para este tipo de archivos. También asumiremos que el archivo shapefile de la cartografía viene acompañado de un archivo complementario con extensión .prj con el mismo nombre que el archivo que tiene extensión .shp. Este archivo .prj contiene la información necesaria para situar la cartografía dentro de la esfera terrestre de forma precisa y así poder superponer las coordenadas de las defunciones con la información de dicha cartografía. En el URL <http://www.ine.es/censos2011_datos/cartografia_censo2011_nacional.zip> se puede descargar un archivo shapefile con la división geográfica de España en secciones censales, según el censo de 2011, que podría ser útil a los grupos del proyecto.

Una vez se disponga de la cartografía de la ciudad con los requisitos que acabamos de comentar, la proyección de las defunciones sobre la cartografía se podría llevar a cabo según el siguiente código de R:

library(rgdal)  
library(sp)  
  
# Leemos cartografía (entre los archivos de la cartografía debe haber uno con extensión .prj).   
# Sustituir "ruta\_al\_fichero" y "nombrefichero" por los argumentos oportunos.  
carto <- readOGR("ruta\_al\_fichero/nombrefichero.shp", layer = "nombrefichero")  
  
# Definimos un objeto con las coordenadas de las defunciones y calculamos su proyección  
# según los parámetros utilizados por CartoCiudad y Google para asignarles las coordenadas.  
puntos <- data.frame(ejemplo2$longitude, ejemplo2$latitude)  
colnames(puntos) <- c("longitude", "latitude")  
coordinates(puntos) <- ~ longitude + latitude  
proj4string(puntos) <- CRS("+proj=longlat +ellps=GRS80 +datum=WGS84 +no\_defs")  
  
# Transformamos la proyección de los puntos al mismo sistema de coordenadas de la   
# cartografía para que las coordenadas de ambas fuentes de información coincidan.  
puntos\_trans <- spTransform(puntos, CRS(proj4string(carto)))  
  
# Por último, asignamos la sección censal que corresponde a cada punto con la función over.   
# En nuestra cartografía el campo DISTRSEC contiene el Identificador de cada sección censal   
# que vamos a asignar a las defunciones. Si en vuestras cartografías dicho identificador se   
# llamara de forma distinta habríais de sustituir DISTRSEC en la siguiente expresión  
# por dicho identificador.   
ejemplo2$SC <- over(puntos\_trans, carto)$DISTRSEC

De esta manera la variable SC, de ejemplo2, contendrá el identificador de la sección censal que corresponde a cada defunción. Aquellos puntos que no encajen en ningún polígono, o bien porque no han sido geocodificados y tienen como valor de las coordenadas (0, 0), o bien porque las coordenadas asignadas llevan el punto fuera de la cartografía por algún error en la dirección o en el proceso de geocodificación, tendrán vacío el valor del campo SC.

# Anexo 1: Funciones de limpieza y normalización de datos

## limpiaTVIA

limpiaTVIA<-function(cadtvia){  
   
 devuelve<-as.character(cadtvia)  
 if (is.na(cadtvia)==TRUE){  
 devuelve<-""  
 exit()  
 }  
 if (cadtvia %in% c("C.","+ALLE","CALLR","CALLL","CALLA","CALER","CALL","CALL,","CALL3","CALE","CAKKE","+ALL","CALER","CALLL","ALLED","ALLEM","C","CAALL","CALEC","CALEL","CALLA","CL","CARRE","CARR4","CCALL","C/","ALLEA")){  
 devuelve<-"CALLE"  
 }  
 if (cadtvia %in% c("ABDA","AVD","AVD.","AVDA-","AVNDA","AVGDA","AVGA","AVENI","AVDAE","AVDAA","AVDA,","AV","AVAD","AVDA.","AVDA","AVGDA")){  
 devuelve<-"AV"  
 }  
 if (cadtvia %in% c("CAMI","CAMIN",".MNO")){  
 devuelve<-"CMNO"  
 }  
 if (cadtvia %in% c("PLZA.","PZA.","PL","PLZA","PLA")){  
 devuelve<-"PLAZA"  
 }  
 if (cadtvia %in% c("PSJE","PJE.")){  
 devuelve<-"PSAJE"  
 }  
 if (cadtvia %in% c("BARR")){  
 devuelve<-"BARRO"  
 }  
 if (cadtvia %in% c("URB.","URBA","URBAN","URBN")){  
 devuelve<-"URB"  
 }  
 if (cadtvia %in% c("PTDA","PTIDA")){  
 devuelve<-"PARTIDA"  
 }   
 if (cadtvia %in% c(".")){  
 devuelve<-""  
 }  
 if (is.na(cadtvia)){  
 devuelve=""  
 }  
 devuelve  
}

## normalizaNVIA

normalizaNVIA<-function(devuelve){  
 devuelve<-gsub("1 ","UNO ",devuelve)  
 devuelve<-gsub("2 ","DOS ",devuelve)  
 devuelve<-gsub("3 ","TRES ",devuelve)  
 devuelve<-gsub("4 ","CUATRO ",devuelve)  
 devuelve<-gsub("5 ","CINCO ",devuelve)  
 devuelve<-gsub("6 ","SEIS ",devuelve)  
 devuelve<-gsub("7 ","SIETE ",devuelve)  
 devuelve<-gsub("8 ","OCHO ",devuelve)  
 devuelve<-gsub("9 ","NUEVE ",devuelve)  
 devuelve<-gsub("DR ","DOCTOR ",devuelve)  
 devuelve<-gsub("DR\\.","DOCTOR ",devuelve)  
 devuelve<-gsub("VTE","VICENTE",devuelve)  
 devuelve<-gsub("FCO","FRANCISCO",devuelve)  
 devuelve<-gsub("STA ","SANTA ",devuelve)  
 devuelve<-gsub("FDEZ ","FERNANDEZ ",devuelve)  
 devuelve<-gsub("Ã","Ã‘",devuelve)  
 devuelve<-gsub("Â¥","Ã‘",devuelve)  
   
 devuelve<-gsub(" I "," UNO ",devuelve)  
 devuelve<-gsub(" II"," DOS ",devuelve)  
 devuelve<-gsub(" III"," TRES ",devuelve)  
 devuelve<-gsub(" IV"," CUATRO ",devuelve)  
 devuelve<-gsub(" V "," CINCO ",devuelve)  
 devuelve<-gsub(" VI "," SEIS ",devuelve)  
 devuelve<-gsub(" VII"," SIETE ",devuelve)  
 devuelve<-gsub(" VIII"," OCHO ",devuelve)  
 devuelve<-gsub(" IX "," NUEVE ",devuelve)  
 devuelve<-gsub(" X "," DIEZ ",devuelve)  
 devuelve<-gsub(" XI"," ONCE ",devuelve)  
 devuelve<-gsub(" XII"," DOCE ",devuelve)  
 devuelve<-gsub(" XIII"," TRECE ",devuelve)  
 devuelve<-gsub(" XIV "," CATORCE ",devuelve)  
 devuelve<-gsub(" XV"," QUINCE ",devuelve)  
 devuelve<-gsub(" XVI"," DIECISEIS ",devuelve)  
 devuelve<-gsub(" XVII"," DIECISIETE ",devuelve)  
 devuelve<-gsub(" XVIII"," DIECIOCHO ",devuelve)  
 devuelve<-gsub(" IXX"," DIECINUEVE ",devuelve)  
 devuelve<-gsub(" XX"," VEINTE ",devuelve)  
 devuelve<-gsub(" XXI"," VEINTIUNO ",devuelve)  
   
 devuelve<-gsub("999"," ",devuelve)  
 devuelve<-gsub("9999"," ",devuelve)  
   
 devuelve  
}

## normalizaCODPOSTAL

normalizaCODPOSTAL<-function(cp){  
 devuelve<-paste(substr("00000",1,5-nchar(as.character(cp))),as.character(cp),sep="")  
 if (devuelve=="00000"){  
 devuelve=""  
 }  
 devuelve  
}

# Anexo 2: Información del sistema y la sesión de R

devtools::session\_info()

## Session info -------------------------------------------------------------

## setting value   
## version R version 3.4.1 (2017-06-30)  
## system x86\_64, mingw32   
## ui RTerm   
## language (EN)   
## collate Spanish\_Spain.1252   
## tz Europe/Paris   
## date 2017-08-03

## Packages -----------------------------------------------------------------

## package \* version date source   
## assertthat 0.2.0 2017-04-11 CRAN (R 3.4.1)  
## backports 1.1.0 2017-05-22 CRAN (R 3.4.0)  
## base \* 3.4.1 2017-06-30 local   
## bindr 0.1 2016-11-13 CRAN (R 3.4.1)  
## bindrcpp 0.2 2017-06-17 CRAN (R 3.4.1)  
## caRtociudad \* 0.5.2 2017-08-03 local   
## colorspace 1.3-2 2016-12-14 CRAN (R 3.3.2)  
## compiler 3.4.1 2017-06-30 local   
## curl 2.8.1 2017-07-21 CRAN (R 3.4.1)  
## datasets \* 3.4.1 2017-06-30 local   
## devtools 1.13.3 2017-08-02 CRAN (R 3.4.1)  
## digest 0.6.12 2017-01-27 CRAN (R 3.3.2)  
## dplyr 0.7.2 2017-07-20 CRAN (R 3.4.1)  
## evaluate 0.10.1 2017-06-24 CRAN (R 3.4.1)  
## geosphere 1.5-5 2016-06-15 CRAN (R 3.4.1)  
## ggmap \* 2.6.1 2016-01-23 CRAN (R 3.4.1)  
## ggplot2 \* 2.2.1 2016-12-30 CRAN (R 3.3.2)  
## glue 1.1.1 2017-06-21 CRAN (R 3.4.1)  
## graphics \* 3.4.1 2017-06-30 local   
## grDevices \* 3.4.1 2017-06-30 local   
## grid 3.4.1 2017-06-30 local   
## gtable 0.2.0 2016-02-26 CRAN (R 3.3.2)  
## highr 0.6 2016-05-09 CRAN (R 3.3.2)  
## htmltools 0.3.6 2017-04-28 CRAN (R 3.4.0)  
## httr 1.2.1 2016-07-03 CRAN (R 3.3.2)  
## jpeg 0.1-8 2014-01-23 CRAN (R 3.4.0)  
## jsonlite 1.5 2017-06-01 CRAN (R 3.4.1)  
## knitr \* 1.16 2017-05-18 CRAN (R 3.4.0)  
## lattice 0.20-35 2017-03-25 CRAN (R 3.4.1)  
## lazyeval 0.2.0 2016-06-12 CRAN (R 3.3.2)  
## magrittr 1.5 2014-11-22 CRAN (R 3.3.2)  
## mapproj 1.2-5 2017-06-08 CRAN (R 3.4.1)  
## maps 3.2.0 2017-06-08 CRAN (R 3.4.1)  
## memoise 1.1.0 2017-04-21 CRAN (R 3.4.0)  
## methods \* 3.4.1 2017-06-30 local   
## munsell 0.4.3 2016-02-13 CRAN (R 3.3.2)  
## pkgconfig 2.0.1 2017-03-21 CRAN (R 3.4.1)  
## plyr 1.8.4 2016-06-08 CRAN (R 3.3.2)  
## png 0.1-7 2013-12-03 CRAN (R 3.3.2)  
## proto 1.0.0 2016-10-29 CRAN (R 3.4.1)  
## purrr 0.2.3 2017-08-02 CRAN (R 3.4.1)  
## R6 2.2.2 2017-06-17 CRAN (R 3.4.1)  
## Rcpp 0.12.12 2017-07-15 CRAN (R 3.4.1)  
## reshape2 1.4.2 2016-10-22 CRAN (R 3.3.2)  
## rgdal \* 1.2-8 2017-07-01 CRAN (R 3.4.1)  
## RgoogleMaps 1.4.1 2016-09-18 CRAN (R 3.4.1)  
## rjson 0.2.15 2014-11-03 CRAN (R 3.4.0)  
## rlang 0.1.1 2017-05-18 CRAN (R 3.4.0)  
## rmarkdown 1.6 2017-06-15 CRAN (R 3.4.1)  
## rprojroot 1.2 2017-01-16 CRAN (R 3.3.2)  
## scales 0.4.1 2016-11-09 CRAN (R 3.3.2)  
## sp \* 1.2-5 2017-06-29 CRAN (R 3.4.1)  
## stats \* 3.4.1 2017-06-30 local   
## stringi 1.1.5 2017-04-07 CRAN (R 3.3.3)  
## stringr 1.2.0 2017-02-18 CRAN (R 3.3.2)  
## tibble 1.3.3 2017-05-28 CRAN (R 3.4.1)  
## tools 3.4.1 2017-06-30 local   
## utils \* 3.4.1 2017-06-30 local   
## withr 2.0.0 2017-07-28 CRAN (R 3.4.1)  
## yaml 2.1.14 2016-11-12 CRAN (R 3.3.2)