Distancias y Tiempos

con *R*

Francisco Goerlich

Universidad de Valencia e Ivie

23/06/2021

Contexto...

Distribución de la población y acceso a los servicios públicos

Inicio del proyecto: **2020**
Finalización del proyecto: **2021**
Entidad financiadora: **Fundación Ramón Areces**

El objetivo de esta monografía es analizar la localización de la población a escala municipal en relación a la disponibilidad de los servicios públicos fundamentales (centros sanitarios y educativos), así como de los servicios financieros (red de oficinas bancarias) y quizás algún otro de especial interés en función de la información disponible. El estudio está estrechamente relacionado con el fenómeno de la despoblación y la España vacía.

Cualquier ejercicio de accesibilidad (geográfica) tiene (al menos) 3 patas:

1. un **origen** por parte de quien accede,
2. una **ruta**, por algún medio de trasporte determinado, y
3. un **destino** al centro donde se accede.

Ya vimos como geocodificar destinos. ¡Como empezamos la casa por el tejado!, seguimos ahora con la determinación de la ruta para un **origen dado**.

¡En realidad solo estaremos interesados en distancias y tiempos de viaje!

Preliminares (1/2)

Los problemas de rutas pueden ser extremadamente complejos.

Si queremos ir de *A* a *B*, ¿como queremos ir?

En coche, a pie, en bicicleta, en tren, en barco –¿es posible?–, en avión.

¿En un único medio de transporte o combinando varios?

¿Podemos ir en línea recta o debemos utilizar una red?

Nos interesa solo la distancia y el tiempo aproximado que tardaremos, o queremos saber los puntos de paso, es decir la **ruta**.

¡Debemos ser específicos sobre todas estas cuestiones **antes** de empezar!

Preliminares (2/2)

Normalmente no podremos ir en linea recta, de forma que **necesitaremos una red de transporte**.

Si solo vamos a utilizar un modo de transporte tenemos un problema de **red unimodal**, y solo necesitaremos la red del modo de transporte correspondiente, la red de carreteras o de ferrocarril.

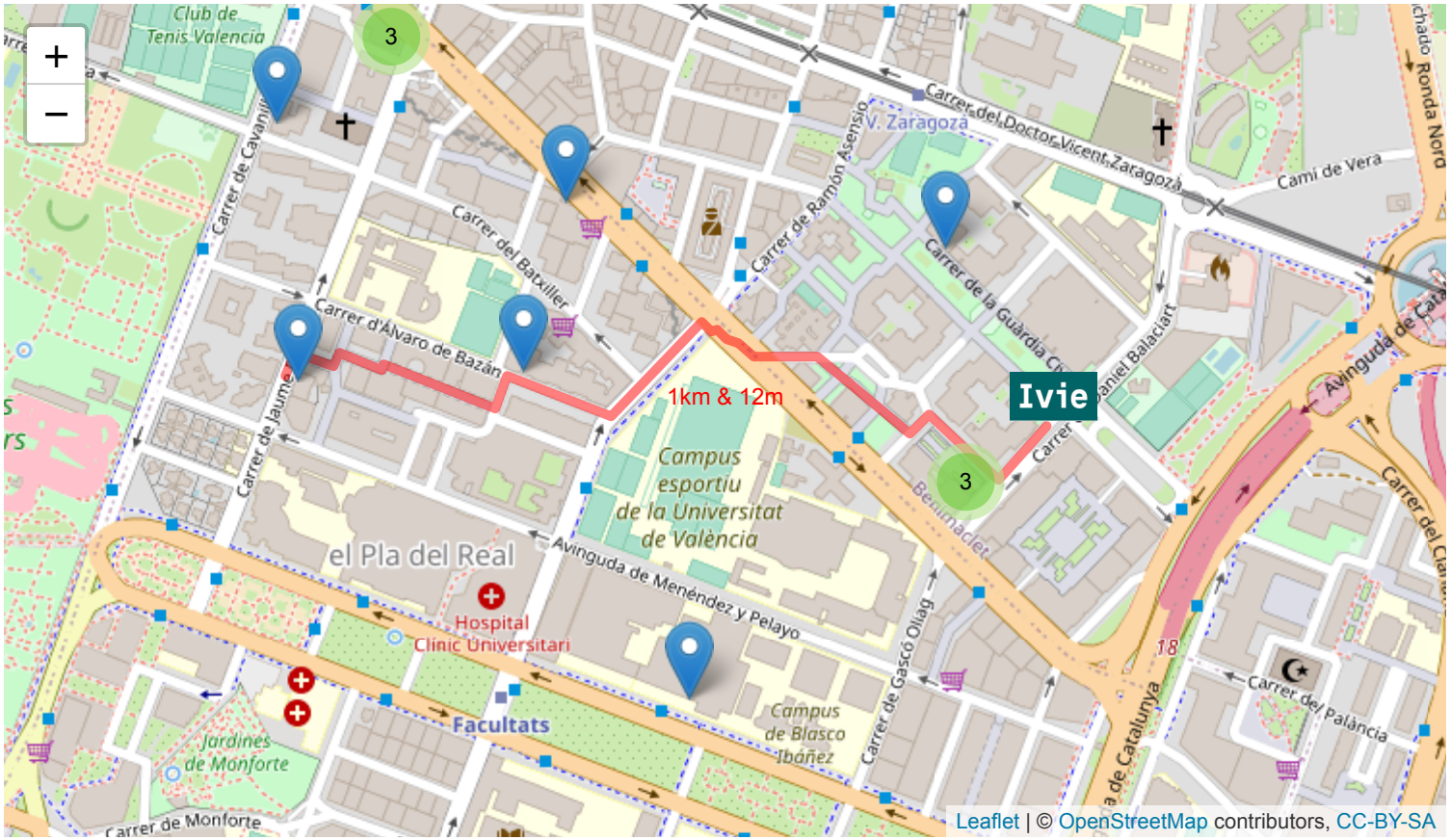
Si vamos a utilizar varios modos de transporte tenemos un problema de **red multimodal**, y necesitaremos los puntos de intercambio.

Si **no** somos autónomos en el medio de transporte elegido –coche, bicicleta, a pie,…– deberemos disponer de las frecuencias de paso y los puntos de parada.

Nuestro ejercicio lo simplifica todo al máximo. Queremos conocer la **distancia** y el **tiempo de viaje** aproximado entre dos puntos concretos en **coche privado** utilizando la **red de carreteras**.

¡Recordar! Las distancias son siempre más fiable que los tiempos de viaje, puesto que las primeras son algo físico, que se miden sobre la red, mientras que los segundos dependen de supuestos sobre velocidades del medio de transporte y las condiciones de tráfico.

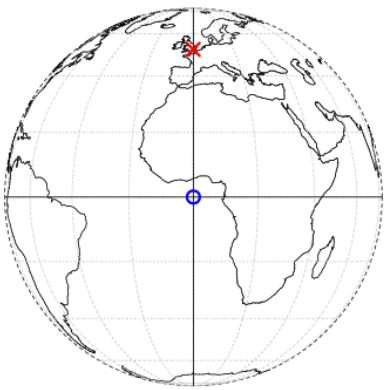
Retomamos el problema donde lo dejamos...



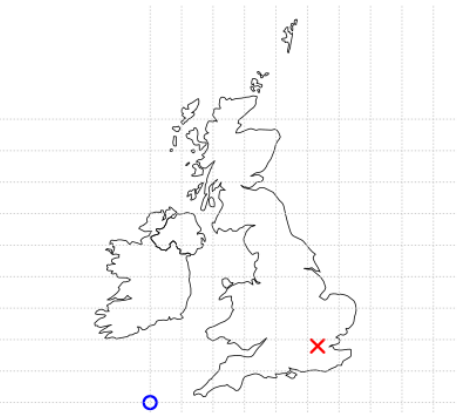
CRS: Coordinate Reference Systems

La información geográfica siempre está necesariamente ligada a un **Sistema de Referencia de Coordenadas (*CRS*)**. Hay básicamente 2 grupos de *CRS*.

Geográficos



Proyectados



IMPORANTE: Siempre que trabajemos con información geográfica debemos asegurarnos que **toda la información** está en el **mismo *CRS***.

Get Started

Problema real

Disponemos del fichero de Oficinas bancarias de diciembre de 2020 del Banco de España georeferenciado y queremos estimar la **distancia** y el **tiempo** de acceso de **cada municipio** a la **oficina más cercana**.

Importante: Las distancias se miden entre puntos, por tanto el municipio es un punto, la coordenada de la capital. Implicitamente suponemos que la población está concentrada en ese punto.

¿Cual es la dimensión de nuestro problema?

Origenes: 8 131 municipios.

Destinos: 22 558 oficinas.

La matriz de **O**rigen-**D**estino (OD) tiene 183 419 098 celdas.

¡Debemos calcular más de 183 millones de rutas y elegir la óptima para cada municipio!

Claramente debemos reducir la dimensión de nuestro problema.

Problemas a resolver

Tenemos, pues, 2 problemas a resolver:

1. Dados dos puntos debemos calcular la **ruta óptima** (menor distancia/tiempo).

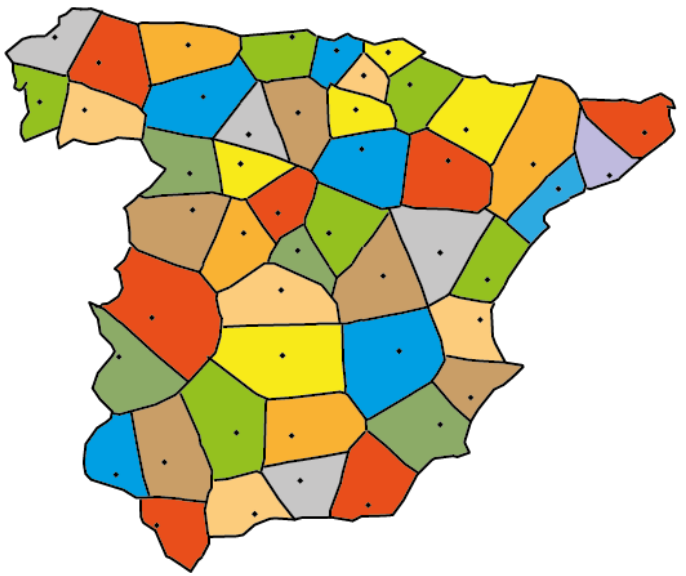
2. Debemos **acotar** el conjunto de **candidatos posibles** para el calculo de rutas hasta una dimensión razonable.

¡Como siempre empezamos por el final!

Acotar el conjunto de oficinas candidatas a las más cercanas de un municipio puede hacerse mediante lo que en *GIS* se conoce como determinación de **áreas de influencia**, ¡y naturalmente no hay una forma única de hacerlo!

¿Áreas de influencia?

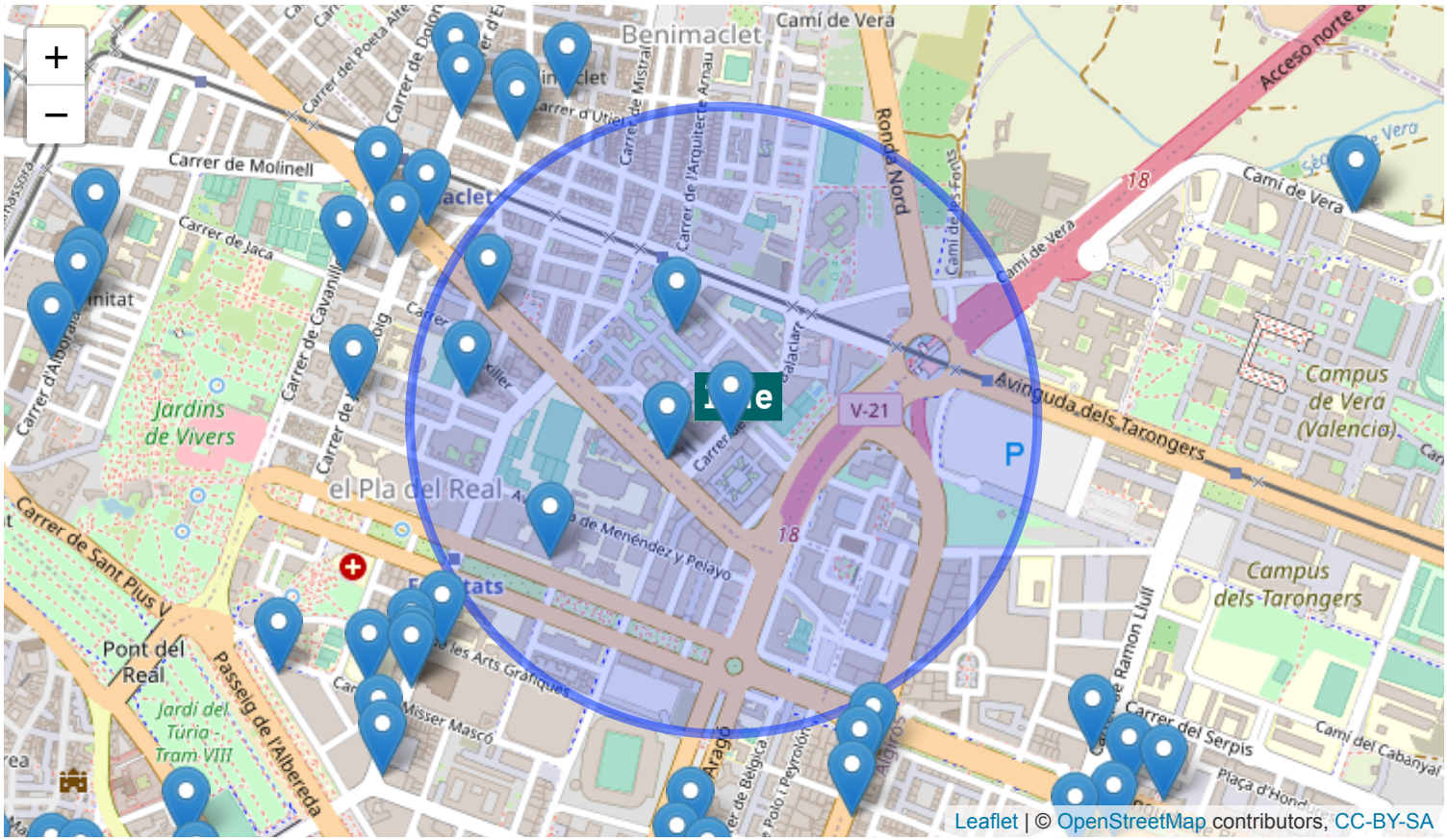
- Si **no disponemos de la red** es posible dividir, de forma **óptima***, el territorio en tantos poligonos como destinos tengamos –oficinas en nuestro caso– y asignar los orígenes a dichos polígonos. Una forma de hacer esto es la teselación de Voronoi (sf::st_voronoi).



(*) Por óptima entendemos que las lineas que determinan los polígonos son equidistantes a los puntos.

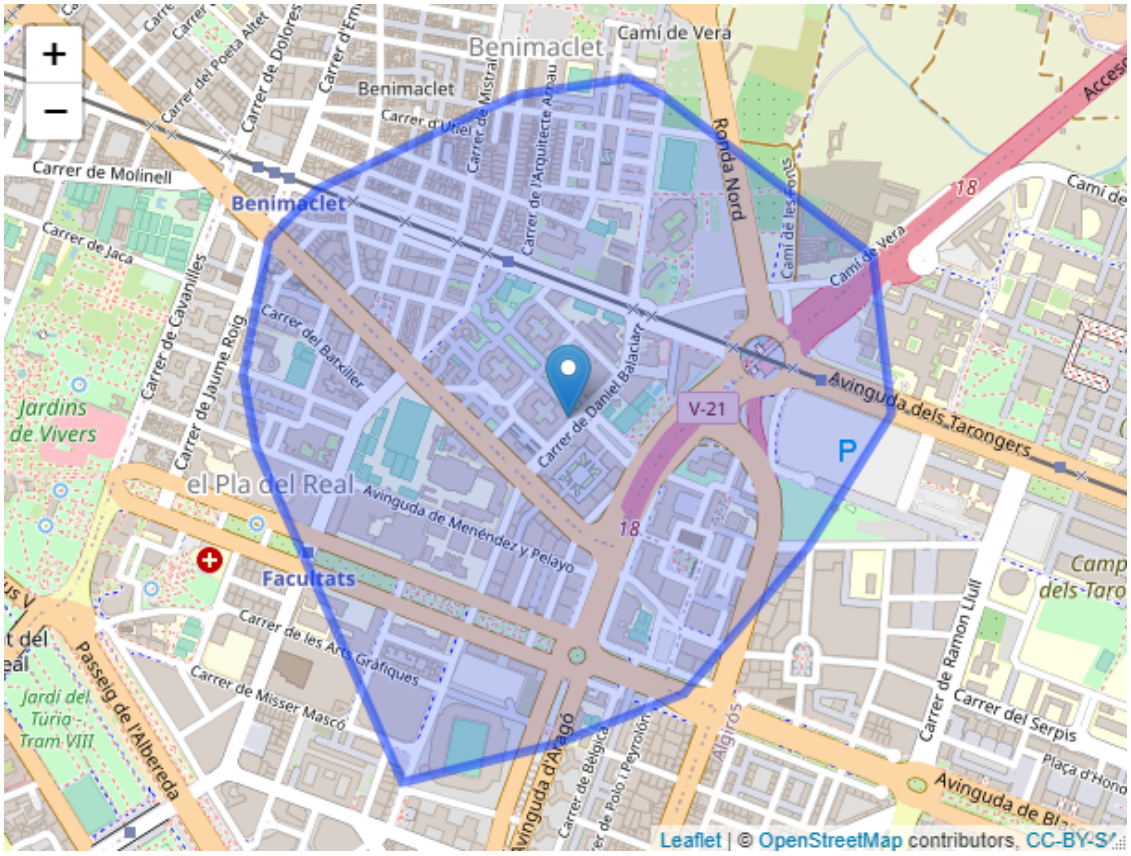
¿Áreas de influencia?

- ¿Otros métodos? **Si**. Incluso si seguimos **sin disponer de la red**.



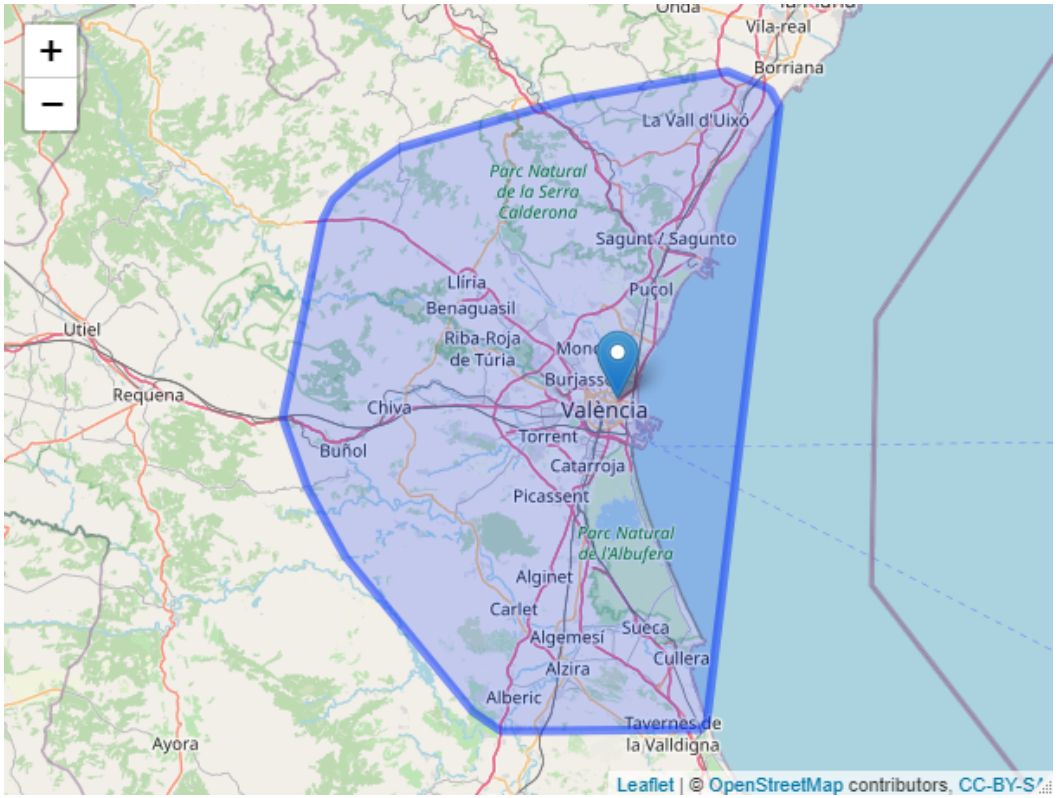
¿Áreas de influencia?

- **Si disponemos de la red** podemos afinar el método anterior. Podemos trazar un polígono a partir de los ejes de la red.



Áreas de influencia: Cartociudad

```
#   Área de influencia a partir de la red de Cartociudad
cartociudad_get_area <- function(longitud, latitud, distancia, plot =
```



Áreas de influencia: Monografía accesibilidad

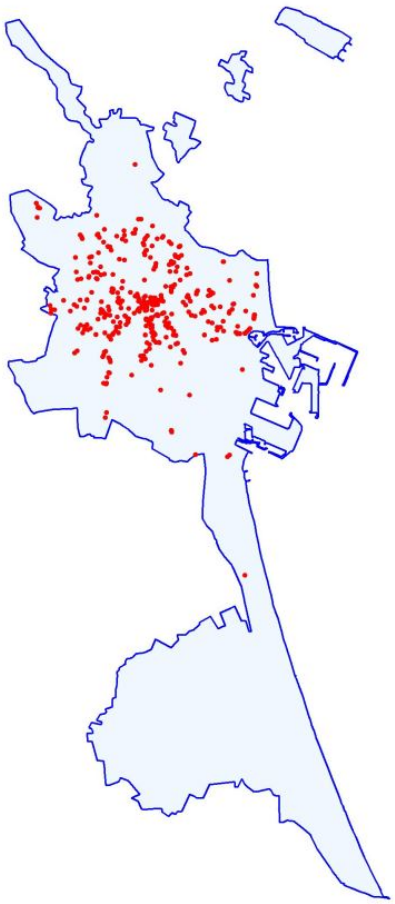
- ¿Cómo se generaron las áreas de influencia en la monografía de accesibilidad?

- Dos casos:

    1. Si el **municipios tiene oficina**, el **área de influencia** es el propio **término municipal**.

    2. Si el **municipio no tiene oficina**, el **área de influencia** viene determinada por un número determinado de **municipios vecinos**, entendidos como otros municipios con contigüidad física, con oficinas.

Municipios **con** Oficina



- En el caso de los **municipios con oficina** se calcularán distancias y tiempos a todas las oficinas del término municipial, que constituye su propia área de influencia.

- El **supuesto implícito** es que el acceso es a las oficinas del propio término municipal, de forma que no buscamos oficinas en otros municipios.

Municipios **sin** Oficina

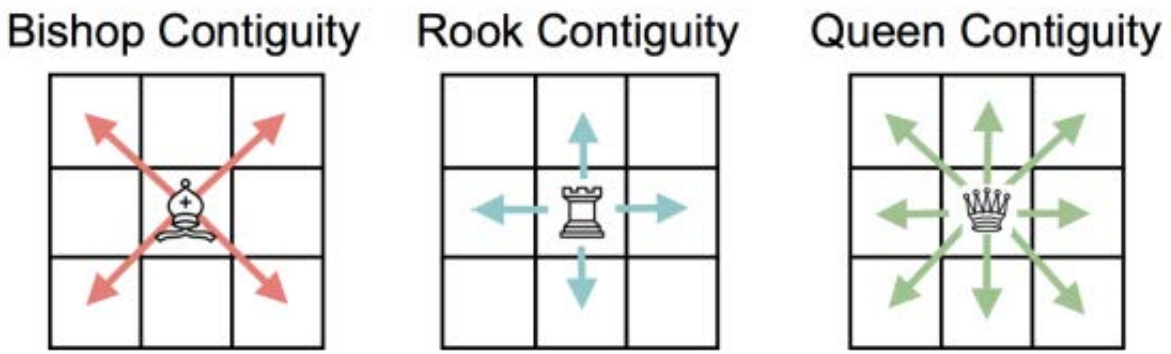
- En el caso de los **municipios sin oficina** el umbral de municipios vecinos en los que buscar la oficina más cercana se fijó en 20.
- El **supuesto implícito** es que la oficina más cercana está en los 20 vecinos más próximos que tienen oficina.
- La búsqueda de vecinos se hace por contiguidad física del término municipal, iterativamente a partir de la contigüidad de primer orden, hasta completar al menos 20 vecinos con oficinas.
- El algoritmo de búsqueda implica que tendremos, como mínimo, 20 municipios vecinos –de un orden de contigüidad no identificado a priori– con oficina, y al menos 20 candidatos –oficinas–, pero normalmente muchos más.

¿Como buscamos los vecinos?

- A partir del mapa de contornos municipales una función hace el trabajo:

```
# Given a POLYGON or MULTIPOLYGON sf and a primary key,
# it generates a list with the neighboring codes.
neighbor_codes <- function(poly, variable, type = c('queen', 'rook',
```

- Existen varios tipos de contigüidad, *queen*, *rook* y *bishop*, y consideramos el menos restrictivo.



Visualmente: Municipio **sin** oficina

Vecinos de **primer orden**

Vecinos de **segundo orden**

Vecinos de **tercer orden**

¡Solo si tienen oficina cuentan en el umbral!

Municipio y su área de influencia

Distancias y tiempos.

- Una vez tenemos **orígenes** –municipios– y **destinos** candidatos –oficinas– estamos en disposición de calcular **distancias** y **tiempos de viaje**.
- Para ello necesitamos una **red de carreteras** con los atributos correspondientes. Básicamente la longitud y velocidad de cada tramo.
- La longitud se puede medir sobre la propia red.
- El tiempo de viaje necesita supuestos sobre velocidades –reales o teóricas– del medio de transporte.
- La **ruta óptima** entre 2 puntos es un problema estándar en GIS, se resuelve mediante alguna variante del algoritmo de Dijkstra y está implementado en cualquier *software* o librería de GIS que tenga funcionalidades de redes (sfnetwroks). ¡Esto **no** es un problema!

Distancias y tiempos.

- Existen 2 posibilidades para llevar a cabo este tipo de cálculos:

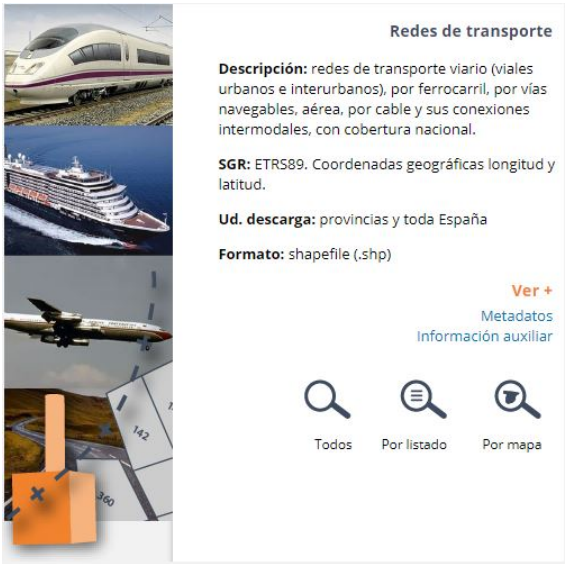
Red en local

1. Costosa de cargar y mantener.
2. ¿Atributos?
3. Requiere sistemas de cálculo medianos.
4. Permite contrafactuales.
5. Resultados replicables.
6. ¡Controlamos la red!

Red en remoto

1. Acceso mediante **API**.
2. No necesitamos realmente la red física.
3. No requiere grandes sistemas.
4. No permite contrafactuales.
5. ¿Replicabilidad?
6. ¡No controlamos la red!

Red Local: IGN - Redes de Transporte



1. Información Geográfica de Referencia.
2. Fuente oficial (IGN).
3. Multimodal (¡pero podemos coger lo que necesitemos!).
4. Carece de atributos.
5. ¡Controlamos la red!
6. Costosa de implementar y mantener.
7. ¡Gratuita!

Red Local: TomTom - *Speed Profile*

View of TomTom Speed Profiles

Barcelona

Sants

Madrid

Atocha

1. Elevada calidad y resolución.
2. Rica en atributos.
3. ¡Datos de tráfico reales! y por cortes diarios y temporales.
4. Unimodal.
5. ¡Controlamos la red!
6. Sencilla de implementar y mantener.
7. Elevado coste.

Red remota: ¿Alternativas?

- Enviamos los orígenes y destinos a un servicio de *routing*. El mundo de los servicios tiene *pros* y *cons*, pero resulta más sencillo que el mantenimiento de una red en local.
- Opción elegida para la monografía de accesibilidad.

Cartociudad *versus* Google

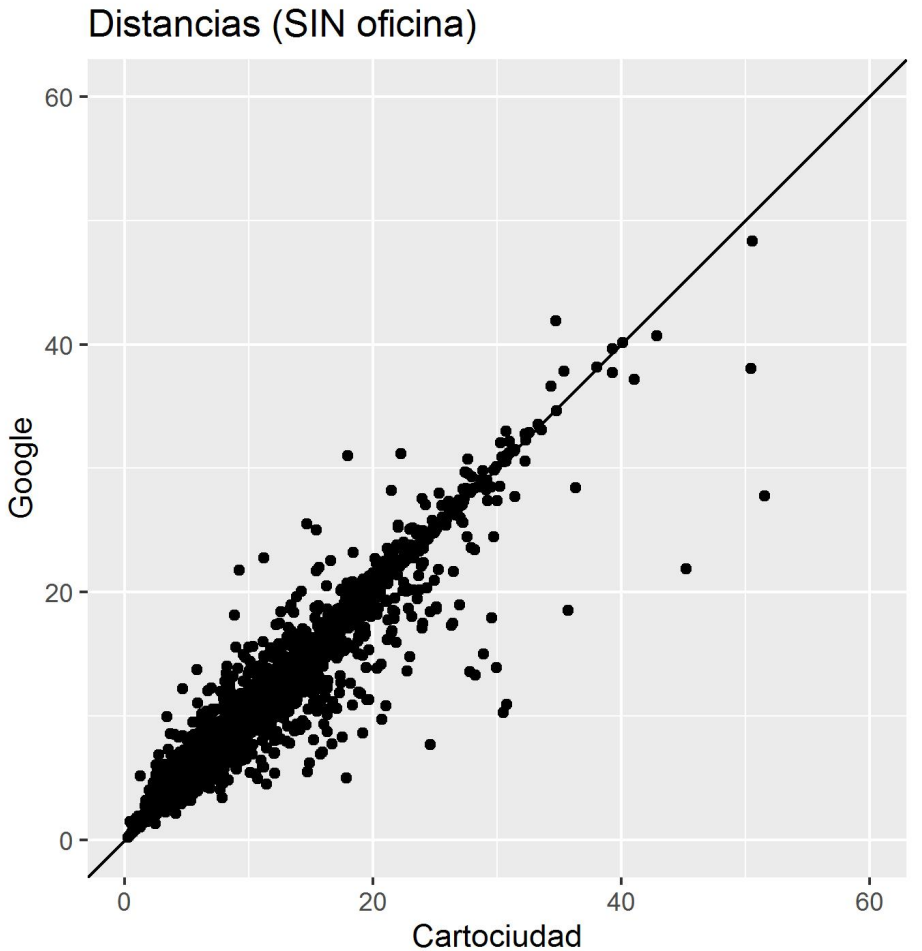
Cartociudad

1. Cobertura nacional.
2. Servicio gratuito e ¡**ilimitado**!
3. Documentación de la API.
4. No es fiable en tiempos de viaje en coche.

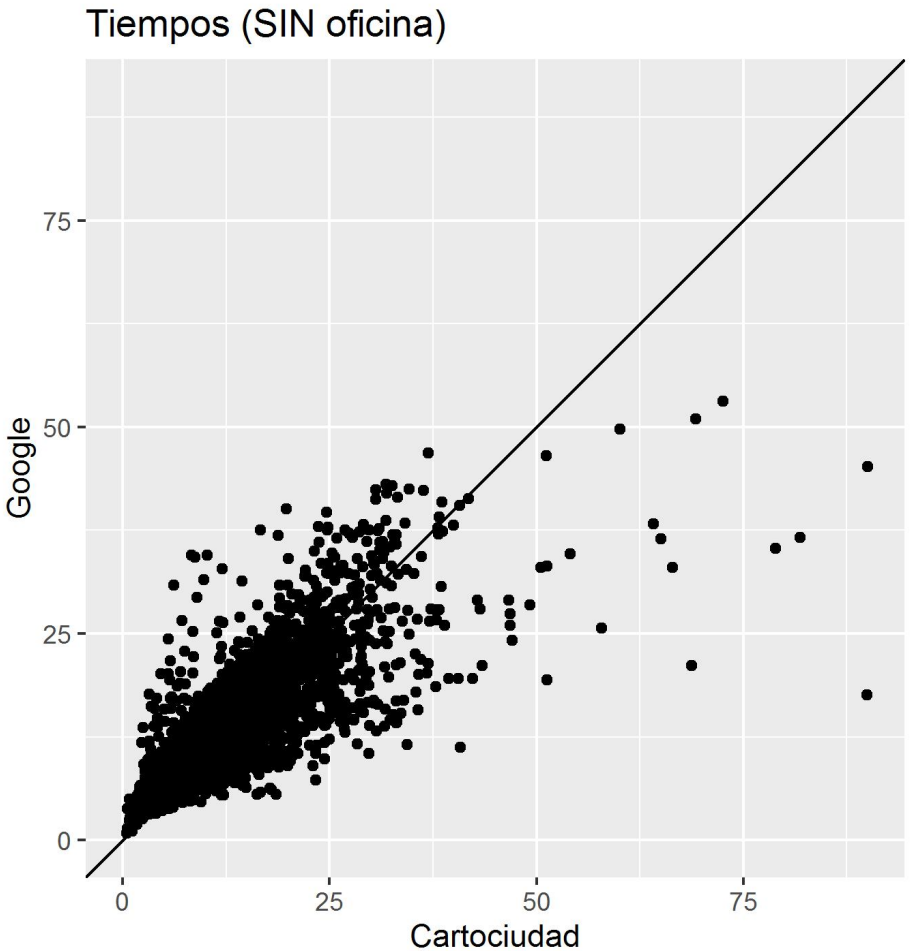
Google

1. Cobertura mundial.
2. Servicio de pago y ¡**limitado**!
3. Documentación de la API.
4. Fiable en distancias y tiempos.

Cartociudad *versus* Google: Distancias



Cartociudad *versus* Google: Tiempos



Monografía accesibilidad

- Dada la mayor fialibilidad de Google, pero al mismo tiempo su elevado coste para un gran número de consultas se implementó una estrategia en dos etapas*:

1. Todos los candidatos determinados en el **área de influencia** se enviaron a Cartociudad: ¡Unas 600 000 rutas y un par de días de cálculo!

2. A partir de las rutas para todos los candidatos se hizo una **selección por distancia** que se envió a Google: Las **4 oficinas** más cercanas de **4 municipios diferentes**.

Se enviaron a Google, además, todas las rutas que Cartociudad no encuentra, ¡porque Cartociudad no lo encuentra todo!

(*) Para los municipios con oficina se enviaron a Google todas las oficinas del municipio.

Routing Cartociudad: Funciones

```
# Distancia y Tiempo de Viaje entre 2 puntos (coordenadas)
cartociudad_get_distance <- function (from, to, vehicle = c("car", "w

# Distancia y Tiempo de Viaje entre 2 vectores de coordenadas
cartociudad_distance <- function(from, to, vehicle = c("car", "walk")
```

- `cartociudad_get_distance()`: ruta única entre 2 puntos (coordenadas) en ETRS89.
- `cartociudad_distance()`: rutas masivas *pairwise* entre 2 vectores de puntos (coordenadas), o entre un origen y múltiples destinos o entre múltiples orígenes y un destino en ETRS89.

Calcula rutas en coche o a pie.

En ambos casos devuelve un `tibble` con la distancia (en metros), el tiempo de viaje (en segundos) y una variable de *status*: `OK` o `NOT_FOUND`.

Routing Google: Funciones

```
# Distancia y Tiempo de Viaje entre 2 puntos (coordenadas)
google_get_distance <- function (from, to, mode = c("driving", "walk

# Distancia y Tiempo de Viaje entre 2 vectores de coordenadas
google_distance <- function(from, to, mode = c("driving", "walking",
```

- `google_get_distance()`: ruta única entre 2 puntos (coordenadas) en WGS84.
- `google_distance()`: rutas masivas *pairwise* entre 2 vectores de puntos (coordenadas), o entre un origen y múltiples destinos o entre múltiples orígenes y un destino en WGS84.

Calcula rutas en coche, a pie, en bicicleta o multimodales.

Los resultados dependen (marginalmente) del momento de la consulta, porque Google dispone información de tráfico por franjas horarias y días.

En ambos casos devuelve un `tibble` con la distancia (en metros), el tiempo de viaje (en segundos) y una variable de *status*: `OK` o `NOT_FOUND`.

¿Hay alternativas a Cartociudad o Google?

Sí.

Google tiene versión *OpenSource*: Open Street Maps, (OSM).

- La información de OSM es accesible mediante **API** o descarga directa.
- La información se actualiza diariamente: Geofabric.
- Dispone de un servicio de geocodificación: Nominatim.
- Dispone de un servicio de *routing*: OSRM.
- ¡Es posible instalar OSM en local! ¡En el Ivie hay un servidor de OSM!
- ¡Es (monetariamente) gratis!
- Hay librerías de *R* que acceden a todo: datos, geocodificación, *routing*,... en remoto o en local.

Routing OSM: Funciones

```
# Distancia y Tiempo de Viaje entre 2 puntos (coordenadas)
osm_get_distance <- function (from, to, profile = c("car", "bike", "f

# Distancia y Tiempo de Viaje entre 2 vectores de coordenadas
oms_distance <- function(from, to, profile = c("car", "bike", "foot")
```

- `osm_get_distance()`: ruta única entre 2 puntos (coordenadas) en WGS84.
- `oms_distance()`: rutas masivas *pairwise* entre 2 vectores de puntos (coordenadas), o entre un origen y múltiples destinos o entre múltiples orígenes y un destino en WGS84.

Calcula rutas en coche, a pie o en bicicleta.

La opción `server` permite acceder a un servidor local.

En ambos casos devuelve un `tibble` con la distancia (en metros), el tiempo de viaje (en segundos) y una variable de *status*: `OK`, `NULL_VALUES` o `NOT_FOUND`.

¿Y si queremos pintar las rutas?

Para ilustrar las rutas...

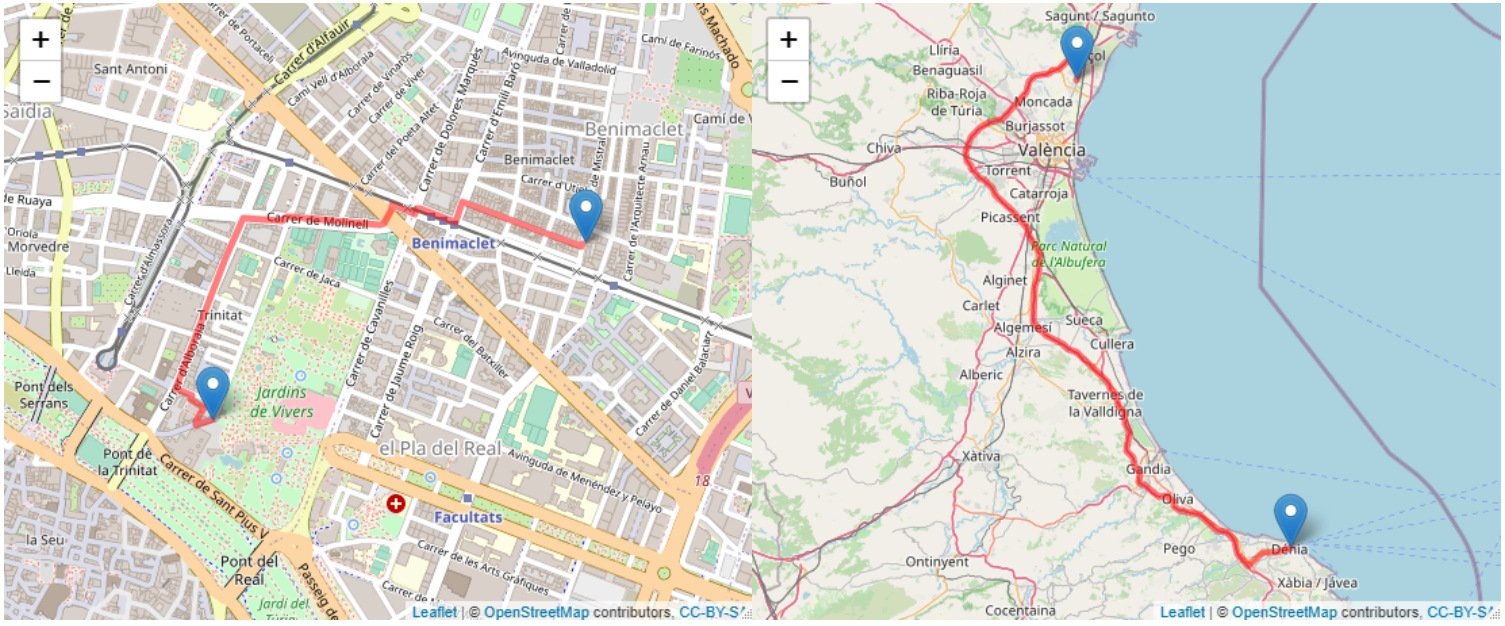
...disponemos de las mismas funciones, pero en lugar de devolver un `tibble` con los resultados de distancia/tiempo devuelve una *simple feature* con esos atributos y la geometría de la ruta para operar con ella o pintarla a efectos ilustrativos.

```
# Cartociudad: Ruta entre 2 puntos (coordenadas)
cartociudad_get_route <- function(from, to, vehicle = c("car", "walk'
# Cartociudad: Rutas entre 2 vectores de coordenadas
cartociudad_routing <- function(from, to, vehicle = c("car", "walk"),

# Google: Ruta entre 2 puntos (coordenadas)
google_get_route <- function(from, to, mode = c("driving", "walking",
# Google: Rutas entre 2 vectores de coordenadas
google_routing <- function(from, to, mode = c("driving", "walking", '

# OSM: Ruta entre 2 puntos (coordenadas)
osm_get_route <- function(from, to, profile = c("car", "bike", "foot'
# OSM: Rutas entre 2 vectores de coordenadas
osm_routing <- function(from, to, profile = c("car", "bike", "foot"),
```

Ejemplos...


Sant Antoni
Benimaclet
Trinitat
Jardins de Vivers
Pont dels Serrans
Pont de la Trinitat
el Pla del Real
Facultats
Pont del Real
la Seu
Sagunt / Sagunto
Llíria
Benaguasil
Riba-roja de Túria
Moncada
Burjassot
Chiva
València
Torrent
Buñol
Catarroja
Picassent
Alginet
Carlet
Algemesí
Sueca
Cullera
Alzira
Alberic
Xàtiva
Gandia
Oliva
Pego
Dénia
Ontinyent
Xàbia / Jávea
Cocentaina
Leaflet | © OpenStreetMap contributors, CC-BY-S

Ivie

¡Gracias por la atención!

¿Preguntas?

VNIVERSITAT DE VALÈNCIA