Aprendizaje de Maquina

Vallarino (175875) - Torre (90226)

10/12/2017

Aprendizaje de Máquina

Integrantes:

.- Arturo Torre - 90226

.- Ariel Vallarino - 175875

DBSCAN

(**D**ensity **B**ased **S**patial **C**lustering of **A**pplications with **N**oise)
Clustering basado en densidad (1996)

.- Clustering:

Objetivo: Encontrar grupos de instancias tales que las instancias en un cluster sean similares entre sí, y diferentes de las instancias en otros clusters, donde se Minimiza la distancia Intra-Cluster y Maximiza la distancia Inter-Cluster. Para esto, la mejor definición de cluster depende de la naturaleza de los datos y los resultados deseados.

.- DBSCAN

¿En qué consiste?
Tiene un enfoque basado en la densidad, modelando los clusters como cúmulos de alta densidad de puntos. Por lo cual, si un punto pertenece a un clúster, debe estar cerca de un montón de otros puntos de dicho clúster. Se basa en el concepto de densidad de un punto, que mide el número de puntos que son alcanzables desde él considerando un determinado radio.
Un cluster en una región densa de puntos, separada por regiones poco densas de otras regiones densas.

¿Cómo funciona?
Se definen dos parámetros, un número **épsilon** (***Eps***) positivo y un número natural **minPoints** (***MinPts***), y se elige un punto arbitrario ***x*** en el conjunto de datos. Si ***x*** es un punto central (*core points*), se empieza a construir un cluster alrededor de él, tratando de descubrir componentes denso-conectadas. A continuación se elige un nuevo punto arbitrario (***y***) y se repite el proceso el cual termina cuando no se pueden añadir nuevos puntos a ningún clúster.
Vecindad de un punto ***x***: esfera centrada en ***x*** y radio ***Eps***
Densidad de un punto ***x***: cantidad de puntos dentro de la vecindad de ***x***.

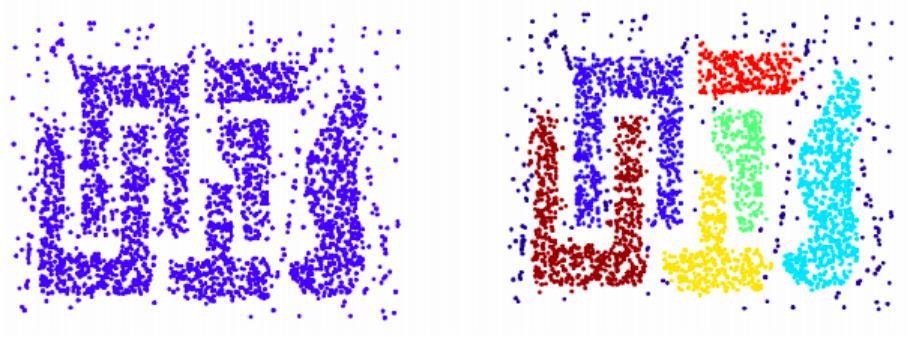
Puntos centrales (*core points*) son aquellos tales que en su vecindad de radio ***Eps***, hay una cantidad de puntos mayor o igual que un umbral ***MinPts*** especificado. Son los puntos interiores de un cluster.
Un punto *borde* o *frontera* tiene menos puntos que ***MinPts*** en su vecindad, pero pertenece a la vecindad de un punto central.
Un *punto ruido* (*noise*) es aquel que no es ni central ni borde.

En consecuencia, los *puntos centrales* están en regiones de alta densidad, los *puntos borde* en la frontera de regiones densas y los *puntos ruido* en regiones de baja densidad. Este algoritmo busca clusters comprobando la vecindad de cada punto de la base de datos y va añadiendo puntos que son denso-alcanzables desde un punto central.

Un punto ***q*** es directamente denso-alcanzable desde otro punto ***t*** (con relación a los parámetros ***MinPts*** y ***Eps***) si ***t*** es un punto central y ***q*** pertenece a la vecindad de ***t***. Un punto ***q*** es denso-alcanzable desde un punto ***t*** si existe una cadena de puntos t_0, t_1**,...** t_m, tales que t_{i-1} es directamente denso-alcanzable desde t_i, 1 ≤ i ≤ m, t_0 **= q** y t_m **= t**.

En los puntos de un mismo cluster, su k-ésimo vecino debería estar más o menos a la misma distancia. En los puntos de ruido, su k-ésimo vecino debería estar más lejos.

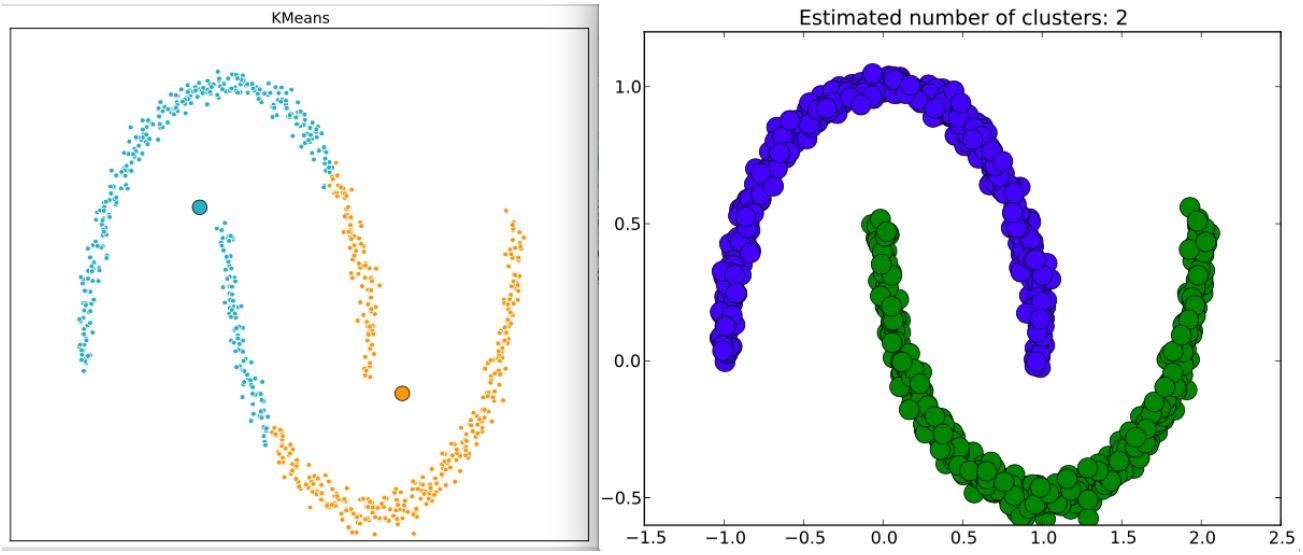
Datos originales vs Cluster DBSCAN:



En esta imagen podemos ver cómo quedaría un agrupamiento luego de utilizar DBSCAN.

Comparación con otros modelos:

K-Means vs DBSCAN:



Con K-Means vemos que los clusters quedan formados por puntos de ambas curvas, mientras que con DBSCAN se pueden separar fácilmente ambas regiones en grupos distintos.

Fortalezas

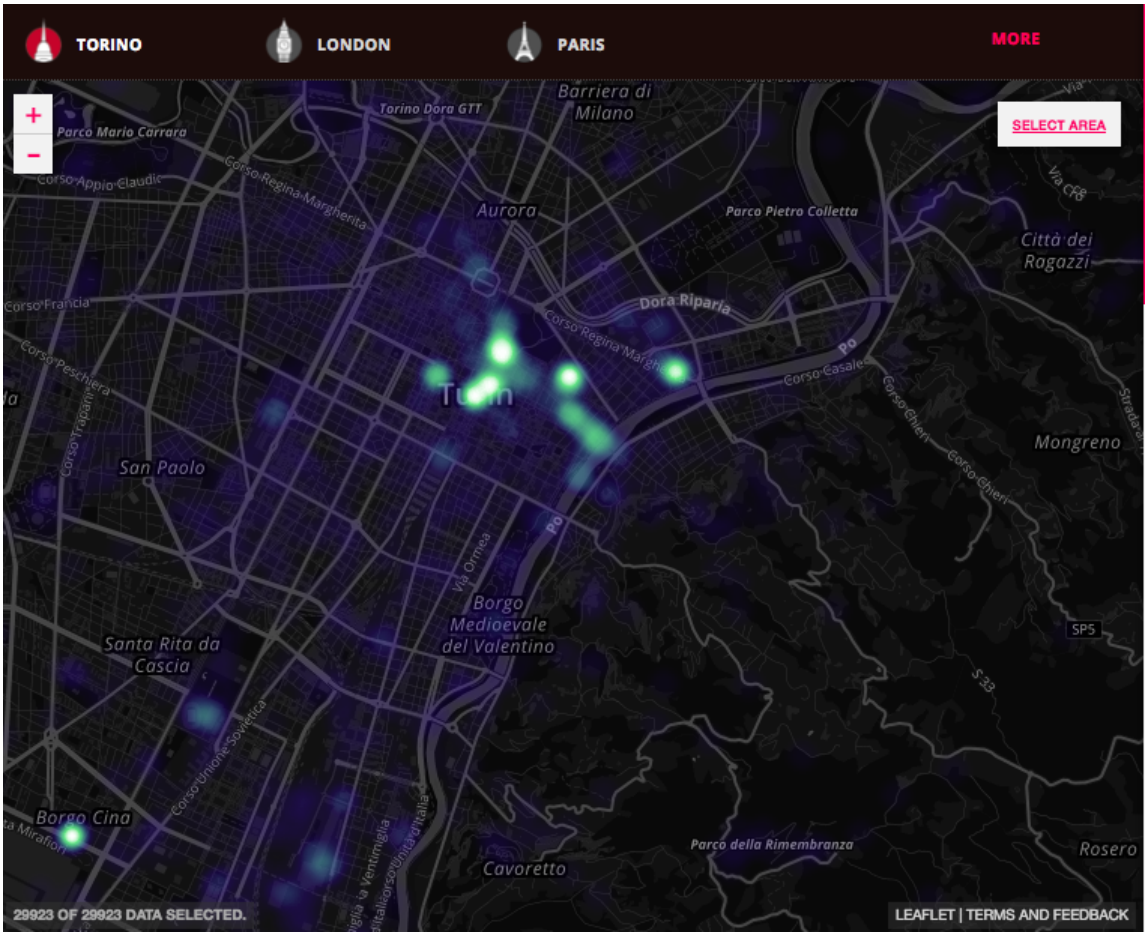
- Encuentra clusters no separables linealmente.
- Identifican clusters de formas arbitrarias.
- No necesita asumir un número fijo de clusters (No hay qye especificar k).
- No depende de las condiciones de inicio.
- Robusto ante la presencia de ruido.
- Es de una sola pasada.

Debilidades

- Asume densidades similares en todos los clusters.
- Puede tener problemas al separar clusters.
- Elegir Eps y MinPts puede requerir tener conocimiento de los datos, y puede ser dificil en casos de alta dimensionalidad.
- No es bueno para datos de alta dimensionalidad, grupos de diferentes densidades y grupos muy solapados.

Ejemplo:

Fuente: http://www.datainterfaces.org/projects/flickr/#torino (http://www.datainterfaces.org/projects/flickr/#torino)
(*La descarga de los datos se realizó haciendo uso de técncias básicas de web scraping*)



¿Qué es Flickr?
Flickr es una plataforma digital, propiedad de Yahoo, que permite guardar, buscar y compartir fotografías y videos en Internet. Servicios similares incluyen a plataformas como Instagrama o Pinterest.

En un esfuerzo por abrir sus datos al público Flickr, en conjunto con YahooLabs, crearon el Flickr cities. El proyecto consiste el desarrollo de una interface exploratoria de millones de imágenes que se han compartido mediante la plataforma en distintas ciudades, que se mapean en una interface visual (un mapa). Dentro de los metadatos de cada fotografía podemos encontrar el lugar, la fecha, el tema, la nacionalidad del usuario y la hora.

Lo anterior nos permite contar con información valiosa de los patrones y hábitos que desarrollan los turistas de cierta nacionalidad a cierta hora del día.

¿Cómo se usará la base de datos y el DBSCAN?
El proyecto de Flickr cities cuenta con información de ciudades como París, Londres, Tokio, Buenos Aires o Torino. Para este ejercicio se utilizarán los datos de Torino por ser de un tamaño razonable comparado con ciudades más grandes y con mayor volumen de turistas.

Dado que el el algoritmo de DBSCAN se base en la densidad de datos principalmente, entendemos que en la ciudad existirán lugares con grandes cantidades de fotografías que denominaremos como lugares de interés general. Por ejemplo, pudiésemos encontrar monumentos históricos, museos, iglesias, estadios de fútbol, universidades, parques públicos entre otros.
Consideraremos como *ruido* a las fotograífas espontáneas que se toman en cualquier punto de la ciudad.
Básicamente, eliminando el “ruido” esperamos encontrar clústers identificados por el algoritmo cerca de lugares de interés general.

```
library(tidyverse)
library(dbscan)
```

Datos de Torino:

```
torino <- read.csv("clean_points_torino_full.csv")
DT::datatable(head(torino[,2:7], 10), escape = FALSE,
              caption = htmltools::tags$caption(style = 'caption-side: bottom; text-align: center;',
                                                'Table: ', htmltools::em('Torino - Fotos')))
```

```
## Warning in instance$preRenderHook(instance): It seems your data is too big
## for client-side DataTables. You may consider server-side processing: https://
## rstudio.github.io/DT/server.html
```

Show 10 entries Search:

	longitude	latitude	date_taken_corrected	is_camera	capture_device	title
1	7.62833	45.13385	1228349861000	1	SONY DSLR-A900	The Sony BRAVIA-drome at night
2	7.67807	45.06916	1139799450000	1	Canon EOS 20D	kk's fan club
3	7.68961	45.07213	1204579437000	1	EASTMAN KODAK COMPANY KODAK EASYSHARE M1033 DIGITAL CAMERA	100_1061
4	7.7673	45.08065	1371300228000	1	Canon EOS 20D	Basilica Superga - 3
5	7.74541	45.14642	1184782016000	1	Canon EOS 350D DIGITAL	Staircase to hell
6	7.69845	45.06424	1193938032000	1	Canon PowerShot S5 IS	welcome in Turin
7	7.65228	45.04161	1330802312000	1	NIKON CORPORATION NIKON D90	NOTJ Torino 2012 DSC_1757
8	7.69112	45.0603	1292662476000	1	NIKON CORPORATION NIKON D3000	Museo Nazionale del Cinema, Turim
9	7.68562	45.06999	1233676600000	1	Plustek OpticFilm 7200	Gelato, Torino
10	7.67601	45.074	1361016512000	1	Canon EOS 5D Mark III	Torino

Table: *Torino - Fotos*

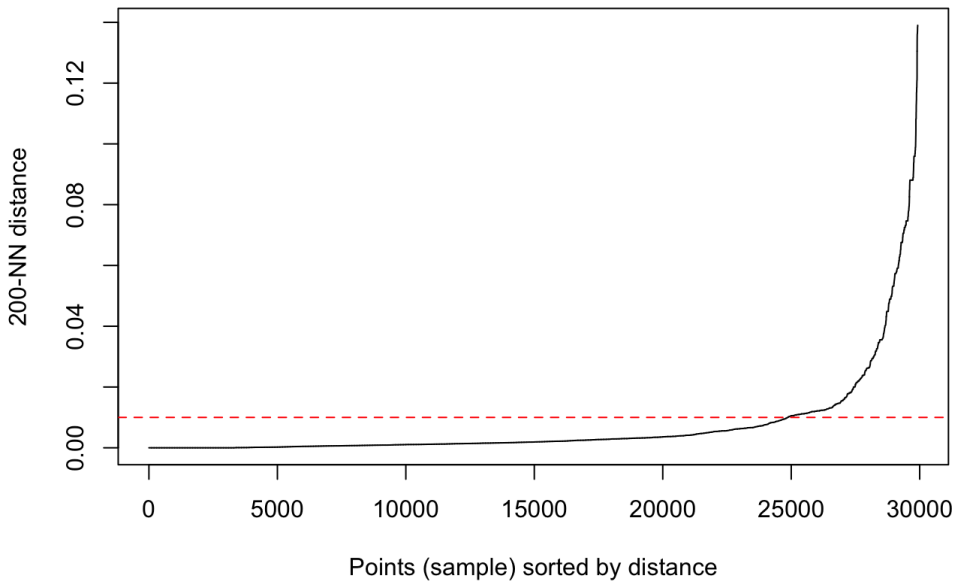
Showing 1 to 10 of 10 entries

Previous 1 Next

Graficamos curva para determinar *Eps*

Con la gráfica de codo podemos determinar el valor del *Eps* (radio) optimo para cierta cantidad de puntos:

```
df_torino <- torino[2:3]
kNNdistplot(df_torino, k = 200)
abline(h=0.01, col = "red", lty=2)
```



Aplicamos DBSCAN y visualizamos los clústers obtenidos

Los parámetros definidos son:

eps = 0.01

minPts = 200

```
res <- dbscan(df_torino, eps = 0.01, minPts = 200)
res
```

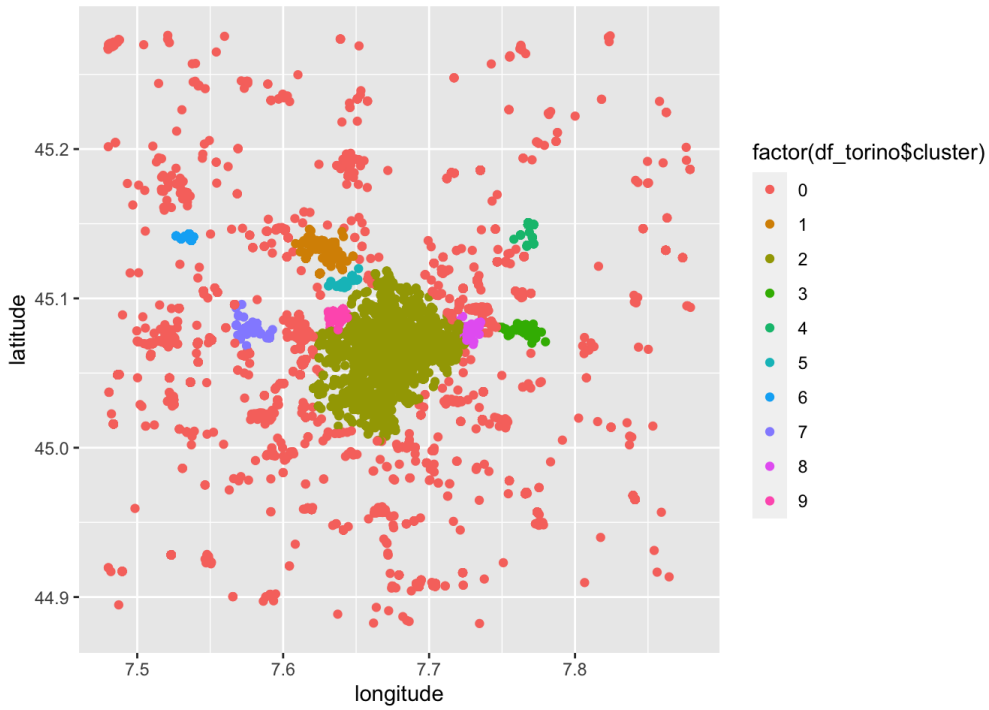
```
## DBSCAN clustering for 29923 objects.
## Parameters: eps = 0.01, minPts = 200
## The clustering contains 9 cluster(s) and 3226 noise points.
##
##     0     1     2     3     4     5     6     7     8     9
##  3226  1001 23603   353   247   225   568   342   183   175
##
## Available fields: cluster, eps, minPts
```

Obtenemos la cantidad de clústers y la cantidad de puntos considerados como *ruido* (clúster 0).

Graficamos clústers

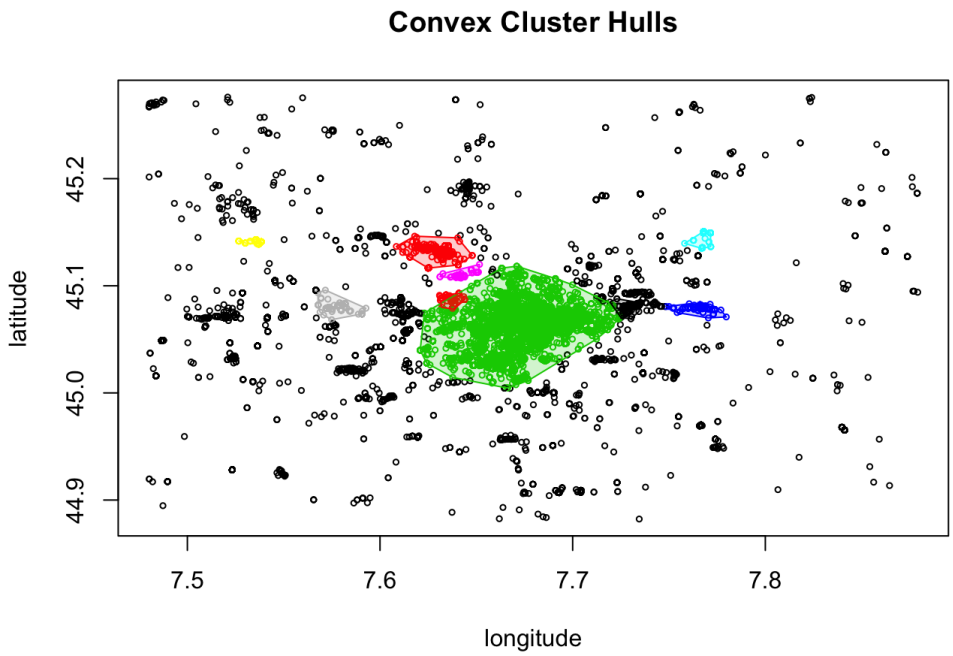
```
df_torino$cluster <- res$cluster
ggplot(df_torino, aes(x=longitude, y=latitude, colour=factor(df_torino$cluster))) +
  geom_point()
```

```
## Warning: Use of `df_torino$cluster` is discouraged. Use `cluster` instead.
```




```
hullplot(df_torino[1:2], res)
```

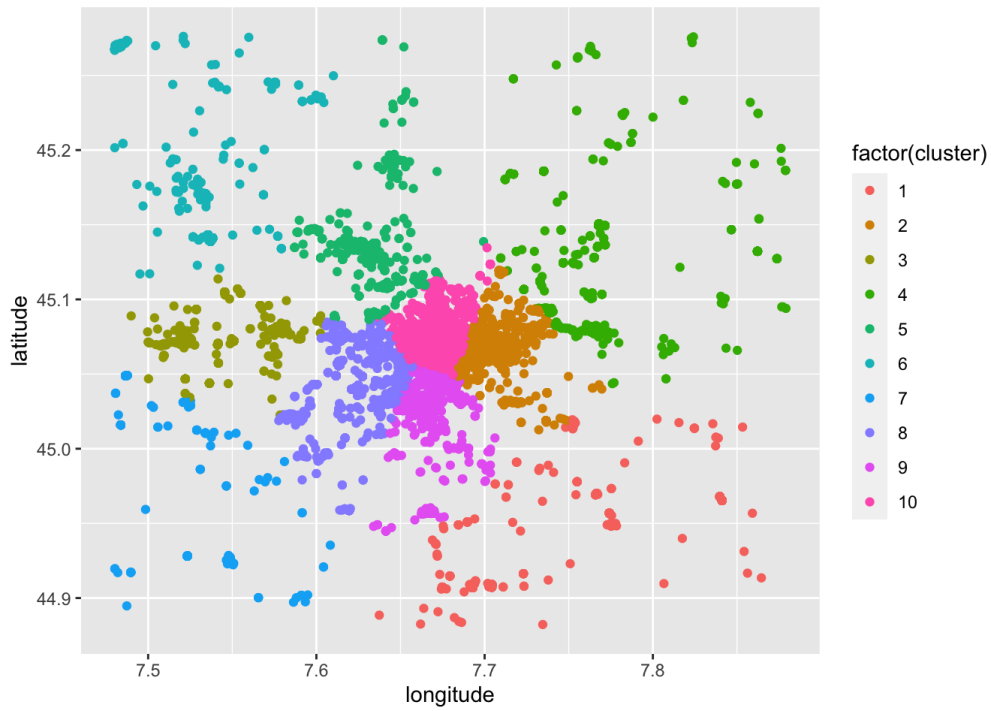
```
## Warning in hullplot(df_torino[1:2], res): Not enough colors. Some colors will be
## reused.
```



Comparación de clustering con K-Means:

Definimos la misma cantidad de grupos obtenidos con DBSCAN. **k = 10**

```
df_torino_KM <- torino[2:3]
df_torino_KM$cluster <- kmeans(df_torino_KM, centers=10, nstart=25)$cluster
ggplot(df_torino_KM, aes(x=longitude, y=latitude, colour=factor(cluster))) +
  geom_point()
```



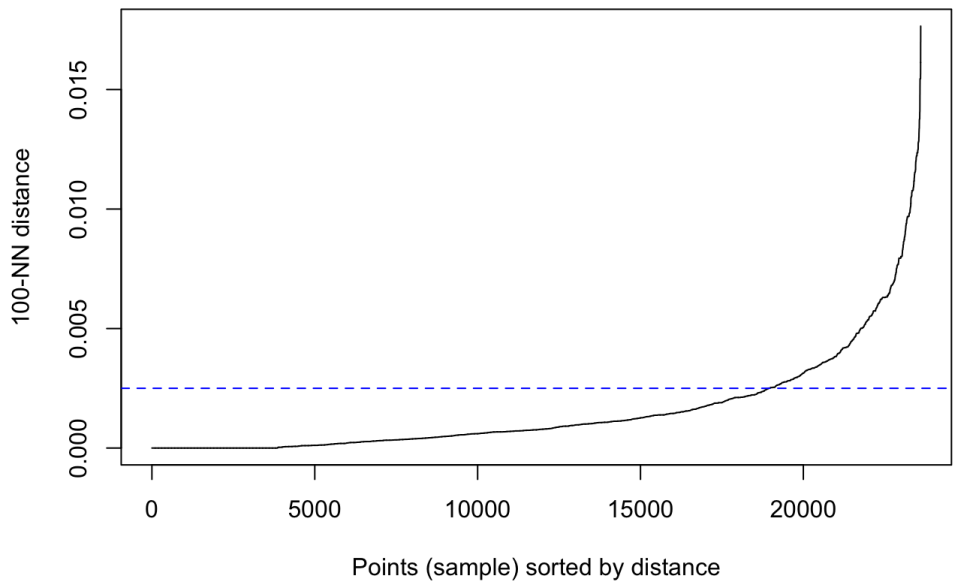
En éste caso, al armar los clústers con K-Means vemos todos los puntos (imagenes) forman parte de algún grupo, donde tambien se consideran las fotos espontáneas las cuales no corresponden a algún sitio de interés general.

Regresando al agrupamiento realizado con DBSCAN, tomamos los puntos del clúster mas grande:

```
df_centro <- df_torino %>%
  filter(cluster == 2)
```

Graficamos curva para determinar *Eps*

```
kNNdistplot(df_centro[1:2], k = 100)
abline(h=0.0025, col = "blue", lty=2)
```



Aplicamos DBSCAN y visualizamos los clústers obtenidos

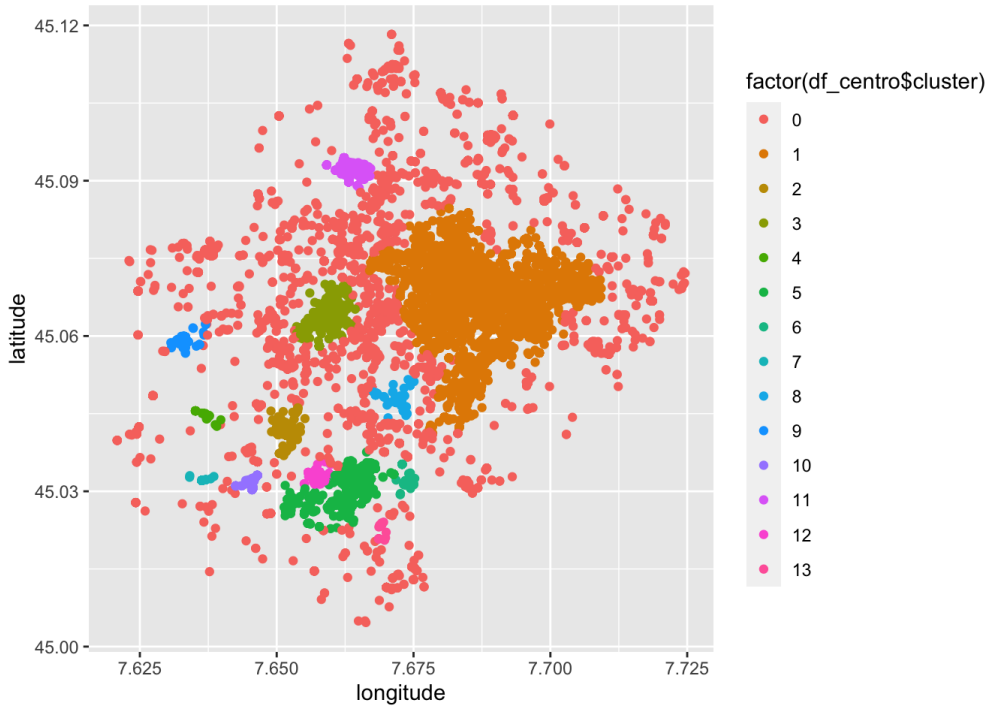
```
res <- dbscan(df_centro[1:2], eps = 0.0025, minPts = 100)
res
```

```
## DBSCAN clustering for 23603 objects.
## Parameters: eps = 0.0025, minPts = 100
## The clustering contains 13 cluster(s) and 2984 noise points.
##
##     0     1     2     3     4     5     6     7     8     9    10    11    12
##  2984 15550   602   655   114  1577   155  1063   157   134   141   260   110
##    13
##   101
##
## Available fields: cluster, eps, minPts
```

Graficamos clústers

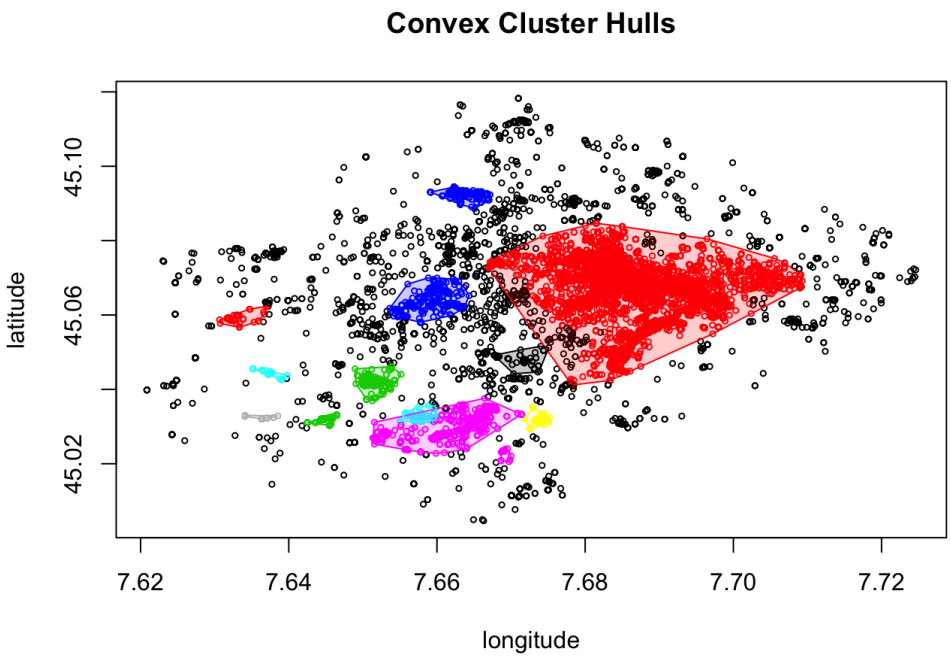
```
df_centro$cluster <- res$cluster
ggplot(df_centro, aes(x=longitude, y=latitude, colour=factor(df_centro$cluster))) +
  geom_point()
```

```
## Warning: Use of `df_centro$cluster` is discouraged. Use `cluster` instead.
```




```
hullplot(df_centro[1:2], res)
```

```
## Warning in hullplot(df_centro[1:2], res): Not enough colors. Some colors will be
## reused.
```



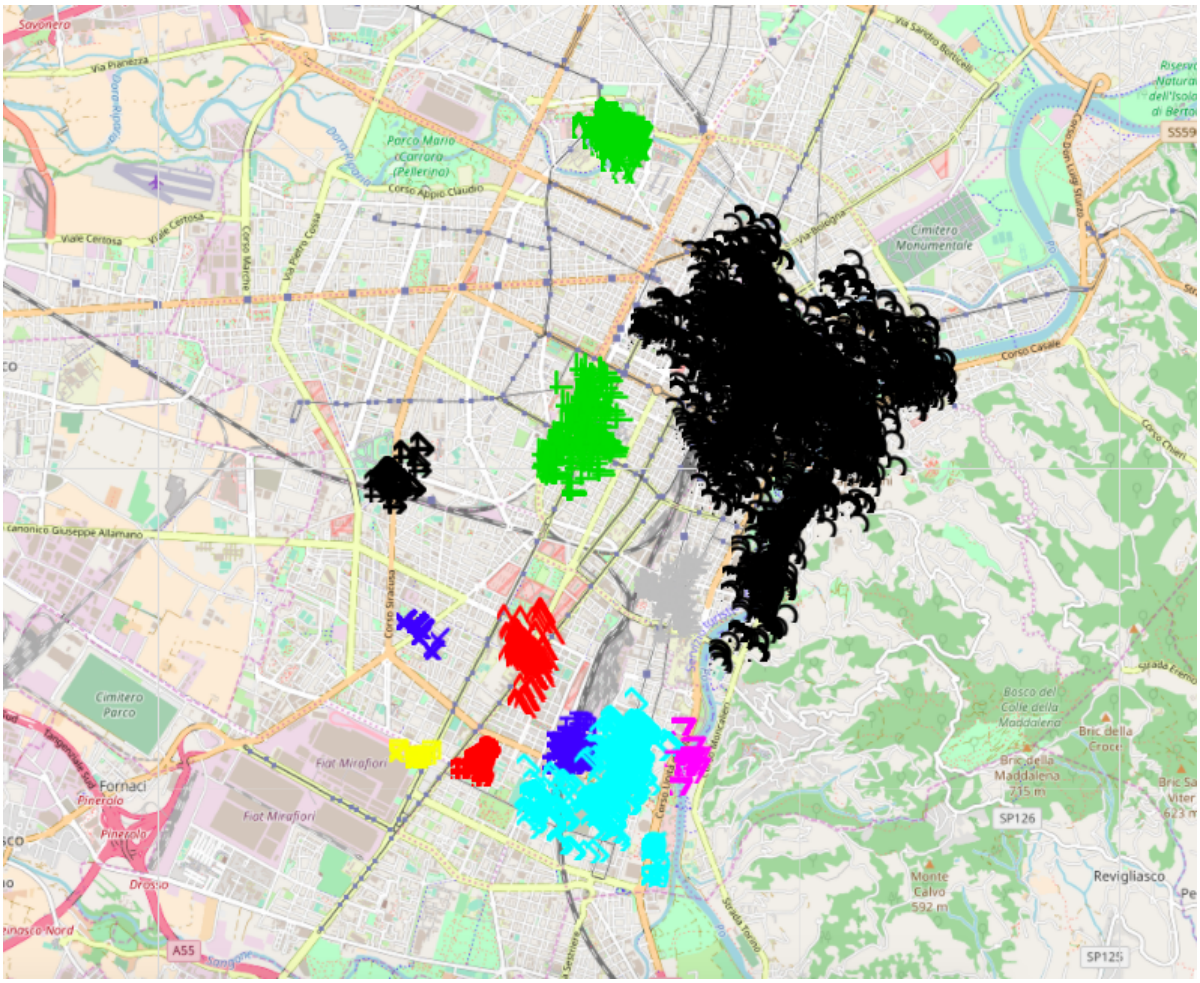
Graficamos los puntos de cada clúster en un mapa:

```
library(leaflet)

pchIcons = function(pch = 1, width = 30, height = 30, bg = "transparent", col = NULL, ...) {
  n = length(pch)
  files = character(n)
  # create a sequence of png images
  for (i in seq_len(n)) {
    f = tempfile(fileext = '.png')
    png(f, width = width, height = height, bg = bg)
    par(mar = c(0, 0, 0, 0))
    plot.new()
    points(.5, .5, pch = pch[i], col = col[i], cex = min(width, height) / 8, ...)
    dev.off()
    files[i] = f
  }
  files
}

leaflet(df_centro)%>% addTiles() %>%
  addMarkers(
    data = df_centro,
    icon = ~ icons(
      iconUrl = pchIcons(pch= cluster,width=40,height=40,col=cluster,lwd=4),
      popupAnchorX = 20, popupAnchorY = 0
    ))
```

(*Para la presentación estamos utiliando una imagen del mapa obtenido. Si se habilita el código anterior puede desplegarse el mismo.*)

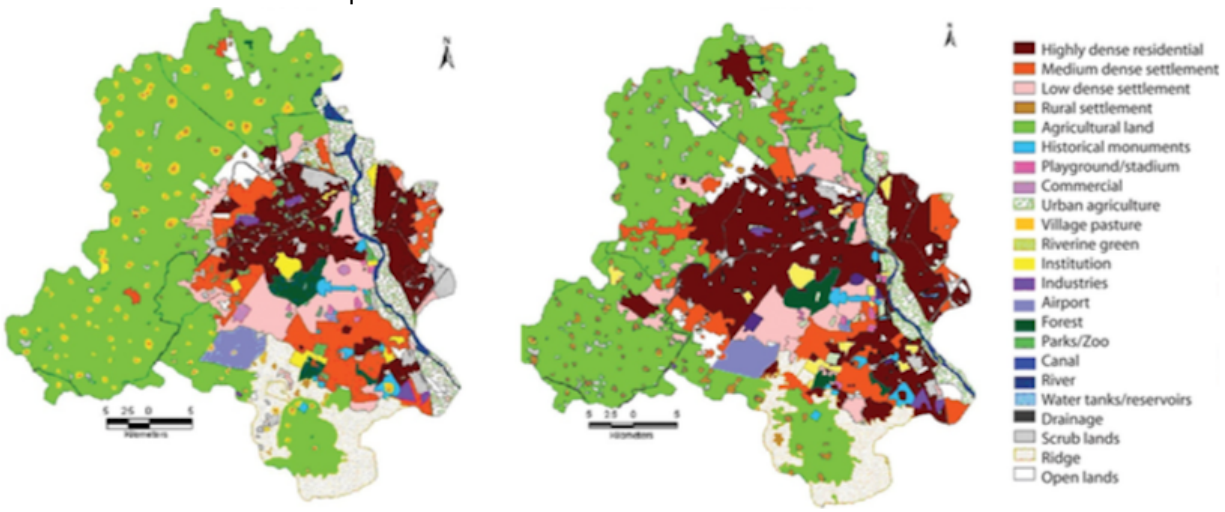


Al eliminar el ruido podemos identificar los sitios de interes, los cuales pueden coincidir con: parques, estadios, estaciones de trenes, universidades, sitios turísticos, fábricas, etc.

.- Aplicaciones:

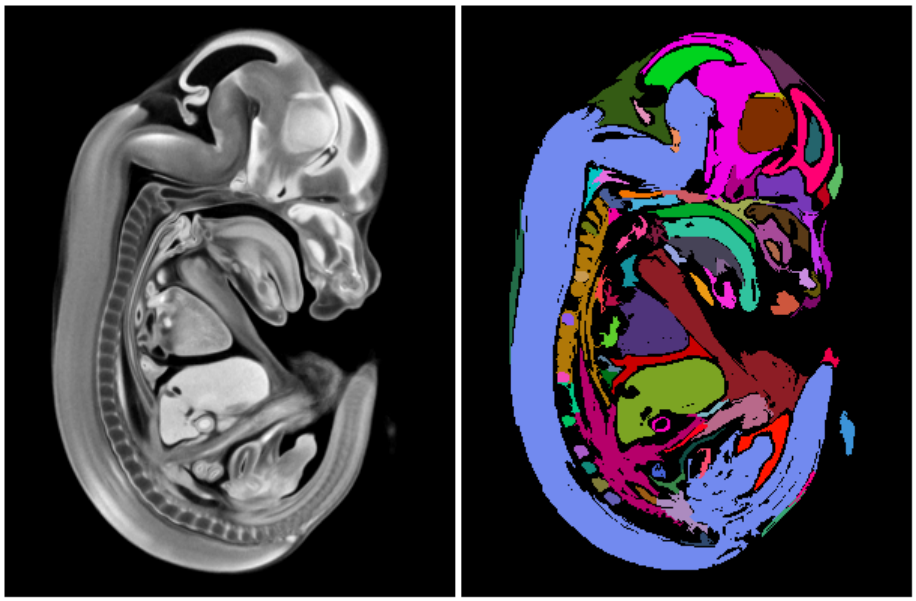
Algunas de las aplicaciones de DBSCAN realizadas con exito son: la deteccion de usos en las tierras a partir de imagenes satelite, la creacion de perfiles de usuarios en Internet mediante la agrupacion de sesiones Web, o el agrupamiento de bases de datos de imagenes en histogramas en color facilitando la busqueda de imagenes similares

Detección del uso del terreno para distintos años:



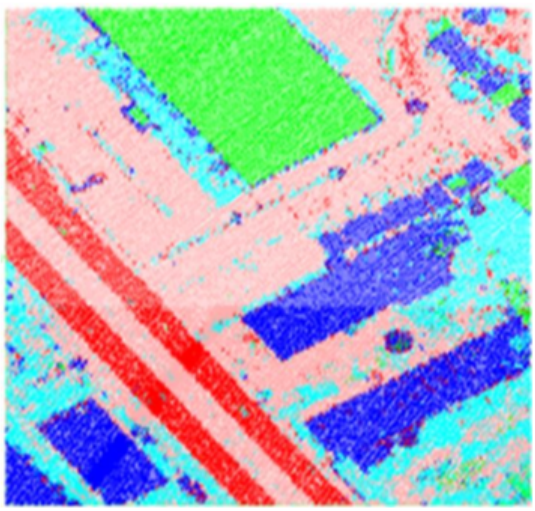
Segmentación de imagenes:

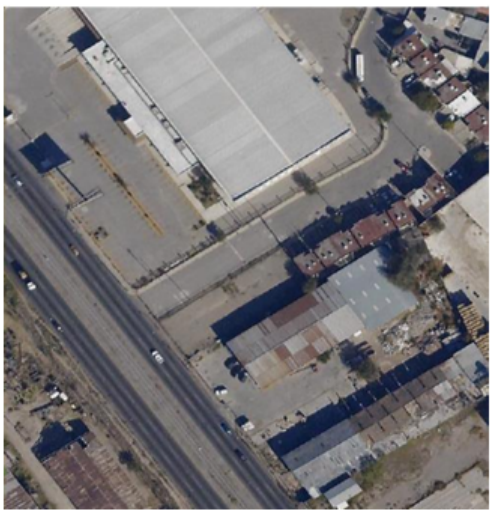
Secciones de un embrión:



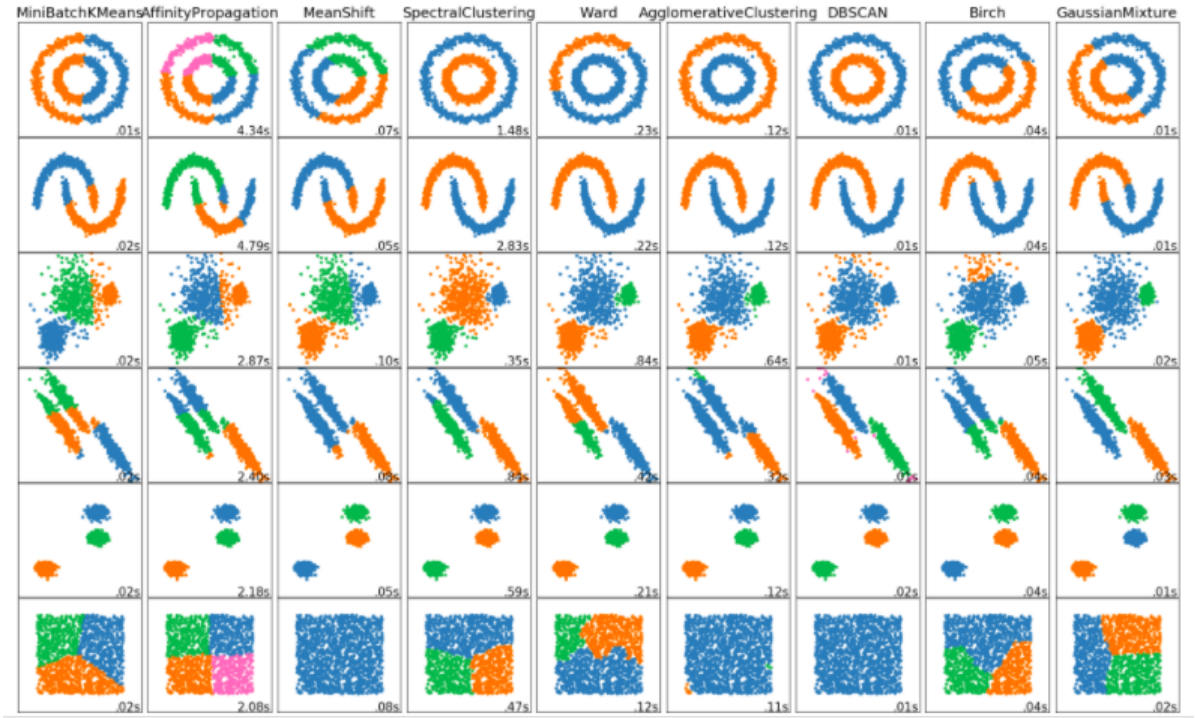
Imagenes satelitales:

Por ej. permite indentificar el porcentaje de concreto en una sección de la ciudad, así mismo también se puede realizar una aproximación sobre la cantidad de casas con techo de laminado o el porcentaje de calles pavimentadas, etc.





Clustering con diferentes algoritmos:



.- Librerías:

- Python: sklearn.cluster DBSCAN
- R: dbscan

.- Otros algoritmos:

- OPTICS: Ordering Points To Identify the Clustering Structure (Ankerst et al. SIGMOD'1999)
- DENCLUE: DENsity-based CLUstEring (Hinneburg & Keim, KDD'1998)
- CLIQUE: Clustering in QUEst (Agrawal et al., SIGMOD'1998)
- SNN (Shared Nearest Neighbor) density-based clustering (Ertöz, Steinbach & Kumar, SDM'2003)