

NIDO DEL BÚHO PRESENTA:

---

# Estadística y Econometría Espacial con R, Módulo I

Clase 3: Tipos de Datos Vectoriales



**PRESENTADO POR**

Alex Bajaña | Pablo Sarango

# Shapefile

Los datos geoespaciales en formato vectorial suelen almacenarse en formato [shapefile](#).

## Obligatorios

- [.shp](#) datos geométricos.
- [.shx](#) índice posicional de los datos geométricos.
- [.dbf](#) almacena atributos de cada *shape*.

## Opcionales

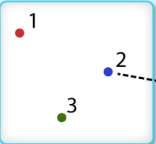
- [.prj](#) texto plano que describe la *proyección*.
- [.sbn](#) y [.sbx](#) índices espaciales.



# Atributos de los datos vectoriales

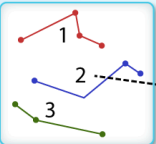
- Son todos los **datos descriptivos** asociados a **entidades geográficas**.
- Los atributos de un shapefile son similares a los campos o columnas de una hoja de cálculo.

Example Attributes for Point Data



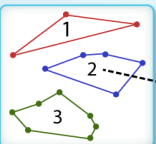
ID	Plot Size	Type	VegClass
1	40	Vegetation	Conifer
2	20	Vegetation	Deciduous
3	40	Vegetation	Conifer

Example Attributes for Line Data



ID	Type	Status	Maintenance
1	Road	Open	Year Round
2	Dirt Trail	Open	Summer
3	Road	Closed	Year Round

Example Attributes for Polygon Data



ID	Type	Class	Status
1	Herbaceous	Grassland	Protected
2	Herbaceous	Pasture	Open
3	Herbaceous / Woody	Grassland	Protected

10077

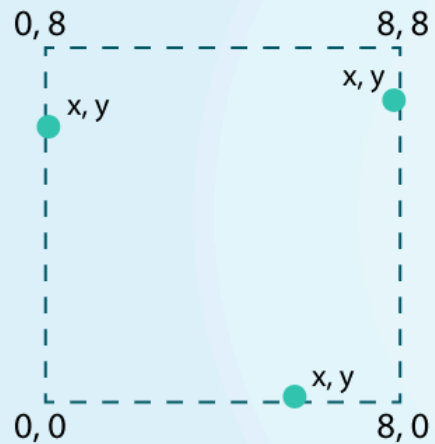
# Metadatos de los datos vectoriales

Son datos acerca de los datos.

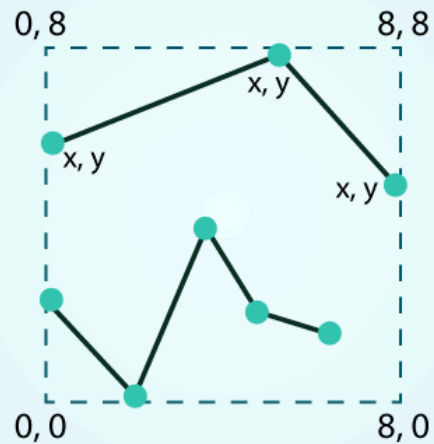
Los metadatos clave para todos los shapefiles incluyen:

- **Tipo de objeto:** la clase del objeto importado.
- **Sistema de referencia de coordenadas (CRS):** la proyección de los datos.
- **Extensión:** el área geográfica que cubre el archivo shape.

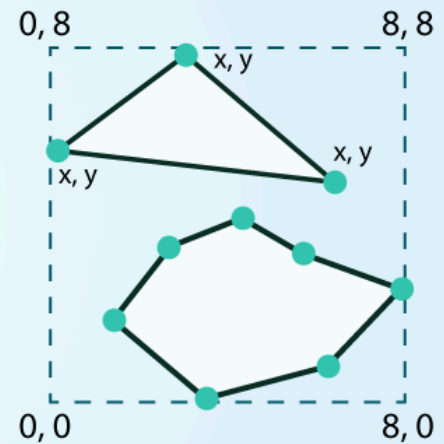
### POINTS EXTENT



### LINES EXTENT



### POLYGONS EXTENT



neon

# GeoTIFF

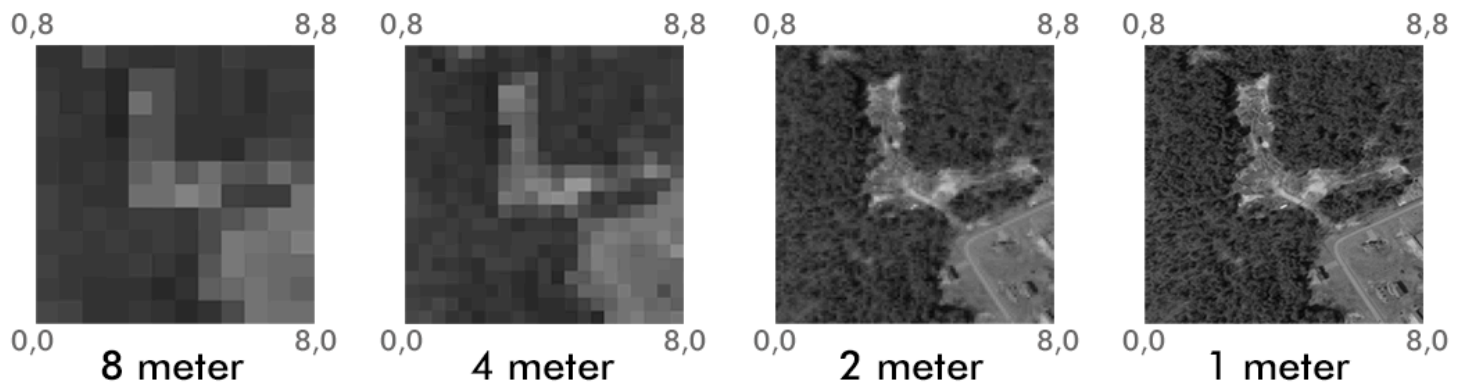
- Los datos ráster suelen presentarse en formato **GeoTIFF**.
- Es una imagen con **referencia espacial** relacionada con el mundo real.
- Su extensión es **.tif**.

# Metadatos de los datos raster

Los metadatos clave para los raster incluyen:

- **Sistema de referencia de coordenadas (CRS):** la proyección de los datos.
- **Extensión:** el área geográfica que cubre el raster.
- **Dimensión:** el tamaño del archivo en píxeles.
- **Resolución:** el tamaño de cada píxel.

Raster over the same extent, at 4 different resolutions



# Sistema de referencia de coordenadas (CRS)

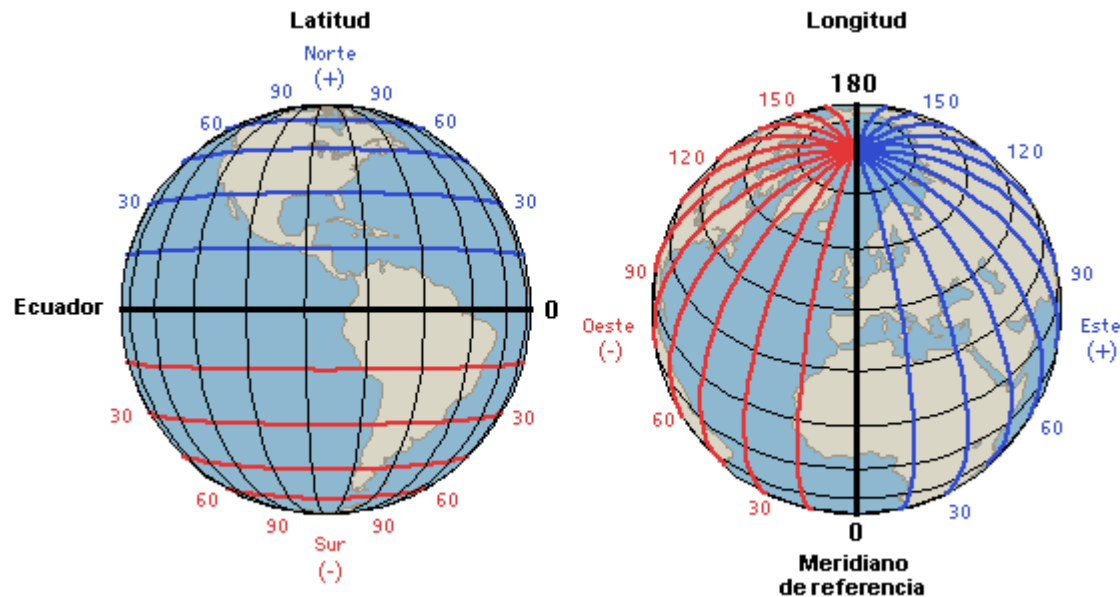
- El CRS de los datos espaciales especifica el **origen** y la **unidad de medida** de las coordenadas espaciales.
- Las localizaciones en la Tierra pueden referenciarse utilizando CRS no proyectadas (también llamadas geográficas) o proyectadas.





## CRS geográfico

- Un **CRS geográfico** utiliza *latitud* y *longitud* para representar ubicaciones en la superficie elipsoidal tridimensional de la Tierra.
- **Latitud**: mide ángulos al norte o al sur del ecuador  $0^\circ$  oscilando entre  $-90^\circ$  S hasta  $90^\circ$  N.
- **Longitud**: mide los ángulos al oeste o al este del primer meridiano oscilan entre  $-180^\circ$  W hasta los  $180^\circ$  E.



# CRS proyectado

- Un **CRS proyectado** utiliza *coordenadas cartesianas* para referenciar una ubicación en una representación bidimensional de la Tierra.
- Todas las **proyecciones** producen alguna **distorsión** de la superficie terrestre y no pueden conservar simultáneamente todas las propiedades de **área, dirección, forma y distancia**.



# Los mapas que han visto están mal

Ejemplo revelador



# Proyección UTM

- Es una **proyección cilíndrica** se proyecta el globo terráqueo sobre una superficie cilíndrica.
- Es una **proyección transversa** el eje del cilindro es coincidente con el eje ecuatorial.
- Mantiene el valor de los **ángulos**.
- Cada **zona UTM**, expresada por un **número de huso (1-60)** y una **letra de zona (C-X)** se descompone en **regiones rectangulares de 100 km de lado**.
- Los valores de las coordenadas **UTM (X e Y)** son **siempre positivos**; los ejes cartesianos X e Y se establecen sobre el huso, siendo el eje X el ecuador y el eje Y el meridiano.





# ¿Cómo se ve un lugar en distintas coordenadas?

## Grados decimales

- Latitud: `-0.2298500`
- Longitud: `-78.5249500`

## UTM

- Zona: `17S`
- X: `775495.16544253 E`
- Y: `9974570.7854669 N`



## Códigos EPSG

- El **código EPSG** corresponde a las siglas, en inglés, de **European Petroleum Survey Group**, organización científica relacionada a la industria petrolera.
- Elaboró una base de datos que contiene información a nivel mundial sobre los sistemas de referencia de coordenadas (nombre, tipo, código), proyecciones cartográficas, entre otros.

*European Petroleum Survey Group*



# Unión de tablas





# ¿Qué son los `join` y por qué son importantes?

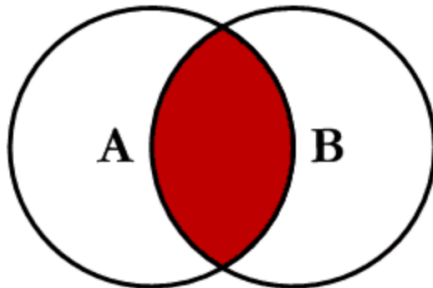
Generalmente necesitamos combinar información de distintas tablas.

Obtener datos específicos de distintas tablas, y presentar la información de una mejor manera, resulta de ayuda para la toma de decisiones.

Entonces, la función `join()` nos permite asociar 2 o más tablas, **en base a una columna que tengan en común.**

## Función: inner\_join()

Mantiene **solo** los registros comunes de ambas bases. Esta función busca coincidencias entre 2 tablas, en función a una o más columnas que tienen en común. De tal modo que sólo la **intersección** se mostrará en los resultados.

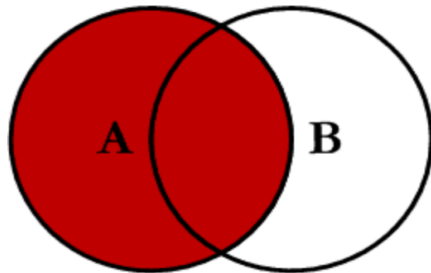


```
inner_join(x = tabla_personas,  
           y = tabla_vivienda,  
           by = "id_hogar")
```

```
# 0 con la pipe:  
tabla_personas %>% inner_join(tabla_vivienda)
```

## Función: `left_join()`

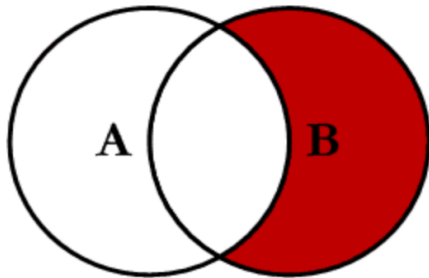
Mantiene los registros de la base de la izquierda y rellena con **Na**s los registros que no están en la base de la derecha. A diferencia de un `inner_join()`, donde se busca una intersección respetada por ambas tablas, con `left_join()` se da prioridad a la tabla de la **izquierda**, y se busca en la tabla derecha. Si no existe ninguna coincidencia para alguna de las filas de la tabla de la izquierda, de igual forma todos los resultados de la primera tabla se muestran.



```
left_join(x = tabla_personas,  
          y = tabla_vivienda,  
          by = "id_hogar")
```

## Función: `right_join()`

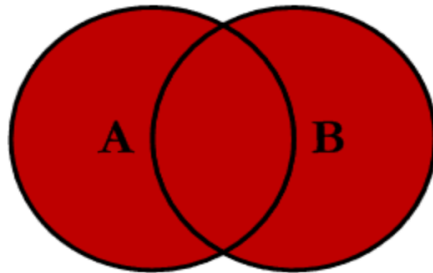
En el caso de `right_join()` la situación es muy similar, pero aquí se da prioridad a la tabla de la derecha. Es decir, mantiene los registros de la base de la derecha y rellena con **NAs** los registros que no están en la base de la izquierda. De tal modo que si usamos dicha función, estaremos mostrando todas las filas de la tabla de la derecha.



```
right_join(x = tabla_personas,  
           y = tabla_vivienda,  
           by = "id_hogar")
```

## Función: `full_join()`

Mientras que `left_join()` muestra todas las filas de la tabla **izquierda**, y `right_join()` muestra todas las correspondientes a la tabla **derecha**, `full_join()` se encarga de mostrar **todas las filas** de ambas tablas, sin importar que no existan coincidencias (usará `NULL` como un valor por defecto para dichos casos). Es decir, mantiene todos los registros en ambas bases (coincidencias y no coincidencias).



```
full_join(x = tabla_personas,  
          y = tabla_vivienda,  
          by = "id_hogar")
```

# Simulamos una base de datos

```
library(tidyverse)
library(gt)

set.seed(1984)

datos_sim <- tibble(
  id = sample(str_c("ID-",1:100),1000,replace = T),
  anio = sample(1936:1970,1000,replace = T),
  mes = sample(1:12,1000,replace = T),
  dia = sample(1:31,1000,replace = T),
  sexo = sample(c("Mujer","Hombre"),1000,replace = T),
  g_edad = sample(c("Joven","Adulto","Anciano"),1000,replace = T),
  valor = rpois(1000,300),
  ahorro = rpois(1000,100)
)
```

En ocasiones sera importante simular datos, para este ejercicio haremos un ejercicio de compras y ahorro diario para 100 personas distintas durante los años 1936 y 1970. Los valores de compras vienen dados por una distribución Poisson con lambda 300 y el ahorro con un lambda de 100.

# Diccionario de variables:

Variables	Descripción
id	Identificador de la persona
anio	Año de la compra
mes	Mes de la compra
dia	Día de la compra
sexo	Sexo del comprador
g_edad	Grupo etario
valor	Valor de la compra
ahorro	Ahorro

**Como funciona ggplot2**



# Argumentos

Tabla:

```
tabla %>%  
  ggplot() + geom_{\{\{tipo\}\}}(\{\{aes\}\})
```

Mapping:

```
tabla %>% ggplot() +  
  geom_{\{\{tipo\}\}}(  
    \{\{aes\}\})
```

Aesthetic:

```
tabla %>% ggplot() + geom_{\{\{tipo\}\}}(  
  \{\{aes\}\}  
)
```

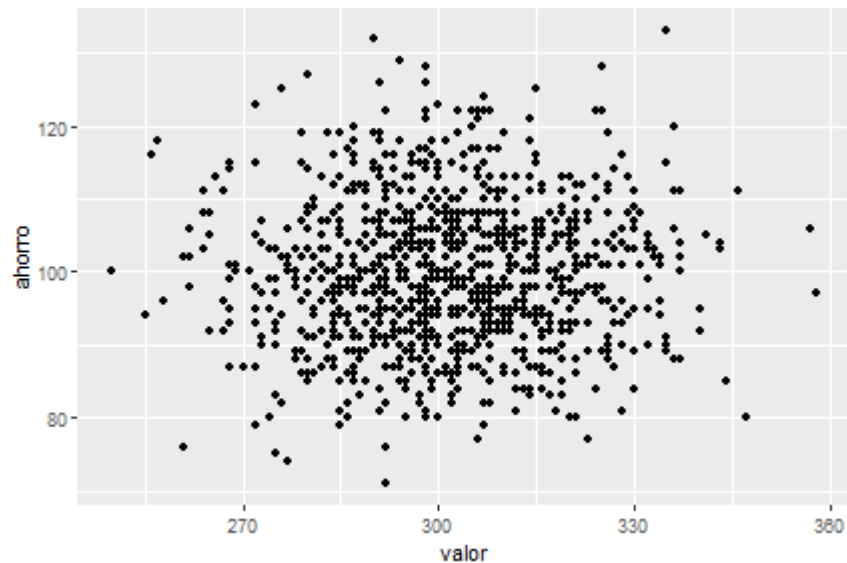
...: Otros argumentos:

```
tabla %>% ggplot() + geom_{\{\{tipo\}\}}( \{\{aes\}\},  
  ...)
```

# Nube de puntos

Grafico básico:

```
ggplot(data = datos_sim) +  
  geom_point(  
    mapping = aes(x = valor,  
                  y = ahorro)  
  )  
)
```

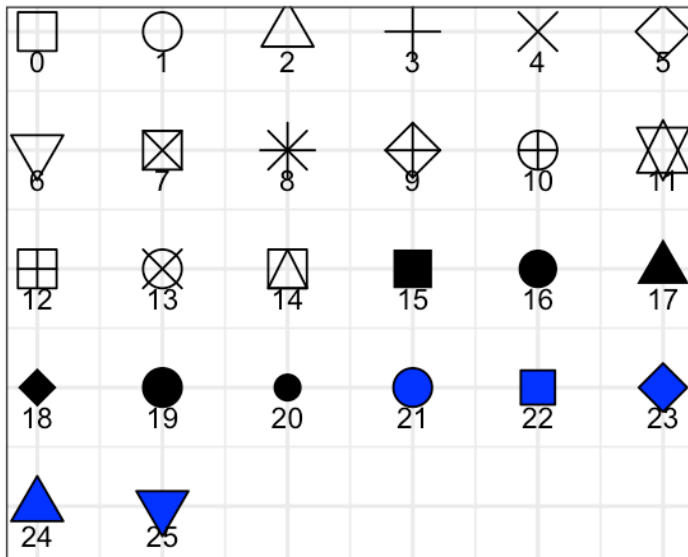


# Personalización

Añadir **color**, **size**, **alpha**, y **shape** en el aesthetic para hacer más bonito el gráfico.

## Shape:

Point shapes available in R



- **size**: Puede ser:
  - Un valor entero si deseo que todos los puntos tengan el mismo tamaño
  - Una variable numérica que determine el tamaño del punto
- **color**: Puede ser:
  - Un string con el nombre del color (en inglés)
  - Colores HTML (Ejemplo: '#d7215b')
- **alpha**: Valor entre 0 (transparente) y 1 (solido)

## Dando formato al gráfico

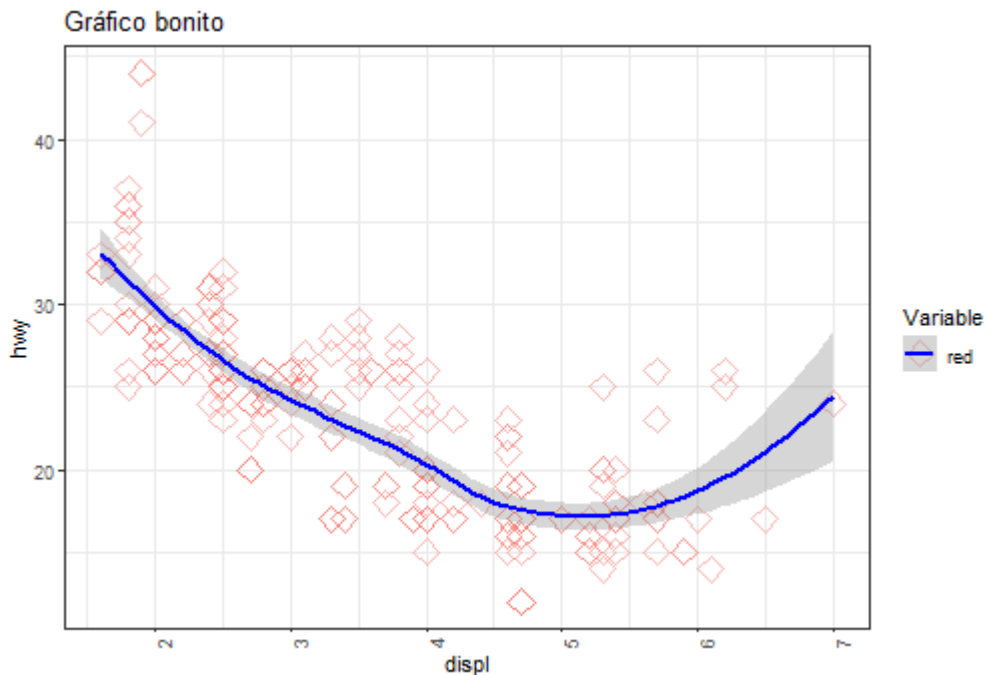
```
plot <- mpg %>%  
  ggplot() +  
  geom_point(aes(x = displ,  
                 y = hwy,  
                 color = "red"),  
             size = 4,  
             shape = 5,  
             alpha = 0.5,  
             show.legend = T) +  
  geom_smooth(  
    aes(x = displ,  
        y = hwy),  
    color = "blue",  
    show.legend = T) +  
  labs(title = "Gráfico bonito",  
        caption = "Elaborado por: Eric Blair",  
        color = "Variable") +  
  theme_bw()+  
  theme(axis.text.x = element_text(angle = 90))
```

# Gráfico personalizado

```
plot
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
## Warning in vp$just: encuentros parciales de 'just' to 'justification'
```



**Gracias por la atención**