

# Visualización y geolocalización de datos

Diego Hernangómez      Gema Fernández-Avilés

2022-02-02

# Índice general

<b>Prólogo</b>	<b>3</b>
¿Por qué leer este libro? . . . . .	3
Objetivos de aprendizaje . . . . .	3
Estructura del libro . . . . .	4
Datos . . . . .	4
<b>1. La revolución de los geodatos</b>	<b>5</b>
<b>2. Datos geográficos</b>	<b>7</b>
2.1. Contexto general . . . . .	7
2.2. Conceptos clave . . . . .	10
<b>3. Formatos de datos espaciales</b>	<b>12</b>
3.1. Tipos de ficheros (COMPLETAR) . . . . .	12
3.2. Datos de vectores . . . . .	12
3.3. Datos raster . . . . .	14
3.4. Sistema de Referencia de Coordenadas (CRS) . . . . .	17
<b>4. Estadística espacial</b>	<b>35</b>
4.1. Antes de continuar... dependencia espacial. . . . .	36
4.2. Datos espaciales . . . . .	37
4.3. Clasificación de datos espaciales . . . . .	40

<b>5. Casos prácticos</b>	<b>44</b>
5.1. Caso 1: Temperatura mínimas del aire en España. . . . .	44
5.2. Caso 2. Distribución espacial de la renta media por municipios . . .	58
5.3. Caso 3. Distribución espacial de los delitos cometidos en la ciudad de Valencia en el año 2010. . . . .	74
<b>A. Tipos de CRS proyectados</b>	<b>85</b>
A.1. Por tipo de superficie de proyección . . . . .	85
A.2. Por métrica a preservar . . . . .	87
<b>B. Referencias</b>	<b>90</b>



# Prólogo

Este libro presenta los conceptos teóricos y prácticos necesarios para llevar a cabo correctamente la de y su  
Poner un poco de rollo

## ¿Por qué leer este libro?

Los están presentes en la mayoría de los fenómenos que nos rodean, es un hecho que no podemos obviar y que aporta información muy valiosa en la mayoría de los análisis estadísticos llevados a cabo en distintas disciplinas tales como la Sociología, la Inteligencia Artificial, la Economía, la Antropología, la Biología, etc...

La representación gráfica de los mismos adquiere un papel crucial, ya que (i) se detectan relaciones y patrones entre las variables más fácilmente, (ii) hacen que la distribución del fenómeno se comprenda y recuerde rápidamente, (iii) grandes medidas de datos estadísticos se puede visualizar de un vistazo, (iv) permiten sacar a la luz hechos y relaciones ocultos que estimulen el pensamiento analítico y la investigación.

La principal contribución del libro es mostrar el proceso de visualización de datos georreferenciados de principio a fin. \*\*\*\*\*

**rematar...**

## Objetivos de aprendizaje

Poner algo

## Estructura del libro

El Capítulo ?? introduces a new topic, and ...

## Datos

DIEGO, YO NO SE SI DEBEMOS DEJAR TODOS, SEGÚN VEAMOS,

- temperatura mínima en el aire en España
- renta
- delitos valencia
- porcentaje de mujeres en clm
- tráfico madrid
- Hospitales en toledo
- Rio tajo ??
- Elevación en España
- Puertos

contar un poco al final PARA decir que tenemos una amplia variedad de aplicaciones para ilustrar los conceptos bla bla

**ojo, ver cómo entra esto en el libro o no**

Libros de referencia:

- Spatial Data Science with applications in R
- Geocomputation with R
- Displaying time series, spatial and space-time data with R

Recursos de estadística espacial en R:

- r spatial
- R-spatial

# Capítulo 1

## La revolución de los geodatos

Que estamos en la la *era del dato*, que los *datos son el petróleo del siglo XXI* y que estamos rodeados de datos es una cuestión que ya hemos hecho inherente a nosotros. Vivimos en el momento del dato, donde la profesión de *Data Scientist* se ha convertido en la **profesión más sexy del siglo XXI** según vaticinó en 2012 Harvard Business Review. Cada segundo se producen 1,7 MB de datos/persona y cada año esta cifra se duplica se duplica.

Este incremento exponencial de los datos ha sido posible, sin duda, gracias al desarrollo de la tecnología, la informática, los ordenadores, los teléfonos móviles, los satélites, internet, etc... y asociado a estas nuevas herramientas, se ha producido una lluvia sin precedentes hasta el momento de **datos espaciales o datos georreferenciados**. Cada teléfono inteligente tiene un **Receptor de posicionamiento Global** (en inglés, *Global Positioning System, GPS*) y, además, convivimos con una multitud de sensores en dispositivos que van desde satélites y vehículos semi-autónomos hasta científicos ciudadanos que miden incessantemente cada parte del mundo. La tasa de datos producidos es abrumadora. Un vehículo autónomo, por ejemplo, puede generar 100 GB de datos por día (The Economist, 2016).

Esta **revolución de los geodatos** y el **análisis de los datos espaciales** junto con los **Sistemas de Información Geográficos** (habitualmente expresados como **GIS** por las siglas de su nombre en inglés *Geographical Information System*) no sólo han impulsado la demanda de hardware informático de alto rendimiento y software escalable y eficiente para manejar y extraer la información, lo que se conoce como **Geocomputación**, sino que han dado lugar una nueva rama de conocimiento, **Ciencia de Datos Espaciales** comúnmente conocida como *Spatial Data Science* (SDS).

Como ejemplo de este abrumador desarrollo de datos georreferenciados, el Ministerio de Transportes, Movilidad y Agenda Urbana llevó a cabo durante los años 2020 y 2021 el denominado Estudio de movilidad con Big Data, cuya fuente principal de datos fue el posicionamiento de los teléfonos móviles anonimizado. Estos datos permiten, por ejemplo, analizar la movilidad entre diversas zonas del territorio español de manera diaria (véase la Fig. 1.1). Este tipo de análisis era impensable hace tan solo unos años.

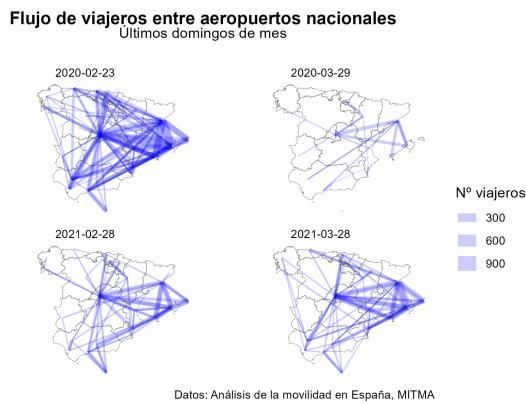


Figura 1.1: Análisis de movilidad COVID

# Capítulo 2

## Datos geográficos

### 2.1. Contexto general

La palabra geográfico puede dividirse en **geo** (tierra) + **gráfico** (dibujo/mapa). Por tanto, los datos geográficos contienen información de cualquier variable referenciada en un punto/área de la superficie terrestre y pueden representarse en mapas. El desarrollo de los datos geográficos ha producido grandes bases de datos espaciales y, a su vez, ha propiciado el desarrollo de herramientas para su tratamiento como los ya mencionados Sistemas de información geográficos y la Geocomputación.

#### ¿Qué hace un Sistemas de información geográfico?

Un Sistema de información geográfica (SIG) es una herramienta que crea, administra, analiza y mapea todo tipo de datos. GIS conecta datos a un mapa, integrando datos de ubicación (**dónde** están las cosas) con todo tipo de información descriptiva (**cómo** son las cosas allí).

Esto proporciona una base para el mapeo y el análisis que se utiliza en la ciencia y en casi todas las industrias. GIS ayuda a los usuarios a comprender patrones, relaciones y contexto geográfico. Los beneficios incluyen una mejor comunicación y eficiencia, así como una mejor gestión y toma de decisiones.

La Fig. 2.1 muestra el flujo de trabajo de los SIG, que va desde (i) la elaboración de mapas, (ii) la obtención de geodatos o datos espaciales, (iii) el análisis de los datos geográficamente referenciados y (iv) la edición, mapeo y presentación de los resultados.

Pero es más, el desarrollo de la **Inteligencia Artificial** y la **Inteligencia computacional** se han convertido en herramientas creativas y complementarias a los

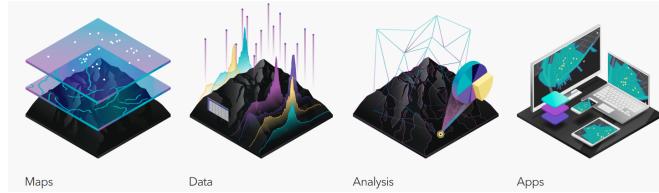


Figura 2.1: Flujo de trabajo de los GIS. Fuente: <https://www.esri.com/en-us/what-is-gis/overview>

convencionales GIS, dando origen a la *Geocomputación*, que trata de utilizar el *poder de los ordenadores para hacer cosas con los datos geográficos*.

### ¿Y qué es la Geocomputación?

En primer lugar, señalar que, aunque la geocomputación es un término relativamente nuevo se encuentra influenciado por otros términos clásicos. De manera sencilla puede definirse como “*el proceso de aplicar tecnologías de computación a problemas geográficos*” [Rees, 1998]. Abrahart et al. [2000] aporta más elementos formales a esta definición destacando que “*la geocomputación trata sobre los diferentes tipos de geodatos, y sobre el desarrollo de geo-herramientas relevantes en un contexto científico*”.

La geocomputación está muy relacionada con otros términos como los SIG, ya definidos, y con diversos tipos de campos científicos, como las Geociencias, las Ciencias atmosféricas y climáticas, la Geoinformática, la Topología, la Ecología y las Ciencia de datos geográficos (GDS, Geographic Data Science).

Cada término comparte un énfasis en un enfoque **científico** (que implica reproducible y falsable) influenciado por los GIS, aunque sus orígenes y principales campos de aplicación difieren. La geocomputación es ampliamente utilizada en ámbitos como la sociología, el análisis político o el desarrollo de aplicaciones para móviles. Por tanto, usamos geocomputación como un sinónimo aproximado que encapsula a todas las ciencias que buscan usar datos geográficos para trabajos científicos aplicados.

### ¿Por qué R para datos geográficos?

R es una herramienta con capacidades avanzadas de análisis, modelado y visualización. Por ejemplo, los nuevos entornos de desarrollo integrado (en inglés, Integrated Development Environment, **IDE**), como RStudio, han hecho que R sea más fácil de usar para muchos, facilitando la creación de mapas con un panel dedicado a la visualización interactiva [Lovelace et al., 2019]. Además, el uso del código R permite la enseñanza de la geocomputación con referencia a ejemplos reproducibles en lugar de conceptos abstractos. Por ejemplo, de una forma relativamente sencilla,

se puede geoposicionar de manera interactiva la localización de la Puerta del Sol en Madrid y, además, dejar la el código R para hacerlo reproducible, ver Fig. 2.2.

```
library(leaflet)
leaflet(width = "100%", height = "500px") %>%
  addTiles() %>%
  setView(-3.703548, 40.417147, zoom = 16)
```



Figura 2.2: Localización interactiva de la Puerta del Sol en Madrid

Por otra parte R dispone de cientos de librerías especializadas para datos espaciales. Una descripción detallada puede ver se en CRAN Task View: Analysis of Spatial Data

Para no abrumar al lector, a continuación se muestran, de manera esquemática, las librerías más usadas para el tratamiento de datos espaciales y que se emplearán a lo largo de la asignatura Estadística Espacial y Espacio-Temporal, no sólo en el tema que nos ocupa:

- **sp** y **sf**: para el tratamiento de clases y métodos de los datos vectoriales.

- `raster`, `terra` y `stars` para datos raster.
- `gstat` y `geoR`: para el análisis de datos geoestadísticos, ajuste y estimación de semivariogramas, interpolación, etc.
- `spdep` para el análisis de datos con modelos de econometría espacial, creación de matrices de contigüidad/distancia  $\mathbf{W}$ , estimación de modelos económicos espaciales, etc
- `spatstat` para el análisis de procesos de puntos espaciales, intensidad, etc.

## 2.2. Conceptos clave

Una vez visto el contexto actual de los datos georreferenciados y antes de entrar en detalle en su análisis, debemos tener en cuenta una serie de conceptos clave que se irán desarrollando a lo largo del tema.

Hemos dicho que Geográfico = Geo (tierra) + gráfico (mapa). Por tanto, si tenemos varios datos geográficos, localizados en distintos puntos de la tierra, es porque tenemos las **coordenadas** que los posicionan en esos puntos concretos. Asociado a estas coordenadas debemos conocer el **Sistema de referencia de espacial** o Coordinate reference system (CRS) en el que están proyectadas dichas coordenadas.

Por otra parte, los formatos de estos datos pueden ser **vectores** o **raster** como se explicará en la Sección 3.

Si damos un paso más e incorporamos el concepto de **distancia**, pues es lógico pensar que en un fenómeno de interés, por ejemplo, la modelización de la cantidad y dirección de lava en La Palma tras la erupción del volcán “Cumbre Vieja”, la distancia es un factor clave, pues aquellas zonas más cercanas al volcán tendrán niveles más parecidos entre sí y con valores más altos que aquellas que están más alejadas.

En este caso el nivel de contaminación en el aire en La Palma no puede ser modelado como si las observaciones fuesen independientes pues las más cercanas entre sí serán más parecidas que las más lejanas, dando lugar al concepto de **dependencia espacial**. Y depende del tipo de datos espaciales tendremos tres grandes formas de abordar el tratamiento de los datos espaciales: **geoestadística**, **procesos de punto** y **econometría espacial** (véase sección 3.4).



Figura 2.3: Información espacial de la concentración de lava en Cumbre Vieja

# Capítulo 3

## Formatos de datos espaciales

### 3.1. Tipos de ficheros (COMPLETAR)

Poner el esquema de los Shapes.

<https://en.wikipedia.org/wiki/Shapefile>

Yo no sé, DIEGO, si en gisco R tienes la definición y lo ponemos de ahí.

Y pensar si tiene que venir aquí o dónde... quizá al final de la sección FORMATOS DE DATOS ESPACIALES

En el ámbito del análisis espacial en **R**, se pueden clasificar **el formato** de datos espaciales en función del modelo de datos [Lovelace et al., 2019]. Se pueden distinguir dos tipos de modelos de datos: vectores y raster.

### 3.2. Datos de vectores

Este modelo está basado en puntos georeferenciados. Los **puntos** pueden representar localizaciones específicas, como la localización de edificios:

```
library(ggplot2)
library(sf)

# Hospitales en Toledo segun Eurostat
```

```
hosp_toledo <- st_read("data/hosp_toledo.geojson", quiet = TRUE)

# Plot
ggplot() +
  geom_sf(
    data = hosp_toledo, aes(fill = "Centros Sanitarios"),
    color = "blue"
  ) +
  labs(
    caption = "Datos: Eurostat",
    title = "Hospitales y Centros de Salud en Toledo",
    fill = ""
  ) +
  theme_minimal() +
  theme(legend.position = "bottom")
```

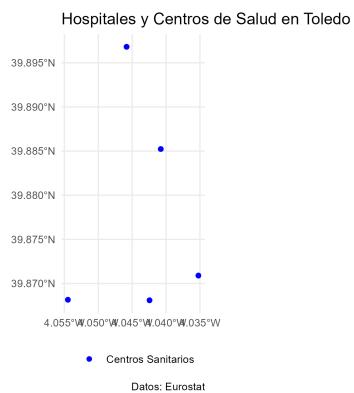


Figura 3.1: Datos vector: Puntos

Estos puntos también pueden estar conectados entre sí, de manera que formen geometrías más complejas, como **líneas y polígonos**:

```
tajo <- st_read("data/tajo_toledo.shp", quiet = TRUE)
toledo <- st_read("data/toledo_ciudad.gpkg", quiet = TRUE)

ggplot(toledo) +
  geom_sf(fill = "cornsilk2") +
```

```
geom_sf(data = tajo, col = "lightblue2", lwd = 2, alpha = 0.7) +
  geom_sf(data = hosp_toledo, col = "blue") +
  coord_sf(
    xlim = c(-4.2, -3.8),
    ylim = c(39.8, 39.95)
  ) +
  theme_minimal()
```

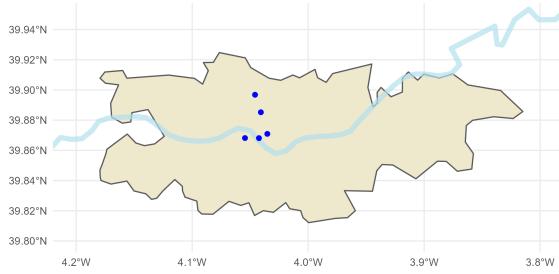


Figura 3.2: Datos vector: Puntos, líneas y polígonos

En la Fig. 3.2, el río Tajo está representado como una línea (sucesión de puntos unidos entre sí) y la ciudad de Toledo como un polígono (línea de puntos cerrada formando un continuo). A modo ilustrativo, la Fig. 3.3 representa la descomposición en puntos de todos los datos espaciales representados en la Fig. 3.2.

### 3.3. Datos raster

Los datos ráster son datos representados en una rejilla rectangular de píxeles (denominada **matriz**) que se puede visualizar en diversos dispositivo de representación. El caso más cotidiano de un ráster es una fotografía, donde la imagen se representa como una serie de celdas, determinadas por la resolución de la imagen (número total de píxeles, determinados como número de píxeles en cada fila por número de píxeles en cada columna) y el color que presenta cada uno de estos píxeles.

En el ámbito de los datos espaciales, la definición es muy similar. Un archivo ráster está formado por una malla regular de píxeles georreferenciada, tal y como muestra la Fig. 3.4:

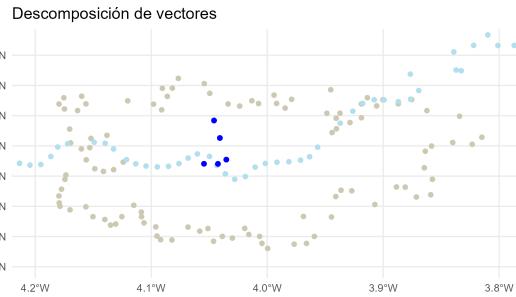


Figura 3.3: Datos vector: Descomposición en puntos

```
library(raster)

elev <- raster("data/Toledo_DEM.tif")
plot(elev, main = "Elevación de la provincia de Toledo")

# Mostramos el grid
polys <- rasterToPolygons(elev)
plot(polys, add = TRUE, border = "grey90")

# Añadimos la provincia
Tol_prov <- st_read("data/Toledo_prov.gpkg", quiet = TRUE)

# Si queremos solamente la forma en sf, usamos st_geometry
plot(st_geometry(Tol_prov), add = TRUE)
```

En la Fig. 3.4, el objeto ráster `elev` tiene únicamente una capa (denominada `ESP_alt`). Eso implica que cada píxel tiene asociado un único valor, en este caso, en este caso la altitud media del terreno observada:

Los rásters pueden contener varias capas (o layers), de manera que cada píxel puede tener asociados varios valores. Volviendo al ejemplo de la fotografía, en un modelo simple de color RGB cada píxel lleva asociado 3 valores (rojo, verde o azul), de manera que al combinar las tres capas se puede definir un color distinto en cada píxel.

En la Fig. 3.6 vamos a usar una imagen de mapa georreferenciada, como las pro-

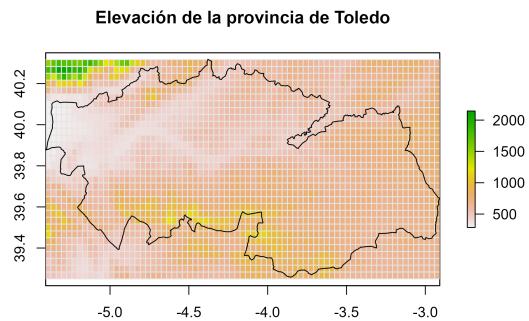


Figura 3.4: Datos ráster

Cuadro 3.1: Datos de un ráster (detalle)

x	y	Toledo_DEM
-5.391667	40.3	1498.312
-5.358333	40.3	1701.125
-5.325000	40.3	1825.312
-5.291667	40.3	1739.062
-5.258333	40.3	1756.062
-5.225000	40.3	1659.688
-5.191667	40.3	1607.375
-5.158333	40.3	1809.562
-5.125000	40.3	1874.625
-5.091667	40.3	1691.312
-5.058333	40.3	1511.500
-5.025000	40.3	1207.000
-4.991667	40.3	1160.125
-4.958333	40.3	1396.125
-4.925000	40.3	1624.125

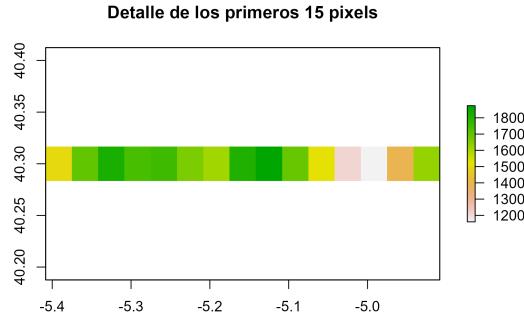


Figura 3.5: Datos ráster: Detalle

porcionadas por servicios de mapas online, para analizar su composición.



Figura 3.6: Datos ráster con varias bandas

El ráster se puede descomponer en las tres capas RGB mencionadas anteriormente:

### 3.4. Sistema de Referencia de Coordenadas (CRS)

Un sistema de referencia de coordenadas (o CRS por sus siglas en inglés, **C**oordinate **R**eference **S**ystem) permite relacionar datos espaciales con su localización en la superficie terrestre.

Cuadro 3.2: Datos de un ráster multicapa (detalle)

x	y	lyr.1	lyr.2	lyr.3
-5.466412	40.34418	215.2128	208.1061	190.5410
-5.463875	40.34418	228.0369	223.1854	211.2115
-5.461338	40.34418	229.3495	224.3414	213.4325
-5.458800	40.34418	215.8592	208.8660	191.2922
-5.456263	40.34418	219.2696	212.8231	196.6812
-5.453725	40.34418	235.0954	231.4222	222.4115
-5.451188	40.34418	240.3514	237.9094	231.4736
-5.448651	40.34418	237.2358	233.7561	226.2005
-5.446113	40.34418	229.9570	225.3262	214.6201
-5.443576	40.34418	226.7812	221.6796	209.2929
-5.441038	40.34418	222.3593	216.5022	202.0188
-5.438501	40.34418	220.9312	214.9060	200.0306
-5.435964	40.34418	224.7755	219.2661	206.2156
-5.433426	40.34418	222.0479	216.0124	201.6103
-5.430889	40.34418	225.0516	219.8074	207.0263



Figura 3.7: Datos ráster multicapa: Descomposición

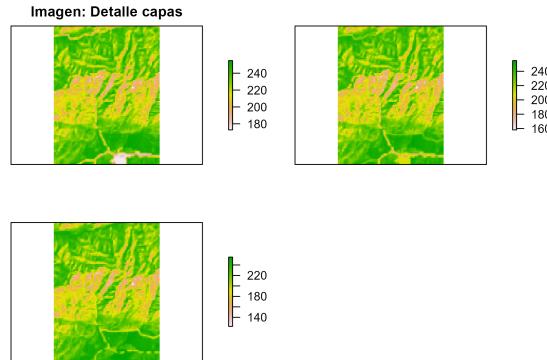


Figura 3.8: Datos ráster multicapa: Descomposición

**Los CRS constituyen por tanto un aspecto fundamental en el análisis y representación de datos espaciales**, ya que nos permiten identificar con exactitud la posición de los datos sobre el globo terráqueo.

Así mismo, cuando se trabaja con datos espaciales provenientes de distintas fuentes de información, es necesario comprobar que dichos datos se encuentran definidos en el mismo CRS:

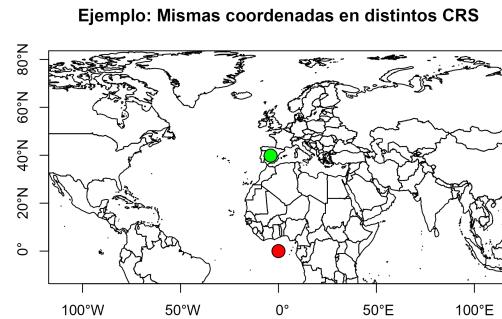


Figura 3.9: Representación de mismos valores de coordenadas en distintos CRS

En la Fig. 3.9, ambos puntos (verde y rojo) tienen los mismos valores de coordenadas en los ejes X e Y, en este caso las correspondientes a la ciudad de Toledo. Sin embargo, presentan distintos CRS. Por este motivo, al representar ambos puntos en un mapa, se observa que no se están refiriendo a la misma localización geográfica. Esto es así porque el CRS define la referencia (punto x=0 e y =0) y las

unidades de los ejes (grados, metros, millas).

Como conclusión, además de disponer de las coordenadas de los datos espaciales, es necesario conocer el CRS en el que están definidos para conocer de manera exacta su localización geográfica. Además, nótese que para cualquier análisis de datos espaciales es necesario que todos los geodatos se encuentren referenciados en el mismo CRS. Esto se consigue transformando (o proyectando) los datos a un CRS común, nunca sobreescritiendo el CRS de los mismos.

### 3.4.1. Tipos de CRS

A continuación se definen los dos grandes tipos de CRS, los CRS geográficos y los CRS proyectados.

#### 3.4.1.1. CRS geográficos

Los CRS geográficos son aquellos en los que los parámetros empleados para localizar una posición espacial son la latitud y la longitud:

- **Latitud:** Es la distancia angular expresada en grados sobre el plano definido por el ecuador terrestre. Determina la posición sobre de una localización en el eje Norte-Sur de la Tierra y toma valores en el rango  $[-90, 90]$ . Las líneas imaginarias determinadas por una sucesión de puntos con la misma latitud a lo largo del eje Este-Oeste se denominan **paralelos** (Ver Fig. 3.10).
- **Longitud:** Es la distancia angular expresada en grados sobre el plano definido por el meridiano de Greenwich. Determina la posición sobre de una localización en el eje Este-Oeste de la Tierra y toma valores en el rango  $[-180, 180]$ . Las líneas imaginarias determinadas por una sucesión de puntos con la misma longitud a lo largo del eje Este-Oeste se denominan **meridianos** (Ver Fig. 3.10).

Es muy importante destacar que en un sistema de coordenadas geográfico, es decir, basado en latitudes y longitudes, las **distancias** entre dos puntos representan **distancias angulares**. Por ejemplo, la distancia entre el meridiano de Greenwich y el meridiano correspondiente a la longitud  $20^\circ$  siempre es de  $+20^\circ$ . Sin embargo, debido a la forma esférica de la Tierra, la longitud en metros entre ambos meridianos no es constante.

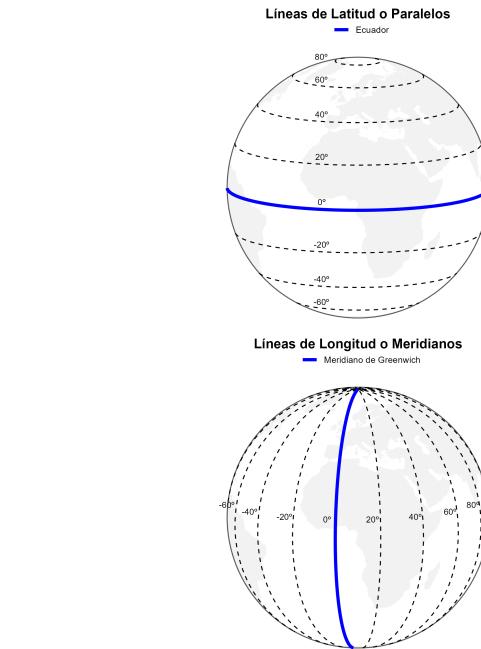


Figura 3.10: Paralelos y Meridianos terrestres

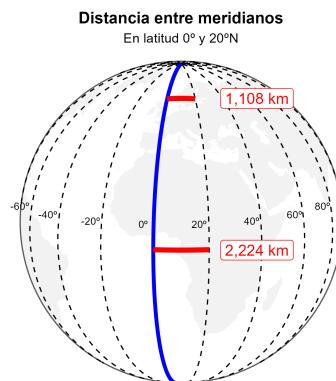


Figura 3.11: Distancia entre meridianos en distintas latitudes

### 3.4.1.2. CRS proyectados

La representación de formas tridimensionales en un soporte plano (dos dimensiones) presenta algunos retos. Por ello, es habitual trabajar con proyecciones de mapas.

Una **proyección geográfica** es un método para reducir la superficie de la esfera terrestre a un sistema cartesiano de dos dimensiones. Para ello, es necesario transformar las coordenadas longitud y latitud en coordenadas cartesianas x e y.

Es importante destacar que las proyecciones pueden incluir un punto de origen ( $X=0, Y=0$ ) y unas unidades de distancia (habitualmente metros) específicas. Por ejemplo, la **proyección cónica equiáreas de Albers** (específica para Estados Unidos) define su punto de referencia (0,0) en la latitud  $40^{\circ}$  N y longitud  $96^{\circ}$ , y la unidad de variación están definida en metros. De ahí la importancia de conocer el CRS de los datos geográficos, como se expuso al principio de este tema.

El Anexo A proporciona más información sobre los tipos de CRS proyectados.

### 3.4.2. Trabajando con proyecciones en R

Existe toda una serie de proyecciones predefinidas, identificadas mediante los **códigos EPSG, ESRI, WKT** o proj4 (en desuso en R, pero todavía admitidos). Existen varios recursos web donde se pueden consultar y seleccionar los códigos correspondientes:

- <https://epsg.io/>
- <https://spatialreference.org/>
- <https://proj.org/operations/projections/index.html>

Algunos de los códigos de proyecciones que es fundamental conocer son:

- **EPSG: 4326:** Proyección correspondiente a WGS 84, que es el sistema usado por los sistemas GPS. Cuando trabajemos con coordenadas geográficas longitud/latitud, este es habitualmente el CRS de referencia.
- **EPSG: 3857:** Código correspondiente a la proyección de Mercator, usada habitualmente por servicios como Google Maps, etc.

Se pueden consultar otros CRS de uso común en España en la página del Ministerio de Agricultura, Pesca y Alimentación

En la sección 3.4.3 veremos cómo encontrar un CRS usando el paquete `crsuggest`.

El paquete `sf` permite obtener los parámetros de cualquier proyección mediante la función `st_crs()`:

**(i) EPSG WGS 84 (Sistema Global GPS): EPSG 4326**

```
library(sf)

# Ejemplo: EPSG WGS 84 (Sistema Global GPS): EPSG 4326
st_crs(4326)
#> Coordinate Reference System:
#>   User input: EPSG:4326
#>   wkt:
#>     GEOGCRS["WGS 84",
#>       DATUM["World Geodetic System 1984",
#>         ELLIPSOID["WGS 84",6378137,298.257223563,
#>           LENGTHUNIT["metre",1]],
#>         PRIMEM["Greenwich",0,
#>           ANGLEUNIT["degree",0.0174532925199433]],
#>         CS[ellipsoidal,2],
#>           AXIS["geodetic latitude (Lat)",north,
#>             ORDER[1],
#>               ANGLEUNIT["degree",0.0174532925199433]],
#>             AXIS["geodetic longitude (Lon)",east,
#>               ORDER[2],
#>                 ANGLEUNIT["degree",0.0174532925199433]],
#>               USAGE[
#>                 SCOPE["Horizontal component of 3D system."],
#>                 AREA["World."],
#>                 BBOX[-90,-180,90,180]],
#>                 ID["EPSG",4326]]]
```

**(ii) ESRI North America Albers Equal Area Conic: ESRI:102008**

```
# Usando código ESRI North America Albers Equal Area Conic

st_crs("ESRI:102008")
#> Coordinate Reference System:
#>   User input: ESRI:102008
#>   wkt:
#>     PROJCRS["North_America_Albers_Equal_Area_Conic",
#>       BASEGEOGCRS["NAD83",
```

```
#>           DATUM["North American Datum 1983",
#>             ELLIPSOID["GRS 1980",6378137,298.257222101,
#>               LENGTHUNIT["metre",1]],
#>             PRIMEM["Greenwich",0,
#>               ANGLEUNIT["Degree",0.0174532925199433]],
#>             CONVERSION["North_America_Albers_Equal_Area_Conic",
#>               METHOD["Albers Equal Area",
#>                 ID["EPSG",9822]],
#>               PARAMETER["Latitude of false origin",40,
#>                 ANGLEUNIT["Degree",0.0174532925199433],
#>                 ID["EPSG",8821]],
#>               PARAMETER["Longitude of false origin",-96,
#>                 ANGLEUNIT["Degree",0.0174532925199433],
#>                 ID["EPSG",8822]],
#>               PARAMETER["Latitude of 1st standard parallel",20,
#>                 ANGLEUNIT["Degree",0.0174532925199433],
#>                 ID["EPSG",8823]],
#>               PARAMETER["Latitude of 2nd standard parallel",60,
#>                 ANGLEUNIT["Degree",0.0174532925199433],
#>                 ID["EPSG",8824]],
#>               PARAMETER["Easting at false origin",0,
#>                 LENGTHUNIT["metre",1],
#>                 ID["EPSG",8826]],
#>               PARAMETER["Northing at false origin",0,
#>                 LENGTHUNIT["metre",1],
#>                 ID["EPSG",8827]]],
#>             CS[Cartesian,2],
#>               AXIS["(E)",east,
#>                 ORDER[1],
#>                 LENGTHUNIT["metre",1]],
#>               AXIS["(N)",north,
#>                 ORDER[2],
#>                 LENGTHUNIT["metre",1]],
#>             USAGE[
#>               SCOPE["Not known."],
#>               AREA["North America - onshore and offshore: Canada -
#>                   Alberta; British Columbia; Manitoba; New Brunswick;
#>                   Newfoundland and Labrador; Northwest Territories; Nova Scotia;
#>                   Nunavut; Ontario; Prince Edward Island; Quebec; Saskatchewan;
#>                   Yukon. United States (USA) - Alabama; Alaska (mainland);
#>                   Arizona; Arkansas; California; Colorado; Connecticut; Delaware;
#>                   Florida; Georgia; Idaho; Illinois; Indiana; Iowa; Kansas;
#>                   Kentucky; Louisiana; Maine; Maryland; Massachusetts; Michigan;
#>                   Minnesota; Mississippi; Missouri; Montana; Nebraska; Nevada;
#>                   New Hampshire; New Jersey; New Mexico; New York; North
#>                   Carolina; North Dakota; Ohio; Oklahoma; Oregon; Pennsylvania;
#>                   Rhode Island; South Carolina; South Dakota; Tennessee; Texas;
#>                   Utah; Vermont; Virginia; Washington; West Virginia; Wisconsin;
#>                   Wyoming."],
```

```
#>           BBOX[23.81,-172.54,86.46,-47.74]],  
#>           ID["ESRI",102008]]
```

(iii) Usando proj4string: Robinson: +proj=robin

```
# Usando proj4string: Robinson  
  
st_crs("+proj=robin")  
#> Coordinate Reference System:  
#>   User input: +proj=robin  
#>   wkt:  
#> PROJCRS["unknown",  
#>   BASEGEOGCRS["unknown",  
#>     DATUM["World Geodetic System 1984",  
#>       ELLIPSOID["WGS 84",6378137,298.257223563,  
#>         LENGTHUNIT["metre",1]],  
#>       ID["EPSG",6326]],  
#>     PRIMEM["Greenwich",0,  
#>       ANGLEUNIT["degree",0.0174532925199433],  
#>       ID["EPSG",8901]],  
#>     CONVERSION["unknown",  
#>       METHOD["Robinson"],  
#>       PARAMETER["Longitude of natural origin",0,  
#>         ANGLEUNIT["degree",0.0174532925199433],  
#>         ID["EPSG",8802]],  
#>       PARAMETER["False easting",0,  
#>         LENGTHUNIT["metre",1],  
#>         ID["EPSG",8806]],  
#>       PARAMETER["False northing",0,  
#>         LENGTHUNIT["metre",1],  
#>         ID["EPSG",8807]]],  
#>   CS[Cartesian,2],  
#>     AXIS["(E)",east,  
#>       ORDER[1],  
#>       LENGTHUNIT["metre",1,  
#>         ID["EPSG",9001]]],  
#>     AXIS["(N)",north,  
#>       ORDER[2],  
#>       LENGTHUNIT["metre",1,  
#>         ID["EPSG",9001]]]]
```

La mayoría de los objetos espaciales serán de la clase **sf**, por tanto, resulta intere-

sante conocer cómo se proyectan estos objetos.

Es posible proyectar un objeto `sf` mediante la función `st_transform()`. En el siguiente ejemplo vemos cómo partimos de un objeto con **EPSG:4326** y cambiamos su proyección a otras proyecciones, como **Mercator** o **Robinson**:

```
# Usa datos del paquete giscoR

library(giscoR)

paises <- gisco_get_countries()

# Comprobamos el CRS de estos datos
# Se puede almacenar en un objeto y usar posteriormente
# Vemos que es EPSG:4326, por tanto son coordenadas geográficas
#> longitud/latitud
st_crs(paises)
#> Coordinate Reference System:
#>   User input: EPSG:4326
#>   wkt:
#>     GEOGCS["WGS 84",
#>       DATUM["WGS_1984",
#>         SPHEROID["WGS 84",6378137,298.257223563,
#>           AUTHORITY["EPSG","7030"]],
#>         AUTHORITY["EPSG","6326"]],
#>       PRIMEM["Greenwich",0,
#>         AUTHORITY["EPSG","8901"]],
#>       UNIT["degree",0.0174532925199433,
#>         AUTHORITY["EPSG","9122"]],
#>       AUTHORITY["EPSG","4326"]]

# Plot
plot(st_geometry(paises), axes = TRUE)
```

```
# Proyectamos a Mercator
# El eje cambia porque Mercator usa metros
paises_merc <- st_transform(paises, st_crs(3857))
plot(st_geometry(paises_merc), axes = TRUE)
```

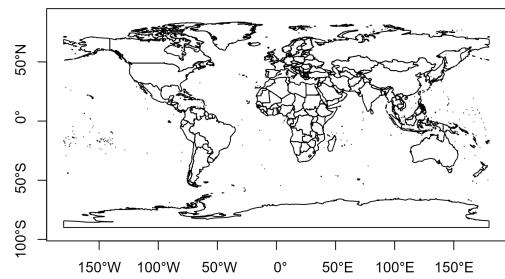


Figura 3.12: Proyección del mundo en coordenadas geográficas (EPSG 4326)

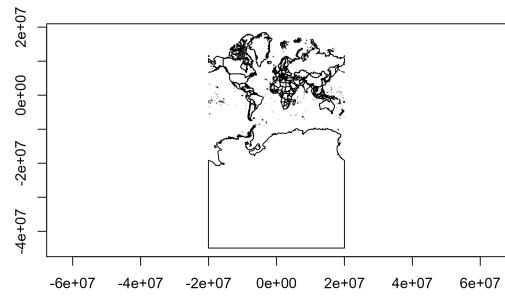


Figura 3.13: Proyección del mundo en Mercator (EPSG 3857)

```
# Proyectamos a Robinson
paises_robin <- st_transform(paises, st_crs("+proj=robin"))
plot(st_geometry(paises_robin), axes = TRUE)
```

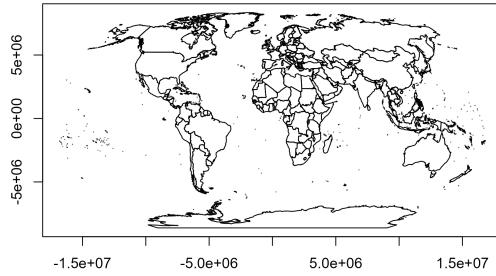


Figura 3.14: Proyección del mundo en Robinson (+proj=robin)

Como se comentó anteriormente, cuando se usan geodatos de diversas fuentes, es necesario que todos presenten el mismo CRS. En la Fig 3.15 se muestra lo que ocurre si esto no se cumple:

```
# Añadimos a este mapa puertos mundiales de giscoR
puertos <- gisco_get_ports()
plot(st_geometry(paises_robin), main = "Puertos en el mundo")
plot(st_geometry(puertos), add = TRUE, col = "red", pch = 20)
```

Vemos que ha habido algún tipo de error, ¿a qué puede deberse?

```
# Comprueba CRS
st_crs(puertos) == st_crs(paises_robin)
#> [1] FALSE

# Los puertos no están en Robinson! Proyectamos al mismo CRS
puertos_robin <- st_transform(puertos, st_crs(paises_robin))
plot(st_geometry(paises_robin), main = "Puertos en el mundo")
plot(st_geometry(puertos_robin), add = TRUE, col = "blue", pch =
  20)
```



Figura 3.15: Ejemplo: Puertos del mundo

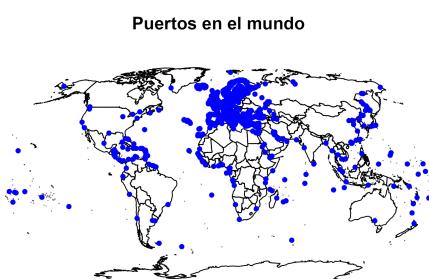


Figura 3.16: Ejemplo: Puertos del mundo, CRS alineados

Como vemos, en el primer mapa (Fig. 3.15) los puertos se concentran en un único punto, dado que no están referenciados en el mismo CRS. Tras proyectarlos al mismo CRS, el mapa se representa adecuadamente (Fig. 3.16).

En otros paquetes, como **sp** o **raster**, existen funciones parecidas que nos van a permitir obtener los parámetros de un CRS y proyectar los objetos al CRS deseado. Cuando empleemos el paquete **sp** podemos usar las funciones **CRS()** y **spTransform()**:

```
library(sp)

# Convertimos sf a sp
paises_sp <- as(paises, "Spatial")

# En sp podemos usar:
# CRS("+proj=robin")
#
# O también desde sf
# CRS(st_crs(paises_robin)$proj4string)

paises_sp_robin <- spTransform(paises_sp, CRS("+proj=robin"))
plot(paises_sp_robin)
```

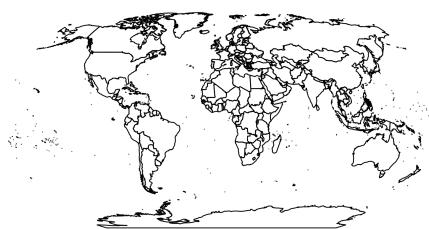


Figura 3.17: Transformaciones en sp

En el caso de un objeto **raster**, podemos usar **crs()** y **projectRaster()**:

```
library(raster)

# Extrae información de altitud para España
elev <- raster("data/ESP_msk_alt.grd")

# Transforma
elev_robinson <- projectRaster(elev, crs = crs("+proj=robin"))
plot(elev_robinson)
```

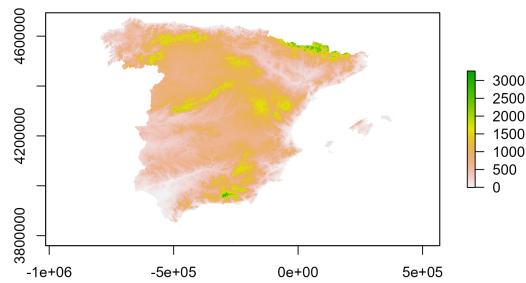


Figura 3.18: Transformaciones en raster

Por último, en el paquete `terra` las funciones correspondientes son `crs()` y `project()`:

```
library(terra)

# Convierte de raster a terra
elev_terra <- rast(elev)

# Transforma
elev_terra_robinson <- terra::project(elev_terra,
                                         crs(elev_terra))
plot(elev_terra_robinson)
```

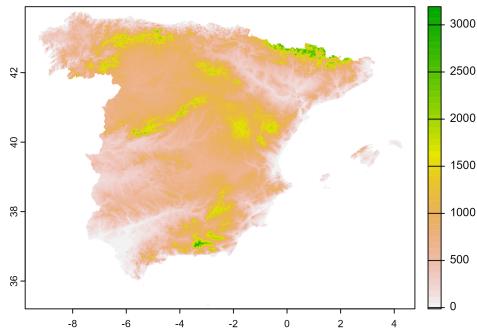


Figura 3.19: Transformaciones en terra

### 3.4.3. ¿Qué proyección uso?

El CRS adecuado para cada análisis depende de la localización y el rango espacial de los datos. Un CRS adecuado para representar un mapa del mundo puede no serlo para representar datos de zonas específicas de la Tierra. Los recursos web mencionados anteriormente permiten la búsqueda de CRS por zona geográfica, y adicionalmente en **R** existe el paquete **crssuggest** [Walker, 2021] que nos facilita la labor, sugiriendo el CRS más adecuado para cada zona:

```
library(crsuggest)

# Usando raster
sugerencias <- suggest_crs(elev)
```

```
# Probamos sugerencia
crs_suggest <- suggest_crs(elev, limit = 1)

elev_suggest <- projectRaster(elev, crs =
  ↪ raster::crs(crs_suggest$crs_proj4))

plot(elev_suggest)
```

```
# Ejemplo con sf: China
```

Cuadro 3.3: Tabla sugerencias, detalle

crs_code	crs_name	crs_type	crs_gcs	crs_units	crs_proj4
2062	Madrid 1870 (Madrid) / Spain LCC	projected	4903	m	+proj=lcc +lat_1
2154	RGF93 / Lambert-93	projected	4171	m	+proj=lcc +lat_0
26191	Merchich / Nord Maroc	projected	4261	m	+proj=lcc +lat_1
3944	RGF93 / CC44	projected	4171	m	+proj=lcc +lat_0
3943	RGF93 / CC43	projected	4171	m	+proj=lcc +lat_0
27573	NTF (Paris) / Lambert zone III	projected	4807	m	+proj=lcc +lat_1
27572	NTF (Paris) / Lambert zone II	projected	4807	m	+proj=lcc +lat_1
27563	NTF (Paris) / Lambert Sud France	projected	4807	m	+proj=lcc +lat_1
30791	Nord Sahara 1959 / Nord Algerie	projected	4307	m	+proj=lcc +lat_1
30493	Voirol 1879 / Nord Algerie (ancienne)	projected	4671	m	+proj=lcc +lat_1

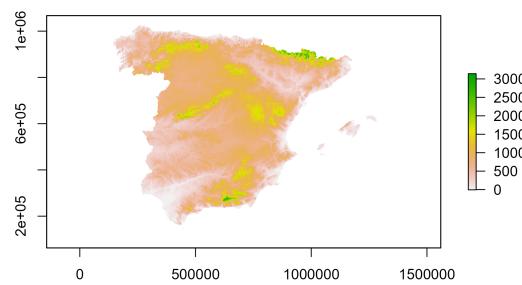


Figura 3.20: raster: Ejemplo de transformación usando crssuggest

```
china <- gisco_get_countries(country = "China")
china_crs <- suggest_crs(china, limit = 1)

china_crs
#> # A tibble: 1 x 6
#>   crs_code crs_name      crs_type crs_gcs crs_units crs_proj4
#>   <chr>     <chr>       <chr>      <dbl>   <chr>      <chr>
#> 1 4584     New Beijing / ~ project~    4555 m
#>   +proj=tmerc +lat_0=0 +lon~

china_suggest <- st_transform(
  china,
  st_crs(as.integer(china_crs$crs_code))
)

plot(st_geometry(china_suggest), axes = TRUE)
```

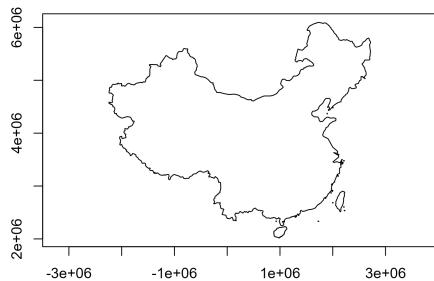


Figura 3.21: sf: Ejemplo de transformación usando crssuggest

## Capítulo 4

# Estadística espacial

La estadística espacial se basa en la suposición de que las unidades georreferenciadas cercanas están relacionadas (son **dependientes**) de alguna manera [Getis, 1999], y por ello, trata de reconocer y aprovechar ésta ubicación espacial a la hora de diseñar, recopilar, gestionar, analizar y mostrar las observaciones (Montero et al. [2011]).

Los métodos estadísticos espaciales que se utilizan actualmente, y sobre los que continúan las investigaciones, incluyen el estudio de la asociación espacial, el análisis de patrones, la escala y la zonificación, la geoestadística, la clasificación, el muestreo espacial y la econometría espacial. Estos métodos se aplican a una gran variedad de disciplinas científicas. Por ejemplo, Montero et al. [2011] destacan los siguientes. Los orígenes de la vida humana vinculan los estudios de la evolución de las galaxias, la estructura de las células biológicas y los patrones de asentamiento arqueológicos. Los ecólogos estudian las interacciones entre plantas y animales. Silvicultores y agricultores necesitan investigar las variaciones que se producen en el terreno para sus experimentos. La estimación de las precipitaciones y de las reservas de oro y petróleo es de vital importancia económica. Estos son, entre otros, buenos ejemplos de la importancia del espacio (espacio-tiempo en su caso) en el mundo de la Ciencia.

Sin embargo, el estudio de la **variabilidad espacial**, y sobre todo espacio-temporal, es una disciplina relativamente nueva en el marco de la Estadística, lo que explica la escasez de instrumentos de estadística espacial 30 años atrás. En los últimos 10 años ha habido una creciente toma de conciencia de esta necesidad, habiéndose realizado un gran esfuerzo por buscar herramientas adecuadas y útiles a tales efectos. Y todo ello porque utilizar modelos espaciales o espacio-temporales para caracterizar y explotar la dependencia espacial (o espacio-temporal) de un conjunto de observaciones tiene importantes ventajas (Montero et al. [2011]):

1. Modelos más generales, ya que, en la mayoría de los casos, los modelos clásicos que no tienen en consideración la dimensión espacial o la interacción de las dimensiones espacial y temporal son un caso particular de un modelo espacial o espacio-temporal.
2. Estimaciones más eficientes: de la tendencia, de los efectos de las variables explicativas, de promedios regionales,...
3. Mejora de las predicciones: más eficientes, con propiedades de extrapolación más estables,...
4. La variación espacial no explicada en la estructura de la media debe ser absorbida por la estructura del error, por lo que un modelo que incorpore la dependencia espacial puede decirse que está protegido frente a una mala especificación de este tipo. Esto, en muchos casos, tiene como resultado una simplificación en la especificación de la tendencia; en general, los modelos con dependencia espacial suelen tener una descripción más parsimoniosa (en ocasiones con muchos menos parámetros) que los clásicos modelos de superficie de tendencia.

#### 4.1. Antes de continuar... dependencia espacial.

Frecuentemente los datos tienen una componente espacial y/o temporal asociada a ellos y es de esperar que datos cercanos en el espacio o en el tiempo sean más semejantes que aquellos que están más alejados; en cuyo caso **no** deben ser modelados como estadísticamente independiente, sino que habrá que tomar en cuenta esa dependencia espacial o espacio-temporal.

De forma natural y de acuerdo a la Ley Tobler (1973) surge la idea de que los datos cercanos en el espacio o en el tiempo serán más similares y estarán más correlacionados entre sí que aquellos que están más lejanos. Por ejemplo, la influencia de un terremoto, su efecto disminuye con distancia del epicentro. Además, esta correlación disminuye al aumentar la separación entre ellos, por lo que se puede pensar en la presencia de una dependencia espacial o espacio-temporal.

Si los datos no exhiben dependencia espacial no tiene sentido aplicar las herramientas de estadística espacial. ¿Y cómo se pueden reconocer un proceso con dependencia espacial? La clave está en que los procesos con dependencia espacial exhiben un patrón en el espacio, mientras que los que son independientes son totalmente aleatorios.

La Fig. 4.1 muestra uno campo aleatorio simulado que presentan una estructura de dependencia espacial (panel izquierdo) frente a unos datos totalmente aleatorios (panel derecho), en ambos casos distribuidos de forma irregular en el espacio. En la

Imagen de dependencia espacial se observa como los puntos rojos (que representan valores más altos de la variable simulada) se concentran en la misma zona, al igual que los puntos azules (que representan valores más bajos) que se distribuyen en otra zona y juntos.

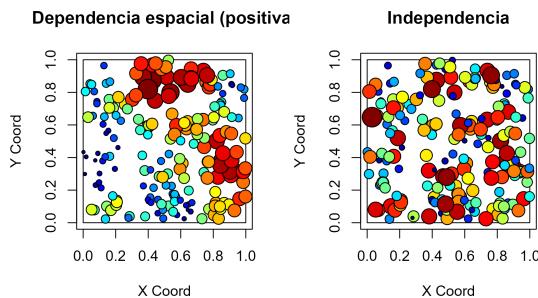


Figura 4.1: Puntos: Simulación patrón de independencia espacial (derecha) frente a dependencia espacial (izquierda)

La Fig. 4.2, al igual que la Fig. 4.1 presenta unos datos simulados donde que muestran una estructura de dependencia espacial (panel izquierdo) frente a unos datos totalmente aleatorios (panel derecho), pero en este caso estos datos se distribuyen de forma ordenada en el espacio a través de una rejilla regular. La interpretación es análoga a la de la Fig. 4.1, solo que en este caso los campos aleatorios simulados se representan en una rejilla.

## 4.2. Datos espaciales

Los **datos espaciales**, también conocidos como datos **geoespaciales**, son aquellos datos relacionados o que contienen información de una localización o área geográfica de la superficie de la Tierra.

La forma más intuitiva de representar los datos espaciales es a través de un mapa.

```
# Mapa de porcentaje de mujeres en Castilla-La Mancha
library(mapSpain)
# Datos de población
```

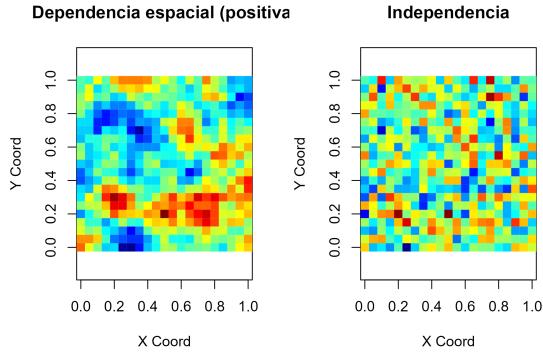


Figura 4.2: Rejilla: Simulación patrón de independencia espacial (derecha) frente a dependencia espacial (izquierda)

```
pob <- mapSpain::pobmun19

# Datos en forma de tabla, sin información en formato espacial
# head(pob)

# Porcentaje
pob$porc_mujeres <- pob$women / pob$pob19 * 100

# Datos espaciales
geo <- esp_get_munic(region = "Castilla-La Mancha")

# Estos datos tienen una columna (geometry) con coordenadas.
# head(geo)

# Une ambos datos
geo_pob <- merge(geo,
  pob,
  by = c("cpro", "cmun"),
  all.x = TRUE
)

# Mapa básico
plot(geo_pob["porc_mujeres"],
  # Cambiamos título
```

```
main = "Castilla-La Mancha: % mujeres (2019)",  
  
# Cambiamos la paleta de colores para hacerlo mas atractivo  
# border = NA,  
pal = hcl.colors(12, "RdYlBu")  
)
```

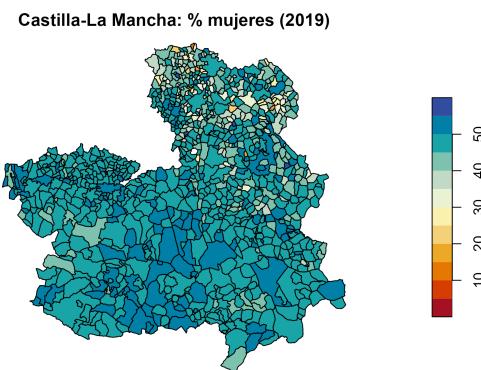


Figura 4.3: Porcentaje de Mujeres en Castilla-La Mancha

La Fig. 4.3 presenta una serie de elementos gráficos, característicos de los objetos espaciales:

- Los municipios de Castilla-la Mancha están representados por polígonos con un contorno negro y se rellenan de colores de acuerdo con la variable que estamos analizando, el porcentaje de mujeres en los municipios de Castilla-La Mancha en el año 2019.
- Una leyenda explica el significado de los colores.
- La variable, el porcentaje de mujeres en los municipios de Castilla-La Mancha en el año 2019, no parece distribuirse de manera independiente sino todo lo contrario, muestra un patrón espacial. Los municipios del las provincias Guadalajara y Cuenca (noreste) presentan tasas más bajas que los municipios del centro de la Comunidad.

```
head(geo_pob["porc_mujeres"])  
#> Simple feature collection with 6 features and 1 field  
#> Geometry type: POLYGON
```

```
#> Dimension:      XY
#> Bounding box:  xmin: -2.18037 ymin: 38.5441 xmax: -1.31112 ymax:
#>           ↗ 39.35597
#> Geodetic CRS:   ETRS89
#>   porc_mujeres           geometry
#> 1 52.02532 POLYGON ((-1.58316 39.20446...
#> 2 43.93064 POLYGON ((-1.40607 39.12384...
#> 3 51.14089 POLYGON ((-2.0562 38.88697, ...
#> 4 48.55491 POLYGON ((-1.54055 38.61066...
#> 5 48.78419 POLYGON ((-1.38514 39.35429...
#> 6 44.49541 POLYGON ((-2.15635 38.71074...
```

Antes de dibujar la Fig. 4.3 tuvimos que leer los datos de la librería `mapSpain` [Hernández, 2022] que contenía tanto la variable que hemos analizado como el formato del mapa. Tras unir variable y mapa con la función `merge()`. Al llevar a cabo un resumen del objeto espacial nos encontramos con las siguiente información:

- el conjunto de datos (seleccionado) tiene 919 registros (municipios)
- el tipo de geometría es POLYGON.
- el CRS es ETRS89

### 4.3. Clasificación de datos espaciales

Tal y como acabamos de señalar y de acuerdo con Schabenberger [2005], p. 6), debido a que los datos espaciales surgen en una gran variedad de campos y aplicaciones, también hay una gran variedad de tipos de datos espaciales, estructuras y escenarios. Por tanto, una clasificación exhaustiva de los datos espaciales sería un reto muy difícil y hemos apostado por una clasificación general, simple y útil de datos espaciales proporcionada por Cressie [1993].

La **clasificación** de Cressie de datos espaciales se basa en la naturaleza del dominio espacial en estudio. Dependiendo de esto, podemos tener: datos geoestadísticos, datos de patrones de puntos y datos lattice (véase Fig. 4.4 en la que se han simulado los distintos procesos espaciales).

Siguiendo a Cressie [1993], sea  $s \in \mathbb{R}^d$  una localización en un espacio Euclídeo  $d$ -dimensional y  $Z(s) : s \in \mathbb{R}^d$  una función aleatoria espacial, donde  $Z$  representa el atributo en el cual estamos interesados, se tendría la siguiente clasificación [Montero et al., 2015]:

1. **Datos geoestadísticos:** Surgen cuando el dominio de estudio es **continuo y fijo**  $D$ . Es decir: (i)  $Z(s)$  se puede observar en cualquier punto del dominio

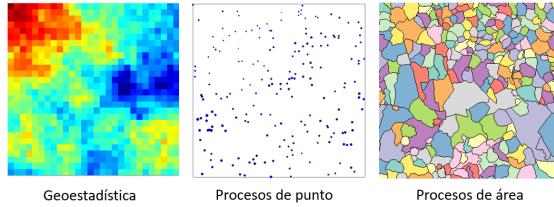


Figura 4.4: Clasificación de datos espaciales propuesta por Cressie (1993))

(continuo); y (ii) los puntos en  $D$  no son estocásticos (son fijos,  $D$  es el mismo para todas las realizaciones de la función aleatoria espacial).

Algunos ejemplos de datos geoestadísticos son el nivel de un contaminante en una ciudad, los valores de precipitación o temperatura del aire en un país, las concentraciones de metales pesados en la capa superior del suelo de una región, etc.

Es obvio que, al menos en teoría, el nivel de un contaminante específico podría medirse en cualquier lugar de la ciudad; Lo mismo puede decirse de las mediciones de precipitaciones o temperaturas del aire en un país o concentraciones de un metal pesado en una región. Sin embargo, en la práctica, no es posible una observación exhaustiva del proceso espacial. Por lo general, el proceso espacial se observa en un conjunto de ubicaciones (por ejemplo, el nivel de un contaminante específico en una ciudad se observa en los puntos donde están ubicadas las estaciones de monitoreo) y, basado en tales valores observados, el análisis geoestadístico reproduce el comportamiento de el proceso espacial en todo el dominio de interés. La Fig. @ref(fig:ejem\_geo) presenta un mapa de interpolación krigeado del nivel de monóxido de carbono (CO) en logaritmos para la ciudad de Madrid.

En el análisis geoestadístico lo más importante es cuantificar la correlación espacial entre observaciones (a través de la herramienta básica en geoestadística, el semivariograma) y utilizar esta información para lograr los objetivos anteriores (véase Montero et al. [2015] para un análisis profundo de los métodos geoestadísticos espaciales y espacio-temporales}.

## AQUÍ ERROR, SÓLO ES UN PLOT

2. **Datos reticulares:** Surgen cuando: (i) el dominio bajo estudio  $D$  es **discreto**, es decir,  $Z(s)$  puede observarse en una serie de ubicaciones fijas que pueden enumerarse. Estas ubicaciones pueden ser puntos o regiones, pero generalmente son códigos postales, pistas censales, vecindarios, provincias, países, etc., y los datos en la mayoría de los casos son datos agregados espacialmente sobre estas áreas. Aunque estas regiones pueden tener una forma

regular, normalmente la forma que tienen es irregular, y esto, junto con el carácter espacialmente agregado de los datos, es por lo que los datos lattice tambien se denominan datos regionales. Y (ii) las ubicaciones en  $D$  no son estocásticas. Por supuesto, un concepto clave en el análisis de los datos lattice es el **vecindario** y la matriz **W**.

Algunos ejemplos de reticulares incluyen la tasa de desempleo por estados, los datos de delincuencia por comarcas, rendimientos agrícolas en parcelas, precios medios de la vivienda por provincias, etc. En este sentido un ejemplo ilustrativo se presenta en la la Fig. 4.5, que muestra la distribución municipal de la renta neta media por hogar en 2017 en España.

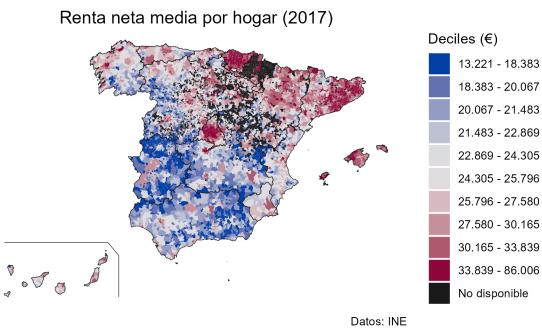


Figura 4.5: Renta neta media municipal española en 2017

3. **Procesos de puntos:** Mientras que en los datos geoestadísticos y reticulares el dominio  $D$  es fijo, en los datos de patrones puntuales el dominio es discreto o continuo, pero **aleatorio**. Los patrones de puntos surgen cuando el atributo bajo estudio es la ubicación de los eventos (observaciones). Es decir, el interés radica en dónde ocurren eventos de interés.

Algunos ejemplos de patrones de puntos son la ubicación de incendios en una región española, la ubicación de los árboles en un bosque o la ubicación de nidos en una colonia de aves reproductoras, la localización de los accidentes de tráfico en una ciudad o área de referencia, la localización de los delitos en una ciudad, entre muchas otras.

En estos En los casos, es obvio que  $D$  es aleatorio y los puntos de observación no dependen del investigador. El principal objetivo del análisis de patrones de puntos es determinar si la ubicación de los eventos tiende a exhibir un patrón

sistemático sobre el área en estudio o, por el contrario, son aleatoriamente repartido.

Más concretamente, nos interesa analizar si la ubicación de los eventos es completamente aleatorio espacialmente (la ubicación donde ocurren los eventos no se ve afectada por la ubicación de otros eventos), uniforme o regular (cada punto está tan lejos de todos sus vecinos como sea posible) o agrupados o agregados (la ubicación de los eventos se concentra en grupos). La Fig. 4.6 muestra la localización de los accidentes de tráfico registrados en la ciudad de Madrid durante el mes de febrero de 2020.

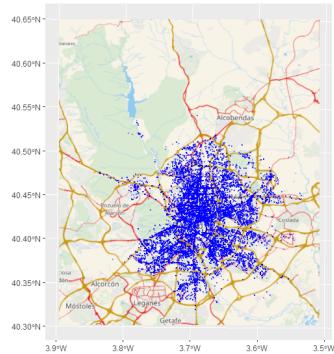


Figura 4.6: Accidentes de tráfico registrados en la ciudad de Madrid (Febrero de 2020)

# Capítulo 5

## Casos prácticos

DIEGO, EN EL CSS SE PUEDE AÑADIR UN CUADRO QUE SEA UNA ADVERTENCIA, YO NO SÉ, LO PROBAMOS/PRUEBAS :) O NO??

### 5.1. Caso 1: Temperatura mínimas del aire en España.

#### Objetivo de aprendizaje

Esta sección presenta un caso de uso en el que aprenderemos a realizar las siguientes tareas básicas:

1. Leer datos espaciales en R.
2. Proyectar datos espaciales.
3. Graficar datos espaciales.

Para ello, se va a trabajar con los datos de temperatura mínima registradas en España por las estaciones meteorológicas de la Agencia Estatal de Meteorología (AEMET). Un conjunto de datos ya depurados y listos para trabajar se encuentra en el fichero `tempmin.csv`. Por otra parte, la información relativa al mapeo de España se obtendrá directamente de la librería `mapSpain`. Todo el análisis se va a realizar empleando RStudio, por lo que se empezará abriendo el programa y creando un nuevo proyecto.

**Ejercicio 5.1** (Creación del proyecto). Cree un proyecto para trabajar todo lo referente al caso.

*Solución.* Para crear un proyecto siga la secuencia: *File> New Project> New File> RMD*

El conjunto de datos proporcionado `tempmin.csv` contiene el nivel de temperatura del aire en España entre el 6 y el 10 de Enero de 2021<sup>1</sup>. Estos datos han sido descargados usando la librería `climaemet` [Pizarro et al., 2021] y han sido posteriormente tratados para su uso en esta práctica.

El primer paso consiste en importar la base de datos de temperatura mínima. El archivo está en formato csv, por lo cual, es un fichero de texto plano. Se pueden usar varias funciones para realizar la importación. Se emplara'na los paquetes del `tidyverse` para realizar todo el tratamiento de datos.

**Ejercicio 5.2** (Importación de los datos). Importe los datos `tempmin.csv` y guárdelos en un objeto llamado `tmin`.

```
Solución: uno debe seleccionar el directorio donde tiene los datos, de
        ↵ ahí
# que sea conveniente trabajar con proyectos.
library(readr)
tmin <- read_csv("data/tempmin.csv")
```

Los el conjunto de datos `tempmin` es un data frame que contiene 5 variables:

- `fecha`: Indicando la fecha de observación.
- `indicativo`: Es el identificador de la estación de la AEMET que registró el dato.
- `tmin`: Dato de temperatura mínima registrada en cada fecha por la estación correspondiente en grados centígrados.
- `longitud, latitud`: Coordenadas geográficas de la estación

**Ejercicio 5.3** (Descripción de los datos). Con la función `head()` describa, en forma de tabla, la información que contiene el objeto `tmin` y compruebe que se corresponde con la descrita.

<sup>1</sup>Las fechas seleccionadas coinciden con el periodo en el que la tormenta Filomena tuvo su auge en la Península Ibérica.

Cuadro 5.1: Detalle del objeto tmin

fecha	indicativo	tmin	longitud	latitud
2021-01-06	4358X	-4.7	-5.880556	38.95556
2021-01-06	4220X	-7.0	-4.616389	39.08861
2021-01-06	6106X	4.7	-4.748333	37.02944
2021-01-06	9698U	-6.8	0.865278	42.20528
2021-01-06	4410X	-3.4	-6.385556	38.91583
2021-01-06	1331A	1.0	-7.031389	43.52472

```
knitr::kable(head(tmin), caption = "Detalle del objeto tmin")
```

Una de las clases de objetos espaciales más utilizada en R es **sf**. Sin embargo, dependiendo del análisis que se quiera realizar hay otras muy comunes como **geodata**, **spatastat**, etc..

A continuación se convertirá el objeto **tmin** (**data.frame**) a un objeto de la clase **geodata**, una clase muy utilizada para trabajar con datos espaciales y requerida por la librería **geoR**. Estos objetos contienen las coordenadas y la variable objeto de estudio. Para mayor detalle ver **??as.geodata**. Obsérvese como varía la variable a través de los ejes longitud y latitud. Observe también la forma campaniforme de la distribución de la variable.

**Ejercicio 5.4** (Descripción de los datos). Del objeto **tmin** seleccione el día **8 de enero de 2022** y las variables **longitud**, **latitud** y **tmin** para crear el objeto y llámelo **tmin\_geor**. A continuación describa analíticamente y gráficamente dicho objeto.

```
library(dplyr)
library(geoR)

tmin_geor <- tmin %>%
  filter(fecha == "2021-01-08") %>%
  # Seleccionamos las columnas de interés
  dplyr::select(longitud, latitud, tmin) %>%
  # Y creamos el objeto geodata
  as.geodata(
    coords.col = 1:2,
    data.col = 3
  )
```

```

summary(tmin_geoR)
#> Number of data points: 211
#>
#> Coordinates summary
#>      longitud latitud
#> min -9.291389 35.27639
#> max 4.215556 43.78611
#>
#> Distance summary
#>      min           max
#> 0.01024389 13.85144264
#>
#> Data summary
#>      Min.    1st Qu.    Median    Mean    3rd Qu.
#>      Max.
#> -14.9000000 -4.6000000 -0.5000000 -0.6293839  3.5000000
#>      13.6000000
plot(tmin_geoR)

```

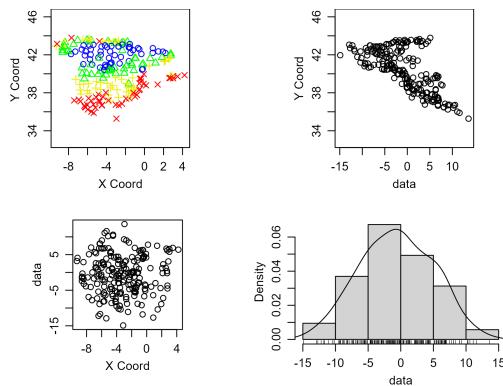


Figura 5.1: Convertir ‘data.frame’ a ‘geodata’

En esta ocasión, convertiremos los datos de `tmin` en un objeto espacial `sf`, es decir, datos espaciales de tipo vector.

Los datos de `tmin` contienen coordenadas geográficas longitud/latitud, así que como se vió en la Sección **Sistema de Referencia de Coordenadas (CRS)** el CRS a emplear ha de ser un CRS geográfico. Usaremos el código EPSG **4326**, que corresponde a coordenadas geográficas y suele ser el habitual en este tipo de situaciones.

**Ejercicio 5.5** (Convertir ‘data.frame’ a ‘sf’). Del objeto `tmin` seleccione las variables `longitud`, `latitud` y `tmin` para crear el objeto sf y llámelo `tmin_sf`. A continuación describa el objeto creado.

```
library(sf)

tmin_sf <- st_as_sf(tmin,
  coords = c("longitud", "latitud"),
  crs = 4326
)

tmin_sf
#> Simple feature collection with 1066 features and 3 fields
#> Geometry type: POINT
#> Dimension:      XY
#> Bounding box:  xmin: -9.291389 ymin: 35.27639 xmax: 4.215556
#>           ymax: 43.78611
#> Geodetic CRS:  WGS 84
#> # A tibble: 1,066 x 4
#>   fecha     indicativo  tmin          geometry
#>   * <date>    <chr>    <dbl>        <POINT [°]>
#> 1 2021-01-06 4358X     -4.7 (-5.880556 38.95556)
#> 2 2021-01-06 4220X     -7   (-4.616389 39.08861)
#> 3 2021-01-06 6106X     4.7  (-4.748333 37.02944)
#> 4 2021-01-06 9698U    -6.8  (0.865278 42.20528)
#> 5 2021-01-06 4410X    -3.4  (-6.385556 38.91583)
#> 6 2021-01-06 1331A      1   (-7.031389 43.52472)
#> 7 2021-01-06 1690A    -0.1  (-7.859722 42.32528)
#> 8 2021-01-06 8489X     -8   (-0.255833 40.43333)
#> 9 2021-01-06 8025       2   (-0.494167 38.3725)
#> 10 2021-01-06 9784P    -10  (0.224722 42.63)
#> # ... with 1,056 more rows
```

La siguiente tarea será representar las estaciones que monitorizan la temperatura mínima en un mapa de España.

**Ejercicio 5.6** (Reprsentación espacial de la clase sf). Represente un mapa de ESpaña, con las Comunidades Autónomas de incluidas, excepto las Islas Canarias (por simplicidad)

*Solución.* Una opción es utilizar un paquete API que nos proporciona esta información en formato sf, el paquete `mapSpain`.

```
library(mapSpain)
# sf object
esp <- esp_get_ccaa() %>%
  # No vamos a usar Canarias en este análisis
  filter(ine.ccaa.name != "Canarias")

plot(esp$geometry) # Dibujamo el mapa de España menos las Islas
  ↵ Canarias
```



Figura 5.2: Mapa de España (Sin Canarias)

Como se comentó en la sección **Sistema de Referencia de Coordenadas (CRS)**, cuando se emplean datos geográficos provenientes de varias fuentes, es necesario asegurarse de que ambos objetos están usando el mismo CRS.

**Ejercicio 5.7** (CRS). ¿Tengo el Sistema de referencia de coordenadas (CRS) de las estaciones de monitoreo en la misma proyección que el contorno de España? Compruébelo.

```
st_crs(tmin_sf) == st_crs(esp)
#> [1] FALSE
```

Se ha comprobado que no lo están, por lo que hay que proyectar las coordenadas a un CRS común.

**Ejercicio 5.8** (CRS). Proyecte las coordenadas de los objetos `tmin_sf` y `esp` al CRS de referencia de `tmin_sf`. Compruébelo.

```
esp2 <- st_transform(esp, st_crs(tmin_sf))

st_crs(tmin_sf) == st_crs(esp2)
#> [1] TRUE
```

Para dibujar las estaciones de monitoreo con el contorno de España, existen varias opciones de paquetes, `ggplot2` (paquete de referencia en representaciones gráficas), `tmap` o `maps` (estos dos últimos especializados en mapas temáticos,)

paquete `ggplot2` como referencia, sin embargo existen varios paquetes

**Ejercicio 5.9** (CRS). Represeñe, con el paquete `ggplot2`, las estaciones de monitoreo de AEMET en la península Ibérica.

```
library(ggplot2)

ggplot(esp2) +
  # Para graficar objetos sf debemos usar geom_sf()
  geom_sf() +
  geom_sf(data = tmin_sf) +
  theme_light() +
  # labs(
  #   title = "Estaciones de monitoreo AEMET en España",
  #   subtitle = "excluyendo las Islas Canarias"
  # ) +
  theme(
    plot.title = element_text(
      size = 12,
      face = "bold"
    ),
    plot.subtitle = element_text(
      size = 8,
      face = "italic"
    )
  )
```

Una vez represtadas las coordenadas, es decir, las estaciones de monitoreo dónde se ha medido la variable temperatura mínima `tmin`, el siguiente paso será representar el valor que toma la variable en esas coodenadas. La base `tmin` contiene información temporal para para varios días, por lo que, como este análisis es meramente espacial se elegirá un día que se fijará para todo el análisis.

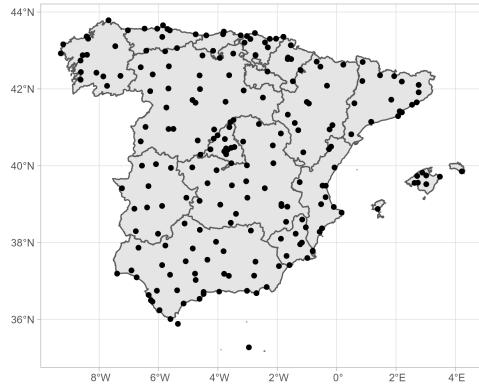


Figura 5.3: Estaciones de AEMET en la Península Ibérica

**Ejercicio 5.10** (plot-sp1). Representamos la variable temperatura mínima `tmin` para el día **8 de enero de 2021**. Guarde la base de datos espacial para ese día en un objeto de nombre `tmin_8enero`.

```
# Seleccionaremos los datos correspondientes al 8 de enero de 2021
tmin_8enero <- tmin_sf %>%
  filter(fecha == "2021-01-08")

# Mapa temático en el que se representan los valores de temperatura
# registrados en cada estación mediante un código de colores
plot(tmin_8enero[["tmin"]],
  # main = "Temperatura mínima (8-enero-2021)",
  pch = 8
)
```

El mapa ha quedado muy bien, pero quizás los colores y el formato elegido no sean los más adecuados para este tipo de representaciones...

**Ejercicio 5.11** (plot-sp2). Utilice los parámetros espaciales de los que dispone, las coordenadas y el contorno de España para graficar y contar la historia de **Filomena** adecuadamente.

La visión que ofrece el la Fig. @ref{fig:spatial-plots} de Filomena es muy informativa, vemos como los datos nos cuentan la historia de lo que ocurrió ese día. La pena

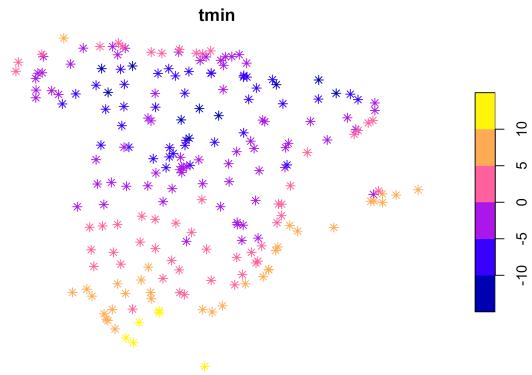


Figura 5.4: Mapa de puntos con temperatura mínima (8-enero-2021)

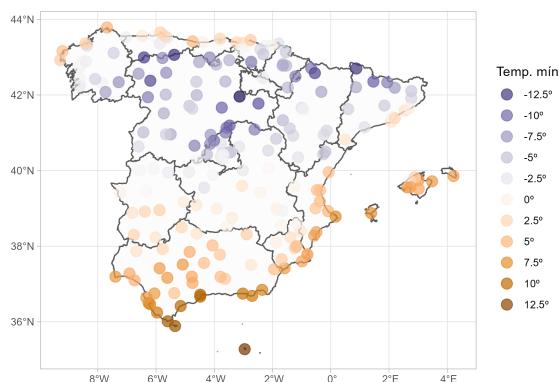


Figura 5.5: Mapa completo con temperatura mínima (8-enero-2021)

es que no existan estaciones de monitorero en todos los puntos de España para conocer el valor de la temperatura mínima en cualquier lugar del país. ¿Podríamos tener un mapa de interpolación para tener una estimación de la temperatura mínima en las partes donde la AEMET no tiene estación de monitoreo?

Tal y como se avanzó en el Capítulo 4, parece lógico pensar que aquellos puntos que estén cerca tendrán valores similares. Por tanto, tomemos ventaja de las propiedades de la dependencia espacial y utilicemos un método de interpolación sencillo, en este caso un método determinista, la Distancia Inversa Ponderada, comúnmente conocido por su acrónimo inglés IDW (Inverse distance weighted), el cual es uno de los métodos más simples para llevar para llevar a cabo una interpolación espacial.

En este tipo de análisis espacial, es crucial que el CRS sea el apropiado. En este caso, ya se definió el CRS como un CRS geográfico (es decir, usando coordenadas de longitud y latitud). Sin embargo, para el ejercicio de interpolación es más adecuado usar un CRS local (que provoca pocas deformaciones en la proyección de España) y en alguna unidad de distancia, como metros (ya se vio en la Sección XXXX que en los CRS geográficos las unidades son grados).

**Ejercicio 5.12** (Obtención de CRS sugerido para un conjunto de datos). Utilice el paquete `crssuggest` para observar los CRS sugeridos y, si es necesario, transforme la proyección de los datos.

```
library(crsuggest)

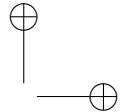
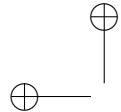
sugiere <- suggest_crs(tmin_8enero, units = "m", limit = 5)

# Usamos la sugerencia del paquete
crs_sugerido <- st_crs(sugiere[1, ]$crs_proj4) # Madrid

esp3 <- st_transform(esp2, crs_sugerido)
tmin_8enero3 <- st_transform(tmin_8enero, crs_sugerido)
```

Una vez solucionado el problema de las proyecciones, antes de llevar a cabo la interpolación, es necesario generar una malla que representará las celdas de las que queremos obtener el valor interpolado. Dado que hemos proyectado nuestros datos a un CRS cuya unidad son los metros, podemos definir el tamaño de cada celda en metros cuadrados. En este caso vamos a usar celdas de 100 kms cuadrados ( $10 \times 10$  kms).

**Ejercicio 5.13** (Creación y representación de una malla de interpolación). Genere un grid y llámelo `malla_sf` (puede fijar una semilla si lo desea) y grafique la superficie construida.



```
# Generación de la superficie a interpolar
set.seed(9876) # Aseguramos que el grid generado siempre es igual

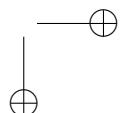
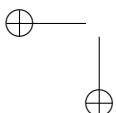
malla_sf <- st_make_grid(
  esp3,
  cellsize = 8000
)

# Representación de la superficie construida añadiendo el contorno
# de España
ggplot(esp3) +
  geom_sf() +
  geom_sf(
    data = malla_sf,
    size = 0.1,
    col = "red", alpha = 1,
    fill = NA
  ) +
  geom_sf(
    data = tmin_8enero3,
    aes(fill = "AEMET Stations"), size = 4, shape = 21,
    color = "blue"
  ) +
  scale_fill_manual(values = adjustcolor("blue", alpha.f = 0.2)) +
  theme_void() +
  theme(legend.position = "bottom")
```

Se puede observar claramente cada una de las celdas que se han creado. La interpolación asignará un valor a cada uno de ellas.

A continuación podemos llevar a cabo la interpolación usando el paquete **gstat**. Además, en lugar de celdas (polígonos) es necesario usar puntos en la interpolación. Calcularemos, por tanto, un punto representativo de cada celda creada en la superficie anterior **malla\_sf**, el centroide, que es el punto resultante de realizar la media aritmética de las coordenadas de los puntos que componen los lados de cada celda.

**Ejercicio 5.14** (Interpolación a través de la Distancia Inversa Ponderada). Calcule los centroides de los polígonos de la malla construida en el Ejercicio 5.13 con la función **st\_centroide** y realice una interpolación de la variable temperatura mínima **tmin** para el día 8 de enero de 2021 con el método IDW usando la librería **gstat** y la función **idw**. Guarde el resultado obtenido en un objeto ll-



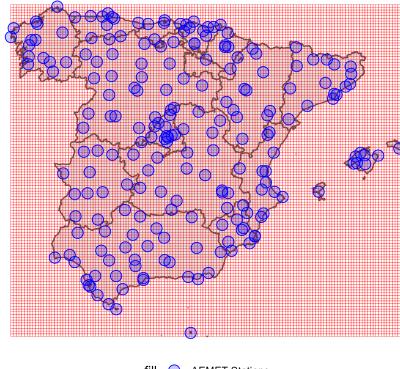


Figura 5.6: Malla de puntos para interpolación

mado `tmin_idw`. Utilice la función `help(idw)` si requiere información sobre cómo introducir los parámetros en la función.

Examine la información del objeto `tmin_idw`, a través de la función `head`,

```
# Calculamos una malla con centroides
malla_sf_cent <- st_centroid(malla_sf, of_largest_polygon = TRUE)

library(gstat)
tmin_idw <- idw(
  # Indicamos la variable que queremos interpolar
  tmin ~ 1,
  # Indicamos el conjunto de datos donde está la variable
  tmin_8enero3,
  # Indicamos la malla de destino, en sf
  newdata = malla_sf_cent,
  idp = 2.0 # Especifica la potencia de la IDW
)
#> [inverse distance weighted interpolation]
head(tmin_idw)
#> Simple feature collection with 6 features and 2 fields
#> Geometry type: POINT
#> Dimension: XY
#> Bounding box: xmin: 146290.9 ymin: 69457.31 xmax: 186290.9
#> ymax: 69457.31
#> CRS: +proj=lcc +lat_1=40 +lat_0=40 +lon_0=0
#> +k_0=0.9988085293 +x_0=600000 +y_0=600000 +a=6378298.3
#> +rf=294.73 +pm=madrid +units=m +no_defs
```

```
#>   var1.pred var1.var           geometry
#> 1 2.518621      NA POINT (146290.9 69457.31)
#> 2 2.586930      NA POINT (154290.9 69457.31)
#> 3 2.656846      NA POINT (162290.9 69457.31)
#> 4 2.728395      NA POINT (170290.9 69457.31)
#> 5 2.801600      NA POINT (178290.9 69457.31)
#> 6 2.876486      NA POINT (186290.9 69457.31)
```

Un tipo de mapas muy utilizado cuando se trabaja con datos espaciales son los mapas de contorno. Es muy visual y ayuda a interpretar el mapa interpolado, añadir unas líneas de contorno al mapa interpolado.

**Ejercicio 5.15** (Mapa de interpolación y contorno con ‘raster’). Represente los valores interpolados, `tmin_idw`, y añada unas líneas de contorno. Utilice el paquete `raster` para convertir el objeto interpolado a pixeles.

```
# Convertimos de sf a SpatialPixels
# Esto funciona porque nuestros puntos sf están espaciados
# regularmente

tmin_pixels <- tmin_idw %>%
  as("Spatial") %>%
  as("SpatialPixels")

library(raster)
# Creamos un raster de nuestros pixels
rast_esp <- raster(tmin_pixels)

# Transferimos valores del objeto sf al raster
rast_esp2 <- rasterize(
  tmin_idw,
  rast_esp,
  field = "var1.pred", ## valores de predicción idw
  fun = mean
)

# Además, podemos recortar el raster a la forma de España
rast_esp_mask <- mask(rast_esp2, esp3)

plot(rast_esp_mask, col = colores)
contour(rast_esp2, add = TRUE)
```

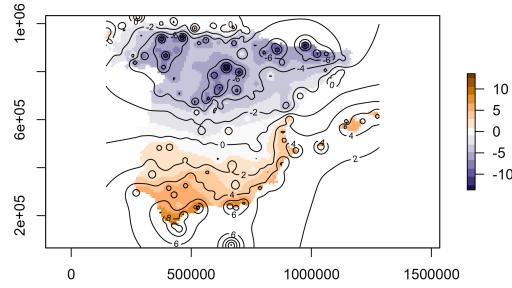


Figura 5.7: Mapa raster con lineas de nivel

**Ejercicio 5.16** (Mapa de interpolación con ‘ggplot’). Repita el mapa de 5.15 usando ggplot2 y la función geom\_contour\_filled.

```
# Creo una tabla para geom contour
coordenadas <- st_coordinates(tmin_idw)
valor <- tmin_idw$var1.pred

idw_df <- data.frame(
  # Necesitamos redondear las coordenadas
  latitud = round(coordenadas[, 2], 6),
  longitud = round(coordenadas[, 1], 6),
  tmin = valor
)

ggplot() +
  geom_contour_filled(
    data = idw_df,
    aes(x = longitud, y = latitud, z = tmin),
    na.rm = TRUE,
    breaks = cortes
  ) +
  # Reajustamos la escala de colores
  scale_fill_manual(values = colores) +
  # CCAA
```

```
geom_sf(data = esp3, fill = NA) +  
  theme_minimal() +  
  theme(axis.title = element_blank()) +  
  labs(  
    fill = "Temp. (°)",  
    # title = "Temperatura mínima interpolada",  
    # subtitle = "8 de Enero 2021",  
    # caption = "Datos: AEMET"  
)
```

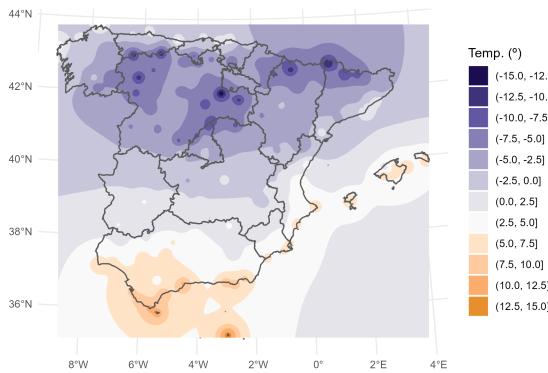


Figura 5.8: Temperatura mínima interpolada. 8 de Enero 2021.

## 5.2. Caso 2. Distribución espacial de la renta media por municipios

\*\* Objetivos de aprendizaje \*\*

Esta sección presenta un caso de uso en el que aprenderemos a realizar las siguientes tareas básicas:

- Importar datos tabulares y datos espaciales.
- Realizar un tratamiento de limpieza de datos y cruzar tablas.
- Hacer mapas temáticos. Aprenderemos también algunas nociones básicas sobre cómo crear diferentes clases para un conjunto de datos continuo.

Para ello, partiremos de dos ficheros:

1. Fichero `renta_municipio.csv`: Este fichero contiene información de la Renta Neta per cápita por municipios (en euros), distritos y secciones censales. Esta información se ha extraído del Atlas de distribución de renta de los hogares proporcionado por el INE, y ha sido tratado previamente para adaptar la información al presente ejercicio.
2. Fichero `municipios.gpkg`: Es un fichero que contiene datos espaciales (polígonos) de los municipios en España en el año 2019. Se ha extraído del Instituto Geográfico Nacional (IGN) usando el paquete `mapSpain`.

El primer paso en cualquier tipo de análisis de datos es importar los datos al software de tratamiento (en nuestro caso, R) y analizarlos para conocer el tipo de información que contiene.

**Ejercicio 5.17** (Importación y análisis del los datos objeto de estudio). Importe el fichero de datos `renta_municipio.csv` y `municipios.gpkg` y guárdelo en un objeto llamado `renta` y `munis`, respectivamente. Observe la información que contienen. Puede ayudarse de la función `head`. Use las librerías oportunas para importar los datos en los distintos formatos.

```
# Usaremos paquetes del tidyverse
library(dplyr)
library(readr)

renta <- read_csv("data/renta_municipio.csv", na = ".")
```

```
library(sf)
munis <- st_read("data/municipios.gpkg", quiet = TRUE)
```

```
head(renta)
#> # A tibble: 6 x 6
#>   Unidad          `2019` `2018` `2017` `2016`
#>   <chr>          <dbl>  <dbl>  <dbl>  <dbl>
#> 1 44001 Ababuj      NA     NA     NA     NA
#> 2 4400101 Ababuj distrito 01      NA     NA     NA
#> 3 4400101001 Ababuj sección 01001    NA     NA     NA
```

```
#> 4 40001 Abades           11429 10731 10314 9816
  ↵ 9904
#> 5 4000101 Abades distrito 01      11429 10731 10314 9816
  ↵ 9904
#> 6 4000101001 Abades sección 01001 11429 10731 10314 9816
  ↵ 9904
```

Se puede comprobar que tenemos información para el periodo 2015-2019. Además, la columna **Unidad** contiene un literal con el municipio o sección correspondiente.

```
head(munis)
#> Simple feature collection with 6 features and 7 fields
#> Geometry type: MULTIPOLYGON
#> Dimension:      XY
#> Bounding box:  xmin: -3.140179 ymin: 36.73817 xmax: -2.057058
  ↵ ymax: 37.54579
#> Geodetic CRS:  ETRS89
#>   codaauto ine.ccaa.name cpro ine.prov.name cmun      name
  ↵  LAU_CODE
#> 1    01 Andalucía 04      Almería 001      Abla
  ↵ 04001
#> 2    01 Andalucía 04      Almería 002 Abrucena
  ↵ 04002
#> 3    01 Andalucía 04      Almería 003      Adra
  ↵ 04003
#> 4    01 Andalucía 04      Almería 004 Albánchez
  ↵ 04004
#> 5    01 Andalucía 04      Almería 005 Alboloduy
  ↵ 04005
#> 6    01 Andalucía 04      Almería 006      Albox
  ↵ 04006
#>                                     geom
#> 1 MULTIPOLYGON (((-2.775594 3...
#> 2 MULTIPOLYGON (((-2.787566 3...
#> 3 MULTIPOLYGON (((-3.051988 3...
#> 4 MULTIPOLYGON (((-2.181086 3...
#> 5 MULTIPOLYGON (((-2.572442 3...
#> 6 MULTIPOLYGON (((-2.128106 3...
```

El objeto **munis** contiene Polígonos y varias columnas, entre ellas dos especialmente relevantes: **cpro** y **cmun**, que corresponden a los códigos de provincia y de municipio

respectivamente. Podemos comprobar que este código también se encuentra en el dataset `renta`.

**Ejercicio 5.18** (Comprobación de campos en común para un municipio: Noblejas). Para comprobar que efectivamente disponemos de dos campos en comun en los ficheros, que serán de vital importancia para posteriormente unirlos, se selecciona un municipio al azar, el municipio de Noblejas en la provincia de Toledo y comprobamos.

```
# Miro un municipio: Noblejas

renta[grep("Noblejas", renta$Unidad), ]
#> # A tibble: 5 x 6
#>   Unidad          `2019` `2018` `2017` `2016` 
#>   <chr>          <dbl>  <dbl>  <dbl>  <dbl>
#> 1 45115 Noblejas      10591  10314  9751  9484
#> 2 9124
#> 3 4511501 Noblejas distrito 01      11039  10717  10135  9711
#> 4 9386
#> 5 3 4511501001 Noblejas sección 01001  11039  10717  10135  9711
#> 6 9386
#> 7 4 4511502 Noblejas distrito 02      10276  10029  9475  9319
#> 8 8938
#> 9 8938

munis[grep("Noblejas", munis$name), c("name", "cpro", "cmun")]
#> Simple feature collection with 1 feature and 3 fields
#> Geometry type: MULTIPOLYGON
#> Dimension:      XY
#> Bounding box:  xmin: -3.489824 ymin: 39.93003 xmax: -3.372611
#>                           ymax: 40.05017
#> Geodetic CRS:  ETRS89
#>           name cpro cmun                      geom
#> 1 4985 Noblejas    45   115 MULTIPOLYGON (((-3.44681 40...
```

En el caso de Noblejas, el código completo es 45115. Sin embargo, en el caso de la tabla `renta`, debemos extraer ese valor del literal. Para ello debemos manipular la columna y extraer la primera palabra de la columna `Unidad`:

```
# Creo una función y la aplico a toda la columna
extrae_codigo <- function(x) {
  unlist(strsplit(x, " ")) [1]
}

renta$codigo_ine <- sapply(as.character(renta$Unidad),
  extrae_codigo)

head(renta[c("Unidad", "codigo_ine")])
#> # A tibble: 6 x 2
#>   Unidad                 codigo_ine
#>   <chr>                  <chr>
#> 1 Ababuj                44001
#> 2 Ababuj distrito 01    4400101
#> 3 Ababuj sección 01001  4400101001
#> 4 Abades                40001
#> 5 Abades distrito 01    4000101
#> 6 Abades sección 01001  4000101001
```

Ahora, es necesario crear la misma variable en `munis` para poder realizar el cruce:

```
munis$codigo_ine <- paste0(munis$cpro, munis$cmun)

head(munis[, c("name", "codigo_ine")])
#> Simple feature collection with 6 features and 2 fields
#> Geometry type: MULTIPOLYGON
#> Dimension:      XY
#> Bounding box:  xmin: -3.140179 ymin: 36.73817 xmax: -2.057058
#>                         ymax: 37.54579
#> Geodetic CRS:  ETRS89
#>           name codigo_ine          geom
#> 1     Abla    04001 MULTIPOLYGON (((-2.775594 3...
#> 2 Abrucena  04002 MULTIPOLYGON (((-2.787566 3...
#> 3     Adra    04003 MULTIPOLYGON (((-3.051988 3...
#> 4 Albanchez 04004 MULTIPOLYGON (((-2.181086 3...
#> 5 Alboloduy  04005 MULTIPOLYGON (((-2.572442 3...
#> 6     Albox   04006 MULTIPOLYGON (((-2.128106 3...
```

Ya estamos listos para realizar el cruce. Además, seleccionaremos sólo las columnas que vamos a usar, en este caso la del año 2019.

DIEGO, letfjoin está puesto sin guión porque da error en latex y no he sido capaz de solucionarlo. Es un mal menor

**Ejercicio 5.19** (Unión de objetos renta y mapas con la función ‘leftjoin’). Realice la unión de los objetos con la función `left_join` y seleccione las variables `name`, `cpro`, `cmun`, `2019`. Guarde el resultado obtenido en un nuevo objeto llamado `munis_renta`.

```
munis_renta <- munis %>%
  left_join(renta) %>%
  dplyr::select(name, cpro, cmun, `2019`)
```

Cuando crucemos datos espaciales con datos no espaciales en R, es importante que el primer dataset sea el que contiene los datos espaciales. Esto es así porque el objeto resultante “hereda” la clase del primer objeto.

¿Qué ocurre si realizáramos el proceso poniendo los datos espaciales en el lado derecho del join? A modo de ejemplo, si realizáramos el proceso poniendo los datos espaciales en el lado derecho del join, los datos finales no serán espaciales:

```
# Miramos la clase de munis_renta

class(munis_renta)
#> [1] "sf"           "data.frame"

# Es un sf, por tanto espacial

# ¿Qué pasa si realizamos el cruce de la otra manera?
renta %>%
  left_join(munis) %>%
  dplyr::select(name, cpro, cmun, `2019`) %>%
  class()
#> [1] "tbl_df"        "tbl"           "data.frame"
```

El resultado es un tibble o data.frame, ¡pero no es espacial!

Una vez que tenemos los datos unidos podemos realizar algunos análisis básicos, como la realización de un histograma.

**Ejercicio 5.20** (Histograma de la variable Renta neta media por persona (€)). A través de un histograma represente la distribución de la variable Renta neta media por persona del objeto `munis_renta` para el año 2019.

```
library(ggplot2)

munis_renta %>%
  ggplot(aes(x = `2019`)) +
  geom_histogram(color = "darkblue", fill = "lightblue") +
  scale_x_continuous(labels = scales::label_number_auto()) +
  scale_y_continuous(labels = scales::label_percent()) +
  labs(
    y = "",
    x = "Renta neta media por persona (€)"
  )
```

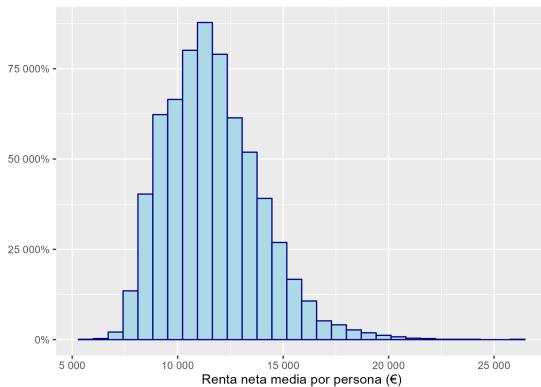
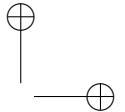
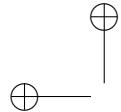


Figura 5.9: Histograma de la variable Renta neta media por persona (€) en 2019

Se puede observar que la renta presenta una distribución Gamma con un gran número de municipios concentrados en zonas medianas de renta y pocos municipios en tramos de rentas altas. Como se verá en el Ejercicio XXX, esta distribución va a afectar a la información que transmite el mapa.

::: {.exercise #ex20 name=“ Mapa de coropletas de la distribución de la renta” }  
Realice un mapa de coropletas mostrando la distribución de la renta usando los valores brutos de renta sin modificar :::

```
ggplot(munis_renta) +
  # Usamos geom_sf, y como aes() lo que queremos mostrar, en este
  # caso, el
  # color del polígono representa la renta. Vamos a retirar los
  # bordes con
```



```
# color = NA
geom_sf(aes(fill = `2019`), color = NA) +
theme_minimal() +
scale_fill_continuous(labels = scales::label_number(
  big.mark = ".",
  decimal.mark = ",",
  suffix = " €"
)) # +
```

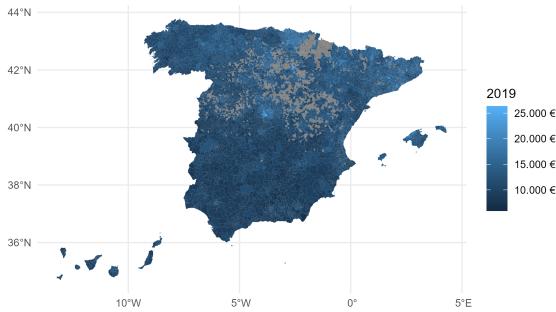


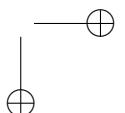
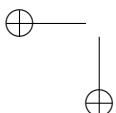
Figura 5.10: Renta neta media por persona en España (2019)

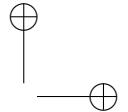
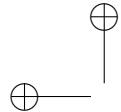
```
# labs(
#   title = "Renta neta media por persona",
#   caption = "Datos: INE"
# )
```

Este primer mapa no es demasiado informativo, por los siguientes motivos:

- Existe una serie de municipios para los que no tenemos datos.
- La escala de color no es la más adecuada.
- Dada la distribución de los datos, puede ser adecuado crear grupos de renta para que el mapa sea más interpretable.

**Ejercicio 5.21** (Adecuacion de los datos al la visualización). Elimine los municipios sin datos y a cambie la escala de color para ver si mejora la visualización de la variable Renta neta media por persona.





```
munis_renta_clean <- munis_renta %>% filter(!is.na(`2019`))

ggplot(munis_renta_clean) +
  geom_sf(aes(fill = `2019`), color = NA) +
  # Cambiamos la paleta de colores, vamos a usar una paleta
  # denominada Inferno,
  # ya incluida en base R con hcl.colors

  # Como son datos continuos, puedo usar Inferno
  scale_fill_gradientn(
    colours = hcl.colors(20, "Inferno", rev = TRUE),
    labels = scales::label_number(
      big.mark = ".",
      decimal.mark = ",",
      suffix = " €"
    )
  ) +
  theme_minimal()
```

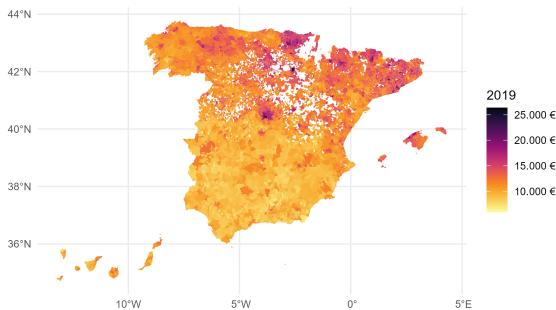
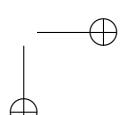
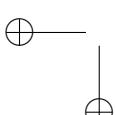


Figura 5.11: Renta neta media por persona en España (2019)

```
# + labs(
#   title = "Renta neta media por persona",
#   caption = "Datos: INE"
# )
```

Este mapa proporciona algo más de información, y parece intuirse que las rentas



más altas se encuentran en zonas de País Vasco, Madrid y Cataluña. Sin embargo, el hecho de que la distribución de los datos no sea normal está afectando a la visualización.

Para intentar atajar este problema, se puede dividir los datos en clases, por ejemplo, cuartiles o deciles. Existen varios métodos de clasificación de datos, que en R se encuentran implementados en el paquete `classInt`. A continuación se van a plantear diversos métodos de clasificación y se observará cómo la “historia” que cuenta el mapa varía en función de dichas clases. Se proponen siguientes métodos de clasificación:

- **El método de deciles:** consiste en crear 10 categorías incluyendo el mismo número de registros en cada una de ellas.
- **El método de intervalos equivalentes:** divide el rango de valores en un número de grupos definido. La distancia de todos los intervalos es idéntica, por lo que este método no tiene en cuenta la distribución de los registros.
- **El método de Fisher-Jenks:** desarrollado específicamente para la clasificación de datos espaciales y su visualización en mapas. Produce agrupaciones de tal manera que los datos de cada grupo son “cercanas” entre sí y sustancialmente distintas de los valores de otros grupos.

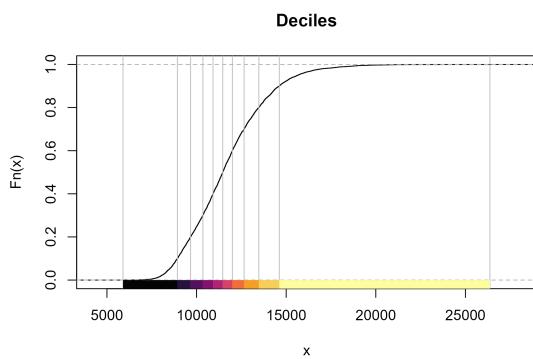
**Ejercicio 5.22** (División de los datos en clases). Utilice la función `classIntervals` del paquete `classInt` y cambie el parámetro `style` para obtener los métodos de clasificación: Deciles, tramos de Renta equidistantes y Fisher and Jenks.

Realice un `plot` de las clases obtenidas y compárelas.

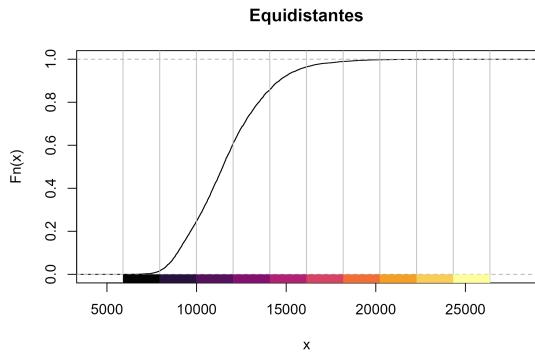
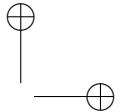
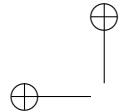
```
library(classInt)

# División en deciles
deciles <- classIntervals(munis_renta_clean$`2019`,
  style = "quantile", n = 10
)
deciles
#> style: quantile
#>      [5898,8935.6)  [8935.6,9662.2)  [9662.2,10352.8)
#>      [10352.8,10918)
#>          656           656           655
#>          654
#>      [10918,11462)   [11462,11998.6)  [11998.6,12651.4)
#>      [12651.4,13475.8)
#>          655           658           656
#>          655
```

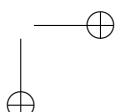
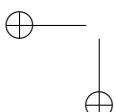
```
#> [13475.8,14618.4]  [14618.4,26367]
#>          656          656
plot(deciles, pal = hcl.colors(20, "Inferno"), main = "Deciles")
```

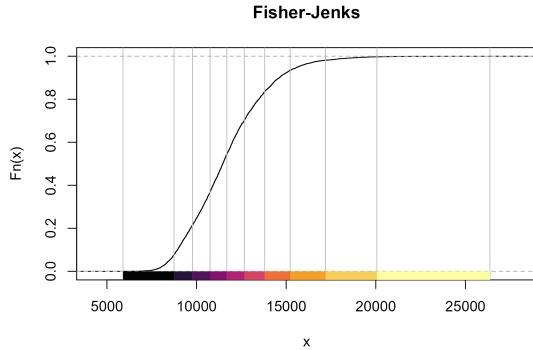


```
# Tramos equidistantes en términos de renta
equal <- classIntervals(munis_renta_clean$`2019`,
  style = "equal", n = 10
)
equal
#> style: equal
#>      [5898,7944.9]  [7944.9,9991.8]  [9991.8,12038.7)
#>      [12038.7,14085.6)
#>          103          1510          2374
#>          1637
#> [14085.6,16132.5) [16132.5,18179.4) [18179.4,20226.3)
#>      [20226.3,22273.2)
#>          702          161          52
#>          14
#> [22273.2,24320.1)  [24320.1,26367]
#>          3            1
plot(equal, pal = hcl.colors(20, "Inferno"), main =
  "Equidistantes")
```



```
fisher <- classIntervals(munis_renta_clean$`2019`,  
  style = "fisher",  
  # Fuerzo para mejorar la comparación entre métodos  
  n = 10  
)  
fisher  
#> style: fisher  
#>      [5898,8743]      [8743,9770.5]      [9770.5,10754)  
#>      [10754,11689)  
#>      505                  904                  1005  
#>      1159  
#>      [11689,12668)      [12668,13803)      [13803,15222.5)  
#>      [15222.5,17196.5)  
#>      1032                  874                  651  
#>      305  
#>      [17196.5,20063.5)      [20063.5,26367]  
#>      103                  19  
plot(fisher,  
  pal = hcl.colors(20, "Inferno"),  
  main = "Fisher-Jenks"  
)
```





Se puede observar lo siguiente:

- El último decil de renta se corresponde a un rango de entre 15.000 y 25.000 €.
- El método por deciles proporciona unos grupos con valores de renta muy parecidos entre sí en los valores medios. Esto es debido a la propia distribución de la variable.
- El método de rangos equidistantes proporciona algunos grupos con un número muy reducido de municipios.
- El método de Fisher-Jenks puede proporcionar unas clases con unos rangos más apropiados para los tramos altos de renta.

**Ejercicio 5.23** (Representación de los mapas según las clases obtenidas). Realice tres mapas distintos, creando clases de renta según cada uno de los métodos anteriormente mostrados y coméntelos.

### Deciles

```
# Extracción de los valores de corte
breaks_d <- deciles$brks

# Creación de etiquetas básicas para cada clase
# Creación de una función específica para crear etiquetas
# formateadas
label_fun <- function(x) {
  l <- length(x)
  eur <- paste0(prettyNum(round(x, 0),
    big.mark = ","),
    "%")
  if (l > 1) {
    eur <- paste0(eur, " - ",
      prettyNum(round(x[l], 0),
        big.mark = ","),
      "%"))
  }
  return(eur)
}
```

```

decimal.mark = ",",
big.mark = "."
), " €")

labels <- paste(eur[-1], "-", eur[-1])
labels[1] <- paste("<", eur[1])
labels[1 - 1] <- paste(">", eur[1 - 1])
return(labels)
}

labels_d <- label_fun(breaks_d)

munis_renta_clean$Deciles <- cut(munis_renta_clean$`2019`,
breaks = breaks_d,
labels = labels_d,
include.lowest = TRUE
)

ggplot(munis_renta_clean) +
# Cambio la variable a representar para crear el mapa
geom_sf(aes(fill = Deciles), color = NA) +
# Cambio el scale, ya no es continua
scale_fill_manual(values = hcl.colors(length(labels_d),
"Inferno",
rev = TRUE
)) +
theme_minimal() # +
# labs(
#   title = "Renta neta media por persona",
#   caption = "Datos: INE"
# )

```

El mapa de la Fig. 5.12 ya permite observar patrones geográficos, donde se ve una clara diferencia entre las Comunidades Autónomas del Norte y las del Sur. Veamos una representación distinta usando otras clases diferentes

```

breaks_e <- equal$brks
labels_e <- label_fun(breaks_e)

```

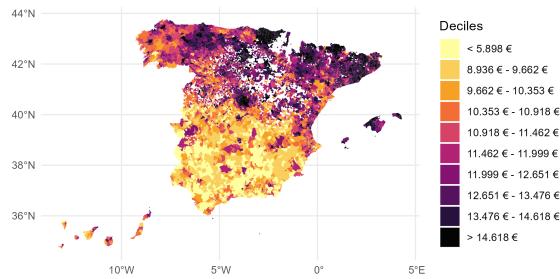


Figura 5.12: Mapa por deciles de renta media por persona (2019)

```

munis_renta_clean$Equal <- cut(munis_renta_clean$`2019`,
  breaks = breaks_e,
  labels = labels_e,
  include.lowest = TRUE
)

ggplot(munis_renta_clean) +
  # Cambiamos la variable que usamos para crear el mapa
  geom_sf(aes(fill = Equal), color = NA) +
  scale_fill_manual(values = hcl.colors(length(labels_e),
    "Inferno",
    rev = TRUE
  )) +
  theme_minimal() # +
  # labs(
  #   title = "Renta neta media por persona",
  #   caption = "Datos: INE"
  # )

```

El mapa de la Fig. 5.12, sin embargo, se parece más al mapa de la Fig. 5.11 con los datos sin clasificar, donde el peso visual se concentra más bien en los municipios con rentas mucho más altas que el resto (por encima de 18.000 €).

Véase a continuación el mismo mapa usando la clasificación Fisher-Jenks:

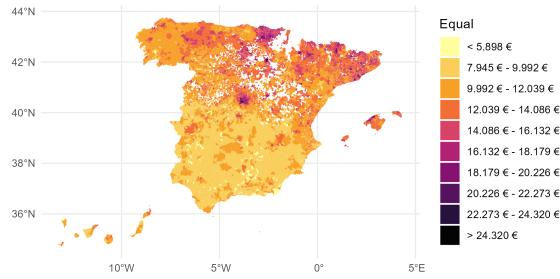


Figura 5.13: Mapa por tramos de renta equidistantes de renta media por persona (2019)

```

breaks_f <- fisher$brks
labels_f <- label_fun(breaks_f)

munis_renta_clean$`Fisher-Jenks` <- cut(munis_renta_clean$`2019`,
  breaks = breaks_f,
  labels = labels_f,
  include.lowest = TRUE
)

ggplot(munis_renta_clean) +
  # Cambiamos la variable que usamos para crear el mapa
  geom_sf(aes(fill = `Fisher-Jenks`), color = NA) +
  scale_fill_manual(values = hcl.colors(length(labels_f),
    "Inferno",
    rev = TRUE
)) +
  theme_minimal() +
  labs(
    title = "Renta neta media por persona",
    caption = "Datos: INE"
)

```

En el mapa de la Fig. 5.12 se puede observar de una manera más clara un cluster adicional de renta en la zona de Asturias y el norte de León. Además, gracias a la

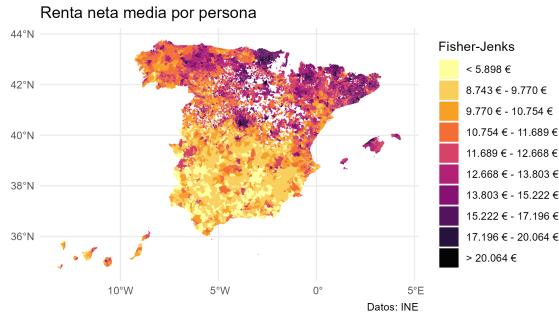


Figura 5.14: Mapa por tramos según Fisher-Jenks

escala de colores puede intuirse que este clúster de renta no presenta valores tan altos como los observados en País Vasco o Madrid.

En conclusión, en el momento de realizar una visualización de datos es importante conocer el dato a representar, así como entender algunas propiedades básicas de la distribución subyacente. También se ha podido observar que hay ciertas decisiones estéticas (datos continuos vs. agrupados, escala de colores) que tienen una influencia significativa en cómo se percibe la información representada. Es responsabilidad del investigado encargado de crear de la visualización el conocer todos estos factores y aplicarlos de manera conveniente.

### 5.3. Caso 3. Distribución espacial de los delitos cometidos en la ciudad de Valencia en el año 2010.

Las técnicas de análisis de patrones de puntos analizan la distribución de eventos geolocalizados que surgen al azar. La diferencia fundamental con otros análisis que comprenden también el uso de localizaciones (como las temperaturas mínimas medidas por estaciones meteorológicas) es que en este caso los puntos representan eventos conocidos y aleatorios (por ejemplo, los delitos ocurridos en una ciudad, accidentes de tráfico o incendios en una región). A diferencia de otros eventos, como el ejemplo de las temperaturas mínimas, la ausencia de datos no se debe a la ausencia de medición (es decir, no existe una estación meteorológica en ese lugar), si no a que no se ha producido el evento en dicha localización.

**Objetivo de aprendizaje:**

El alumno debe ser capaz de conocer los datos de tipo patrones de punto, identificarlos y representarlos adecuadamente.

- describir los ficheros. Hay que generar uno que sólo tenga 2010 y los datos que utilizamos. lo llamamos igual y vale todo

**Ejercicio 5.24** (Representación de la información sesgada). El presente análisis se va a realizar empleando RStudio, por lo que empezaremos abriendo el programa y creando un nuevo script de R en *Proyecto>File>New File>R script*.

\*\* habría que poner esto al inicio de las 3 prácticas. comprobar.

*Tarea 2: Importamos y describimos los datos objeto de estudio*

El primer paso consiste en importar la base de datos de crímenes en la ciudad de Valencia. El archivo está en formato csv, por lo que usaremos el paquete `readr` para importar los datos:

```
library(readr)
library(dplyr)

# En este caso el archivo está en la carpeta "data" de nuestro
# proyecto
crimen <- read_csv("data/crime-data-Valencia.csv")

summary(crimen)
#>      crime_id          crime_date        crime_time
#>   <dbl>        <date>        <time>
#> Length:90247    Length:90247    Length:90247
#>   <dbl>        <dbl>        <dbl>
#> Class :character  Class :character  Class1:hms      Class
#>   <dbl>        <dbl>        <dbl>
#>   <dbl>        <dbl>        <dbl>
#>   <dbl>        <dbl>        <dbl>
#> Mode  :character  Mode  :character  Class2: difftime Mode
#>   <dbl>        <dbl>        <dbl>
#>   <dbl>        <dbl>        <dbl>
#>
#>
#>
#>
#>
#>
#>
#>
#>
#>      muni          year       month       week
#>   <dbl>        <dbl>        <dbl>        <dbl>
#> Length:90247    Min.     :2010    Min.     : 1.000  Min.     :
#>   <dbl>        <dbl>        <dbl>        <dbl>
```

```
#> Class :character 1st Qu.:2013 1st Qu.: 4.000 1st
  ↵ Qu.:14.00
#> Mode :character Median :2016 Median : 7.000 Median
  ↵ :27.00
#>           Mean :2015 Mean : 6.537 Mean
  ↵ :26.65
#>           3rd Qu.:2018 3rd Qu.: 9.000 3rd
  ↵ Qu.:38.00
#>           Max. :2020 Max. :12.000 Max.
  ↵ :53.00
#>   day week_day week_day_name crime_hour
  ↵
#> Min. : 1.0 Min. :1.000 Length:90247 Min. :
  ↵ 0.00
#> 1st Qu.: 96.0 1st Qu.:3.000 Class :character 1st Qu.:
  ↵ 5.00
#> Median :187.0 Median :5.000 Mode :character Median
  ↵ :14.00
#> Mean :183.6 Mean :4.323 Mean
  ↵ :12.52
#> 3rd Qu.:266.0 3rd Qu.:6.000 3rd
  ↵ Qu.:19.00
#> Max. :366.0 Max. :7.000 Max.
  ↵ :23.00
#>   crime_lon crime_lat atm_dist
  ↵ bank_dist
#> Min. :-0.4296 Min. :39.42 Min. : 1.549 Min. :
  ↵ 1.713
#> 1st Qu.:-0.3883 1st Qu.:39.46 1st Qu.: 380.535 1st Qu.:
  ↵ 133.908
#> Median :-0.3749 Median :39.47 Median : 630.867 Median :
  ↵ 228.003
#> Mean :-0.3724 Mean :39.47 Mean : 790.228 Mean :
  ↵ 305.315
#> 3rd Qu.:-0.3593 3rd Qu.:39.48 3rd Qu.: 986.897 3rd Qu.:
  ↵ 384.706
#> Max. :-0.3208 Max. :39.53 Max. :4838.847 Max.
  ↵ :3411.537
#>   bar_dist cafe_dist industrial_dist
  ↵ market_dist
#> Min. : 0.489 Min. : 1.342 Min. : 0.1 Min. :
  ↵ 1.821
```

```
#> 1st Qu.: 149.879   1st Qu.: 133.274   1st Qu.: 268.4   1st Qu.:
#>   ↵ 487.874
#> Median : 292.246   Median : 264.504   Median : 509.6   Median :
#>   ↵ 869.516
#> Mean    : 392.072   Mean    : 360.275   Mean    : 675.2   Mean
#>   ↵ :1008.414
#> 3rd Qu.: 507.701   3rd Qu.: 462.982   3rd Qu.: 909.9   3rd
#>   ↵ Qu.:1263.405
#> Max.   :4352.380   Max.   :4237.915   Max.   :4703.9   Max.
#>   ↵ :4855.407
#> nightclub_dist      police_dist        pub_dist
#> restaurant_dist
#> Min.   : 1.189   Min.   : 0.688   Min.   : 1.489   Min.
#>   ↵ : 0.427
#> 1st Qu.: 468.164   1st Qu.: 455.810   1st Qu.: 190.150   1st
#>   ↵ Qu.: 73.401
#> Median : 810.185   Median : 706.094   Median : 383.330   Median
#>   ↵ :147.797
#> Mean    : 930.501   Mean    : 875.246   Mean    : 496.472   Mean
#>   ↵ :223.875
#> 3rd Qu.:1256.269   3rd Qu.:1105.625   3rd Qu.: 665.599   3rd
#>   ↵ Qu.: 272.288
#> Max.   :4700.567   Max.   :4765.745   Max.   :4168.869   Max.
#>   ↵ :3746.536
#> taxi_dist          grid_id           grid_lon
#> grid_lat
#> Min.   : 2.441   Min.   : 8.0   Min.   :-0.4288   Min.
#>   ↵ :39.42
#> 1st Qu.: 393.438   1st Qu.:173.0   1st Qu.:-0.3898   1st
#>   ↵ Qu.:39.46
#> Median : 651.644   Median :215.0   Median :-0.3731   Median
#>   ↵ :39.47
#> Mean    : 717.381   Mean    :210.8   Mean    :-0.3724   Mean
#>   ↵ :39.47
#> 3rd Qu.: 931.083   3rd Qu.:248.0   3rd Qu.:-0.3620   3rd
#>   ↵ Qu.:39.48
#> Max.   :4284.239   Max.   :398.0   Max.   :-0.3230   Max.
#>   ↵ :39.53
```

**¿Qué información contiene los datos que se van a analizar?** Este archivo contiene en total 90.247 registros, y proporciona 28 campos asociados a cada registro.

Cuadro 5.2: Crímenes en Valencia; Coordenadas

crime_lon	crime_lat
-0.3826325	39.46332
-0.3906850	39.43517
-0.3747971	39.47927
-0.3992082	39.47982
-0.3596011	39.46740
-0.3552423	39.47461

Entre los campos disponibles, destacamos los campos `crime_lon` y `crime_lat`: Son las coordenadas en las que se produjo el crimen.

**Ejercicio 5.25** (CRS de las coordenadas). Determine el CRS en el que se encuentras las coordenadas del conjunto de datos `crimen`, etiquetas con `crime_lon` y `crime_lat`

Obsérvese que, las coordenadas parecen corresponder con longitudes y latitudes, ya que como se explicó el rango posible de valores es  $[-180, 180]$  (para longitudes) y  $[-90, 90]$  (para latitudes):

**Ejercicio 5.26** (Proyección y representación de localizaciones). Convierta el objeto `crimen` a un objeto `sf` llamado `crimen_sf`, teniendo en cuenta el sistema CRS más adecuado. A modo de recordatorio, el CRS correspondiente a coordenadas geográficas longitud/latitud es **EPSG:4326**.

Una vez proyectadas las coordenadas represéntelas en un mapa. Si lo desea puede usar una imagen de fondo con la función `esp_getTiles` de la librería `mapSpain`.

```
library(sf)
# Objeto sf sin CRS

crimen_sf <- st_as_sf(
  crimen,
  coords = c(
    "crime_lon",
    "crime_lat"
  ),
  crs = st_crs(4326)
)

# Comprobamos con un mapa base
```

```

library(mapSpain)
library(ggplot2)

# Usamos imagen como mapa de fondo
tile <- esp_getTiles(crimen_sf, "IDERioja",
                      zoommin = 1,
                      crop = TRUE
)

ggplot() +
  layer_spatraster(tile) +
  geom_sf(
    data = crimen_sf,
    col = "blue",
    size = 0.3,
    alpha = 0.3
)

```

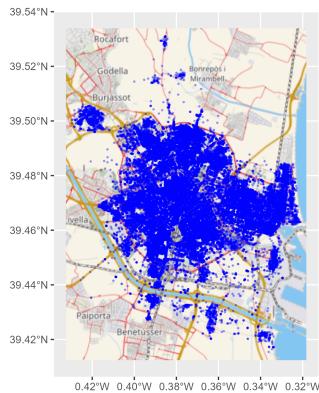


Figura 5.15: Crímenes en Valencia

¿Hay algún patrón en la ocurrencia de crímenes?\*\*

En la Fig. 5.15 podemos intuir ciertos patrones en la ocurrencia de crímenes. Por ejemplo, parecen concentrarse en zonas céntricas y no hay crímenes registrados en la zona del puerto.

Para el siguiente análisis, vamos a analizar el patrón de crímenes del año 2010.

\*\* diego, engancha esto con lo anterior, porque sólo vamos a usar un año, y así

evitamos problemas

```
crimen_2010_sf <- crimen_sf %>%
  filter(
    year == "2010"
  )

ggplot() +
  layer_spatraster(tile) +
  geom_sf(
    data = crimen_2010_sf,
    col = "blue",
    size = 0.3,
    alpha = 0.3
  )
```

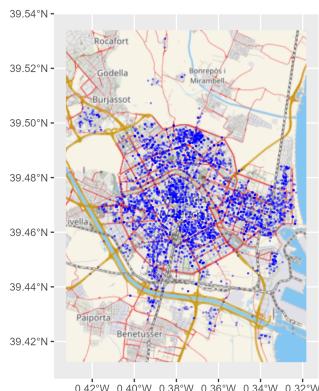
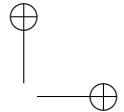
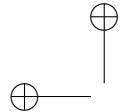


Figura 5.16: Crímenes en Valencia (2010)

El paquete **spatstat** (Baddeley and Turner [2005]) es el paquete de referencia cuando se trabaja con patrones de puntos

Siguiendo el anterior ejemplo, vamos a analizar el patrón de crímenes en el año 2010. Además, en el análisis de patrones de puntos es necesario delimitar la ventana espacial de observación (owin). En este caso será el municipio de Valencia.

**Ejercicio 5.27** (Análisis de patrones con ‘spatstat’). FALTA



```
# Extraigo Valencia con mapSpain

valencia <- esp_get_munic(munic = "Valencia$") %>%
  # Necesito proyectar, en este caso usamos ETRS89-UTM huso 30
  #   ↳ EPSG:25830
  st_transform(25830)

library(spatstat)
# Necesitamos un recinto de observación: owin

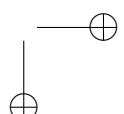
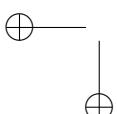
val_owin <- as.owin(valencia)

# Extraemos las coordenadas de los crímenes. Han de estar en el
#   mismo CRS
# que el owin

coords <- crimen_2010_sf %>%
  st_transform(25830) %>%
  st_coordinates()

mydata_ppp <- ppp(
  x = as.numeric(coords[, 1]),
  y = as.numeric(coords[, 2]),
  window = val_owin
)

plot(mydata_ppp)
```



**mydata\_ppp**



ERROR EN EL NOMBRE mydata\_ppp, es por el guión.

**Ejercicio 5.28** (Información del objeto ‘mydatapp’). FALTA

¿Qué información contiene nuestro objeto en formato ppp?

```
summary(mydata_ppp)
#> Planar point pattern: 5332 points
#> Average intensity 3.917619e-05 points per square unit
#>
#> *Pattern contains duplicated points*
#>
#> Coordinates are given to 1 decimal place
#> i.e. rounded to the nearest multiple of 0.1 units
#>
#> Window: polygonal boundary
#> 4 separate polygons (no holes)
#>      vertices     area relative.area
#> polygon 1        4 1977930    0.01450
#> polygon 2       82 131953000   0.97000
#> polygon 3        7  958085    0.00704
#> polygon 4        7 1214060    0.00892
#> enclosing rectangle: [720573.4, 735116.6] x [4351274, 4382995]
#>   units
#>           (14540 x 31720 units)
#> Window area = 136103000 square units
#> Fraction of frame area: 0.295
#>
#> *** 6 illegal points stored in attr("rejects") ***
```

Tarea 4: .

Es importante determinar si los puntos se distribuyen al azar o tienen algún patrón. Por ello, lo primero que haremos será representar el objeto `feb_ppp` y superponer unos cuadrantes para su comportamiento (véase Fig. 5.17).

**Ejercicio 5.29** (Cálculo de la densidad de los delitos en Valencia en 2010 mediante cuadrantes). FALTA

\*\* ESte mapa se puede poner un poco más mono?? Yo ya lo he arreglado un poco

```
## Hallamos los cuadrantes
cuadrante <- quadratcount(mydata_ppp,
  nx = 5,
  ny = 5
)

## Dibujamos el número de crímenes que hay en cada cuadrante
plot(mydata_ppp, pch = "+", main = "", cex = 0.3)
plot(cuadrante, add = TRUE, col = "red", cex = 0.6, lwd = 2)
```



Figura 5.17: Crímenes en Valencia por cuadrantes (2010)

Como se puede apreciar en la Fig. 5.17 hay cuadrantes que registran cero crímenes y otros que registran hasta 2.860 crímenes.

**Ejercicio 5.30** (Estimación de la densidad de los delitos en Valencia en 2010 mediante de patrones de puntos). FALTA

`density {stats}`

```
# Permite ver la variación local de la intensidad, graficando la
# estimación kernel de la intensidad.
densidad <- density(mydata_ppp)

plot(densidad, main = " ")
points(mydata_ppp, pch = "+", cex = 0.5)
```

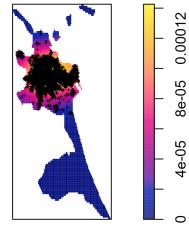


Figura 5.18: Estimación kernel de la intensidad de crímenes en la ciudad de Valencia

En la Fig. 5.18 se muestra la estimación kernel de la densidad de patrones de puntos Los conocimientos teóricos necesarios para llevar a cabo este tipo de estimación superan el ámbito de este manual. El objetivo de este ejemplo es meramente ilustrativo.

## Apéndice A

# Tipos de CRS proyectados

Existen varias familias de proyecciones, que se pueden clasificar de diversas maneras: por tipo de superficie de proyección y por métrica a preservar.

### A.1. Por tipo de superficie de proyección

El proceso de trasladar puntos de una esfera a un plano puede plantearse de manera práctica como el ejercicio de envolver una esfera con una superficie plana (como una hoja de papel) y trasladar los puntos de la esfera de manera lineal al punto de la superficie plana más cercano a ella. La Fig. A.1 muestra estos tres tipos de proyección.

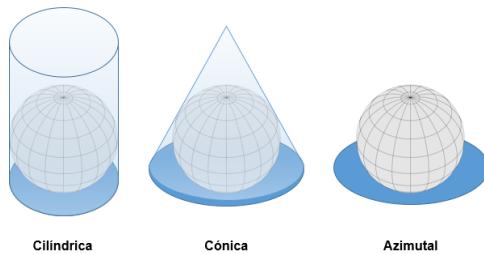


Figura A.1: Tipos de proyección por superficie de proyección

A partir de este ejercicio, se plantean tres posibles soluciones (cilíndrica, cónica y acimutal o planar), dependiendo del tipo de superficie que se use para proyectar.

- **Proyecciones cilíndricas:** Son aquellas proyecciones donde la superficie de proyección conforma un cilindro alrededor de la Tierra. Una de las proyecciones cilíndricas más conocidas es la **proyección de Mercator** (Ver Fig. A.2).

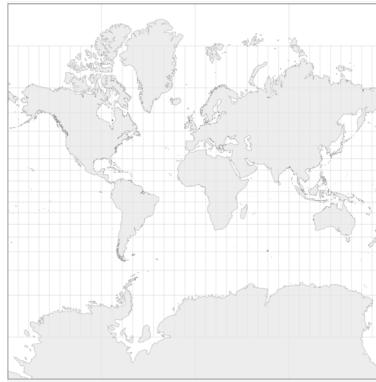


Figura A.2: Proyección Mercator

- **Proyecciones cónicas:** En este tipo de proyecciones, se plantea la superficie de proyección como una forma cónica. Como ejemplo, la **proyección cónica equiáreas de Albers** es una de las proyecciones que más suele usarse en la representación de mapas de América del Norte (Ver Fig. A.3).

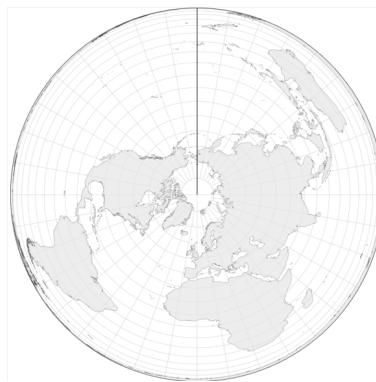


Figura A.3: Proyección cónica equiáreas de Albers

- **Proyecciones acimutales o planares:** En este tipo de proyección se proyecta una porción de la Tierra sobre un plano que es tangente a la misma en

el punto de referencia. Como ejemplos de proyecciones acimutales podemos destacar la **proyección ortográfica** (Ver Fig. A.4).



Figura A.4: Proyección ortogonal

## A.2. Por métrica a preservar

Es importante tener en cuenta que cualquier proyección de la superficie de la Tierra produce distorsiones en una o varias características geográficas. Como ejemplo clásico, la proyección de Mercator produce distorsiones del **tamaño** especialmente en aquellas regiones más cercanas a los polos (Groenlandia, que la proyección de Mercator presenta una área similar a la de África, tiene menor superficie real que Argelia). Otras de las métricas que suele verse distorsionada son la **distancia** entre dos puntos geográficos, la **dirección** o la **forma** de regiones de la Tierra.

A lo largo de la Historia se han desarrollado diversas proyecciones cuyo objetivo es preservar alguna o varias de las propiedades mencionadas anteriormente, sin embargo es importante destacar que **no existe una proyección que sea capaz de preservar todas las métricas a la vez**.

Según la métrica a preservar, las proyecciones se pueden clasificar en:

- **Proyecciones conformales:** Intentan preservar los **ángulos** que se forman en la superficie terrestre. Por ejemplo, la proyección de Mercator representa ángulos rectos en las intersecciones de los paralelos y los meridianos (Ver Fig. A.5).

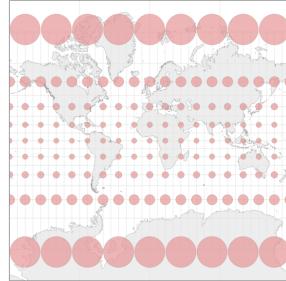


Figura A.5: Ejemplo de proyección conformal: Mercator

- **Proyecciones equivalentes:** Preservan las proporciones de las **áreas**, provocando a su vez deformaciones en el resto de características, como la forma o los ángulos. La proyección acimutal equivalente de Lambers es un tipo de proyección equivalente (Ver Fig. A.6).

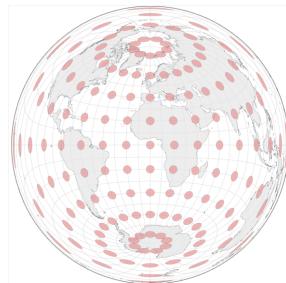


Figura A.6: Ejemplo de proyección equivalente: Proyección acimutal equivalente de Lambers

- **Proyecciones equidistantes:** Preservan la **distancia** entre dos puntos geográficos específicos. Por ejemplo, la proyección Plate carré preserva la distancia entre el Polo Norte y el Polo Sur (Ver Fig. A.7).

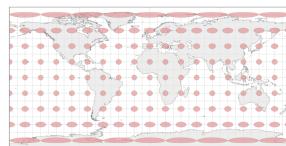


Figura A.7: Ejemplo de proyección equidistante: Platé carre

- **Proyecciones de compromiso:** No intentan preservar ninguna métrica en concreto. En su lugar, se centran en intentar encontrar un **equilibrio** entre las diversas distorsiones que provocan para intentar dar una representación más o menos representativa de la superficie terrestre. La proyección de Winkel Tripel, usada en los mapas de National Geographic, es un ejemplo de proyección de compromiso (Ver Fig. A.8).

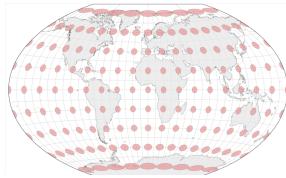


Figura A.8: Ejemplo de proyección de compromiso: Winkel Tripel

En los anteriores ejemplos se ha añadido a cada proyección la **indicatriz de Tissot**. Ésta consiste en una serie de círculos imaginarios de igual área distribuidos sobre la superficie esférica de la Tierra en determinados puntos. De este manera, al presentar la indicatriz de Tissot en una proyección específica, se puede entender de una manera intuitiva la distorsión provocada por dicha proyección, ya que los círculos se ven distorsionados o preservados según los parámetros y la naturaleza de la proyección en cuestión.

## Apéndice B

# Referencias

LO PONEMOS EN OTRO .RMD??

# Bibliografía

Robert J. Abrahart, Stan Openshaw, Robert J. Abrahart, and Linda M. See, editors. *Geocomputation*. CRC Press, 05 2000. doi: 10.4324/9780203305805. URL <http://dx.doi.org/10.4324/9780203305805>.

Adrian Baddeley and Rolf Turner. spatstat: An R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12(6):1–42, 2005. doi: 10.18637/jss.v012.i06.

Noel A. C. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, Inc., 09 1993. doi: 10.1002/9781119115151. URL <http://dx.doi.org/10.1002/9781119115151>.

A. Getis. *Spatial statistics*. Longley, P., Goodchild, M., Maguire, D. y Rhind, D. (Eds.) Geographical Information Systems, 1999. ISBN 0471-33132-5.

Diego Hernangómez. *mapSpain: Administrative Boundaries of Spain*, 2022. URL <https://ropenspain.github.io/mapSpain/>.

Robin Lovelace, Jakub Nowosad, and Jannes Muenchow. *Geocomputation with R*. R Series. CRC Press, 2019. ISBN 1-138-30451-4.

J.M. Montero, G. Fernández-Avilés, J. Mateu, and E Porcu. *Geoestadística espacial y espacio-temporal: vino nuevo, cepas viejas*. Riobóo and Riobóo (Eds.) Historia de la Probabilidad y la Estadística, 2011. ISBN 978-84-938774-9-1.

J.M. Montero, G. Fernández-Avilés, and J. Mateu. *Spatial and Spatio-Temporal Geostatistical Modeling and Kriging*. John Wiley and Sons, Inc., 2015. ISBN 978-1-118-41318-0.

Manuel Pizarro, Diego Hernangómez, and Gema Fernández-Avilés. *climaemet: Climate AEMET Tools*, 8 2021. URL <http://hdl.handle.net/10261/250390>.

I Rees, P y Turton. Guest editorial. *Environment and Planning A: Economy and Space*, 30(10):1835–1838, 10 1998. doi: 10.1068/a301835. URL <http://dx.doi.org/10.1068/a301835>.

C.A. Schabenberger, O. y Gotway. *Statistical Methods for Spatial Data Analysis*. Chapman and Hall, CRC Press., Chichester ; New York, 2005. ISBN 978-1584883227.

Kyle Walker. *crsuggest: Obtain Suggested Coordinate Reference System Information for Spatial Data*, 2021. R package version 0.3.1.