

Estadística Espacial con R

Rubén Fernández Casal (MODES, CITIC, UDC; ruben.fcasal@udc.es)
Tomás Cotos Yáñez (SIDOR, UVIGO; cotos@uvigo.es)

2021-11-24

Índice general

Prólogo	5
1 Introducción: Procesos espaciales y Geoestadística	7
1.1 Procesos espaciales	7
1.2 Geoestadística	10
1.3 Procesos espaciales estacionarios	12
1.4 Objetivos y pasos	17
2 Datos espaciales	19
2.1 Tipos de datos espaciales	19
2.2 Introducción al paquete sf	20
2.3 Representación de datos espaciales	28
2.4 Operaciones con datos espaciales	30
2.5 Análisis exploratorio de datos espaciales	34
2.6 El paquete gstat	37
3 Modelado de procesos geoestadísticos	39
3.1 Estimadores muestrales del semivariograma	40
3.2 Modelos de semivariogramas	44
3.3 Ajuste de un modelo válido	51
3.4 Modelado conjunto de la tendencia y del variograma	51
4 Predicción Kriging	53
4.1 Introducción	53
4.2 Kriging con media conocida: kriging simple	54
4.3 Kriging con media desconocida: kriging universal y kriging residual	54
4.4 Consideraciones acerca de los métodos kriging	54
5 Procesos espaciales multivariantes	55
6 Procesos espaciales espacio-temporales	57
A Introducción al paquete sp	59
A.1 Tipos de objetos	59
A.2 Métodos y procedimientos clases sp	73
A.3 Representaciones gráficas	74
B Introducción al paquete geoR	79
B.1 Inicio de una sesión y de carga de datos	79
B.2 Análisis descriptivo de datos geoestadísticos	80
B.3 Modelado de la dependencia	84
B.4 Predicción espacial (kriging)	97
Referencias	105
Bibliografía completa	105

Prólogo

La versión actual del libro *se está desarrollando* principalmente como apoyo a la docencia de la última parte de la asignatura de Análisis estadístico de datos con dependencia del Grado en Ciencia e Ingeniería de Datos de la UDC. El objetivo es que (futuras versiones del libro con contenidos adicionales) también resulte de utilidad para la docencia de la asignatura de Estadística Espacial del Máster interuniversitario en Técnicas Estadísticas).

La teoría en este libro está basada en gran parte en la tesis doctoral:

Fernández Casal, R. (2003). *Geoestadística espacio-temporal: modelos flexibles de variogramas anisotrópicos no separables*. Tesis doctoral, Universidad de Santiago de Compostela.

donde se puede encontrar información adicional.

Este libro ha sido escrito en R-Markdown empleando el paquete `bookdown` y está disponible en el repositorio Github: [rubenfcasal/estadistica_espacial](https://rubenfcasal.github.io/estadistica_espacial). Se puede acceder a la versión en línea a través del siguiente enlace:

https://rubenfcasal.github.io/estadistica_espacial (también https://bit.ly/estadistica_espacial).

donde puede descargarse en formato pdf.

Para ejecutar los ejemplos mostrados en el libro sería necesario tener instalados los siguientes paquetes: `sf`, `sp`, `starts`, `gstat`, `geoR`, `spacetime`, `sm`, `fields`, `rgdal`, `rgeos`, `maps`, `maptools`, `ggplot2`, `plot3D`, `lattice`, `classInt`, `viridis`, `dplyr`, `mapSpain`, `tmap`, `mapview`, `osmdata`, `rnaturalearth`, `ncdf`, `quadprog`, `spam`, `DEoptim`. Por ejemplo mediante los siguientes comandos:

```
pkgs <- c("sf", "sp", "starts", "gstat", "geoR", "spacetime", "sm", "fields",
         "rgdal", "rgeos", "maps", "maptools", "ggplot2", "plot3D", "lattice",
         "classInt", "viridis", "dplyr", "mapSpain", "tmap", "mapview",
         "osmdata", "rnaturalearth", "ncdf", "quadprog", "spam", "DEoptim" )

install.packages(setdiff(pkgs, installed.packages() [,"Package"])), dependencies = TRUE)
```

Si aparecen errores (normalmente debidos a incompatibilidades con versiones ya instaladas), probar a ejecutar en lugar de lo anterior:

```
install.packages(pkgs, dependencies=TRUE) # Instala todos...
```

Además, para geoestadística no paramétrica se empleará el paquete `npsp` *no disponible actualmente en CRAN* (aunque esperamos que vuelva a estarlo pronto... incluyendo soporte para el paquete `sf`). Se puede instalar la versión de desarrollo en GitHub, siguiendo las instrucciones de la web:

```
# install.packages("devtools")
devtools::install_github("rubenfcasal/npsp")
```

Aunque al necesitar compilación los usuarios de Windows deben tener instalado previamente la versión adecuada de Rtools, y Xcode los usuarios de OS X (para lo que se pueden seguir los pasos descritos aquí). Alternativamente, los usuarios de Windows (con una versión 4.X.X de R) pueden instalar este paquete ya compilado con el siguiente código:

```
install.packages('https://github.com/rubenfcasal/npsp/releases/download/v0.7-8/npsp_0.7-8.zip',
                 repos = NULL)
```

Para generar el libro (compilar) serán necesarios paquetes adicionales, para lo que se recomendaría consultar el libro de “Escritura de libros con bookdown” en castellano.



Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional (esperamos poder liberarlo bajo una licencia menos restrictiva más adelante...).

Capítulo 1

Introducción: Procesos espaciales y Geoestadística

Es bien sabido que al utilizar en la práctica métodos estadísticos no siempre es adecuado suponer que las observaciones del fenómeno de interés han sido tomadas bajo condiciones idénticas e independientes unas de otras (i.e. que los datos son independientes e idénticamente distribuidos). Esta falta de homogeneidad en los datos suele ser modelada a través de la suposición de media no constante (por ejemplo suponiendo que ésta es una combinación lineal de ciertas variables explicativas) pero con la consideración de que los errores son independientes e idénticamente distribuidos. Sin embargo, esta suposición puede influir crucialmente en la inferencia; pudiendo ser en ocasiones preferible la suposición más realista de errores correlados.

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 independientes, siendo más conveniente emplear modelos que exploten adecuadamente dicha componente espacial o espacio-temporal. De forma natural surge la hipótesis de que los datos cercanos en el espacio o en el tiempo están correlados y que la correlación disminuye al aumentar la separación entre ellos, por lo que se puede pensar en la presencia de una dependencia espacial o espacio-temporal. Esto da lugar al concepto de *proceso espacial o espacio-temporal* (Sección 1.1). La *geoestadística* (Sección 1.2) es una de las ramas de la estadística que se centra en el estudio de procesos de este tipo.

La metodología espacial y espacio-temporal ha sido utilizada de forma creciente (especialmente durante los últimos 50 años) para resolver problemas en muchos campos. En muchos casos interesa analizar datos que tienen asociada una componente espacial o espacio-temporal de forma natural, por ejemplo, en campos relacionados con la geología, hidrología, ecología, ciencias medioambientales, meteorología, epidemiología, recursos mineros, geografía, astronomía, proceso de imágenes, experimentos agrícolas, etc. En estas disciplinas la metodología espacial puede ser de ayuda en alguna o en muchas etapas del estudio, desde el diseño inicial del muestreo hasta la representación final de los resultados obtenidos (p.e. para la generación de mapas o animaciones).

1.1 Procesos espaciales

Supongamos que $Z(\mathbf{s})$ es un valor aleatorio en la posición espacial $\mathbf{s} \in \mathbb{R}^d$. Entonces, si \mathbf{s} varía dentro del conjunto índice $D \subset \mathbb{R}^d$ se obtiene el proceso espacial:

$$\{Z(\mathbf{s}) : \mathbf{s} \in D \subset \mathbb{R}^d\}$$

(también se suele denominar función aleatoria, campo espacial aleatorio o variable regionalizada). Una realización del proceso espacial se denotará por $\{z(\mathbf{s}) : \mathbf{s} \in D\}$, pero normalmente solo se observará $\{z(\mathbf{s}_1), z(\mathbf{s}_2), \dots, z(\mathbf{s}_n)\}$ (una realización parcial) en n posiciones espaciales.

Se suele distinguir entre distintos tipos de procesos espaciales dependiendo de las suposiciones acerca del dominio D :

- **Procesos geoestadísticos** (índice espacial continuo): D es un subconjunto fijo que contiene un rectángulo d -dimensional de volumen positivo. El proceso puede ser observado de forma continua dentro del dominio. Un ejemplo claro sería la temperatura, aunque normalmente solo se dispone de datos en estaciones meteorológicas fijas, se podría observar en cualquier posición (y por tanto tiene sentido predecirla). [Figura 1.1]

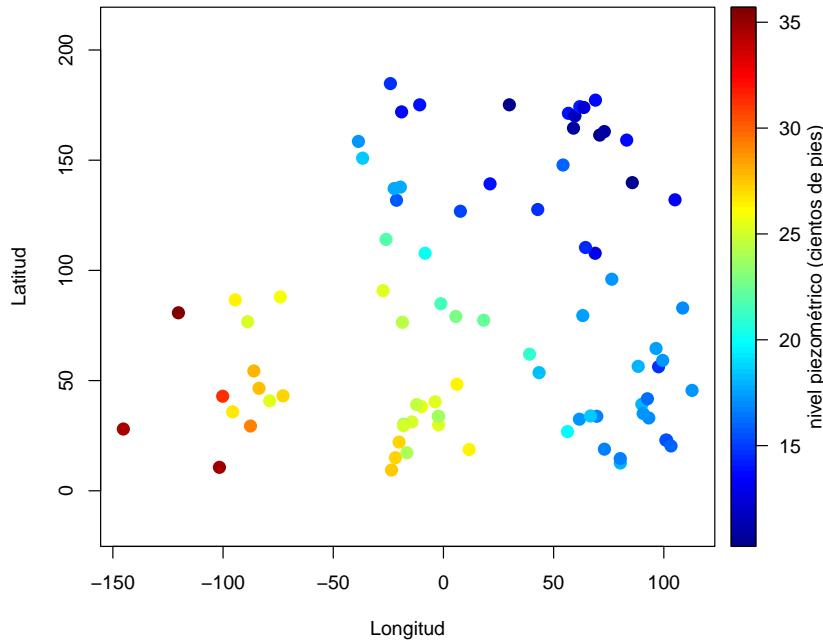


Figura 1.1: Nivel del agua subterránea en 85 localizaciones del acuífero Wolfcamp (procedentes de un estudio sobre el posible emplazamiento de un depósito de residuos nucleares).

- **Procesos reticulares/regionales** (índice espacial discreto): D es un conjunto numerable de posiciones o regiones. El proceso solo puede ser observado en determinadas posiciones. Es habitual que los datos se correspondan con agregaciones (totales o valores medios) de una determinada zona (por ejemplo, países, provincias, ayuntamientos, zonas sanitarias...). Son muy comunes en econometría o epidemiología. [Figura 1.2]
- **Procesos/patrones puntuales** (índice espacial aleatorio): D es un proceso puntual en \mathbb{R}^d . Las posiciones en las que se observa el proceso son aleatorias. En muchas casos interesa únicamente la posición donde se observa el evento de interés (por ejemplo la posición en la que creció un árbol de una determinada especie). En el caso general, además de la posición se podría observar alguna otra característica (una marca; por ejemplo la altura o el diámetro del árbol), es lo que se conoce como *proceso puntual marcado*. Este tipo de datos son habituales en biología, ecología, criminología, etc. [Figura 1.3]

Nos centraremos en el caso de procesos geoestadísticos (también denominados procesos espaciales continuos). El caso de posiciones espaciales discretas se considerará como resultado de la discretización de un proceso continuo. Esto sería válido también para el caso espacio-temporal, por ejemplo podríamos considerar posiciones de la forma $s = (s_1, \dots, s_{d-1}, t) \in \mathbb{R}^{d-1} \times \mathbb{R}^{+,0}$, donde $\mathbb{R}^{+,0} = \{t \in \mathbb{R} : t \geq 0\}$. Por tanto, las definiciones y métodos para procesos espaciales son en principio aplicables también al caso espacio-temporal. Sin embargo, la componente temporal presenta diferencias respecto a la componente espacial...

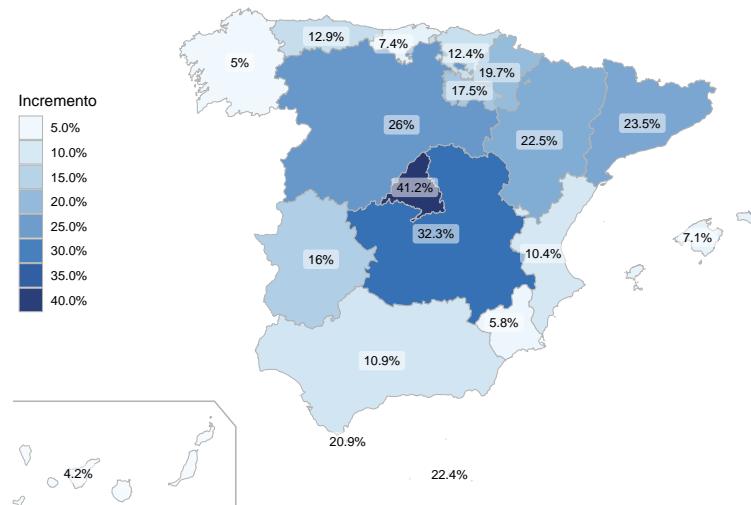


Figura 1.2: Porcentaje de incremento de las defunciones en el año 2020 respecto al 2019 por CCAA (datos provisionales [INE](<https://www.ine.es/jaxi/Tabla.htm?tpx=21856>)).



Figura 1.3: Mapa (de John Snow) del brote de cólera de 1854 en Londres.

1.1.1 Paquetes de R

En R hay disponibles una gran cantidad de paquetes para el análisis estadístico de datos espaciales (ver por ejemplo CRAN Task View: Analysis of Spatial Data). Entre ellos podríamos destacar:

- Procesos geoestadísticos: **gstat**, **geoR**, **geoRglm**, **fields**, **spBayes**, **RandomFields**, **VR:spatial**, **sgeostat**, **vardiag**, **npsp**...
- Procesos reticulares/regionales: **spdep**, **DCluster**, **spgwr**, **ade4**...

- Procesos puntuales: `spatstat`, `VR:spatial`, `splancs`...

Algunos de estos paquetes son la referencia para el análisis de este tipo de datos, aunque también están disponibles otros, como por ejemplo `maptools`, `geosphere`, `tmap`, `maps`, `leaflet`, `mapview`, `mapdeck`, `ggmap`, `rgrass7`, `RSAGA`, `RPyGeo`, `RQGIS` o `r-arcgis`, que implementan herramientas adicionales y permiten, por ejemplo, generar gráficos interactivos o interactuar con sistemas externos de información geográfica (GIS).

En todos estos paquetes se trabajan con similares tipos de datos (espaciales y espacio-temporales) por lo que se han desarrollado paquetes que facilitan su manejo. Entre ellos destacan el paquete `sp` Bivand et al. (2013) y el paquete `sf` E. Pebesma y Bivand (2021). Otros paquetes para la manipulación de datos que pueden ser de interés son: `raster`, `terra`, `stolars`, `rgdal` y `rgeos`, entre otros. En la Sección 2.1 se incluye información adicional sobre estos paquetes.

En estos apuntes emplearemos principalmente el paquete `gstat` para el análisis de datos geoestadísticos (aunque se incluye una introducción al paquete `geoR` en el Apéndice B) y el paquete `sf` para la manipulación de datos espaciales (en el Apéndice A se incluye una breve introducción a las clases `sp` para datos espaciales).

1.2 Geoestadística

La geoestadística (Matheron 1962) surgió como una mezcla de varias disciplinas: ingeniería de minas, geología, matemáticas y estadística, para dar respuesta a problemas como, por ejemplo, el de la estimación de los recursos de una explotación minera (se desarrolló principalmente a partir de los años 80). La diferencia (ventaja) respecto a otras aproximaciones es que, además de tener en cuenta la tendencia espacial (variación de gran escala), también tiene en cuenta la correlación espacial (variación de pequeña escala). Otros métodos sin embargo, sólo incluyen la variación de larga escala y suponen que los errores son independientes (Sección 1.2.1). Hoy en día podemos decir que la geoestadística es la rama de la estadística espacial que estudia los procesos con índice espacial continuo.

Uno de los problemas iniciales más importantes de la geoestadística fue la predicción de la riqueza de un bloque minero a partir de una muestra observada. A este proceso Matheron (1963) lo denominó kriging¹, y también predicción espacial lineal óptima (estos métodos de predicción se muestran en el Capítulo 4).

El modelo general habitualmente considerado en geoestadística considera que el proceso se descompone en *variabilidad de gran escala* y *variabilidad de pequeña escala*:

$$Z(\mathbf{s}) = \mu(\mathbf{s}) + \varepsilon(\mathbf{s}), \quad (1.1)$$

donde:

- $\mu(\mathbf{s}) = E(Z(\mathbf{s}))$ es la tendencia (función de regresión, determinística).
- $\varepsilon(\mathbf{s})$ es un proceso de error de media cero que incorpora la dependencia espacial.

Como en condiciones normales únicamente se dispone de una realización parcial del proceso, se suelen asumir hipótesis adicionales de estacionariedad sobre el proceso de error $\varepsilon(\mathbf{s})$ para hacer posible la inferencia. En la Sección 1.3 se definen los principales tipos de estacionariedad habitualmente considerados en geoestadística y se introducen dos funciones relacionadas con procesos estacionarios, el covariograma y el variograma. Algunas propiedades de estas funciones, que podríamos decir que son las herramientas fundamentales de la geoestadística, se muestran en la Sección 2.2.

¹ D. G. Krige fue un ingeniero de minas de Sudáfrica que desarrolló en los años 50 métodos empíricos para determinar la distribución de la riqueza de un mineral a partir de valores observados. Sin embargo la formulación de la predicción espacial lineal óptima no procede del trabajo de Krige. Al mismo tiempo que la geoestadística se desarrollaba en la ingeniería de minas por Matheron en Francia, la misma idea se desarrollaba en la meteorología por L.S. Gandin en la antigua Unión Soviética. El nombre que Gandin le dio a esta aproximación fue análisis objetivo y utilizó la terminología de interpolación óptima en lugar de kriging. Para más detalles sobre el origen del kriging ver p.e. Cressie (1990).

1.2.1 Modelos clásicos y modelos espaciales

En general, cuando se considera que la componente espacial (o espacio-temporal) puede ser importante en el modelado y el análisis de los datos es necesaria una aproximación estadística distinta a la tradicionalmente usada.

Uno de los modelos más utilizados en estadística para el caso de datos no homogéneos es el conocido modelo clásico de regresión lineal. Si $\{Z(\mathbf{s}) : \mathbf{s} \in D \subset \mathbb{R}^d\}$ es un proceso espacial, podemos suponer que:

$$Z(\mathbf{s}) = \sum_{j=0}^p x_j(\mathbf{s})\beta_j + \varepsilon(\mathbf{s}), \quad \mathbf{s} \in D,$$

(un caso particular del modelo general (1.1)), donde $\beta = (\beta_0, \dots, \beta_p)^\top \in \mathbb{R}^{p+1}$ es un vector desconocido, $\{x_j(\cdot) : j = 0, \dots, p\}$ un conjunto de variables explicativas (típicamente $x_0(\cdot) = 1$) y $\varepsilon(\cdot)$ un proceso de media cero incorrelado (i.e. $Cov(\varepsilon(\mathbf{u}), \varepsilon(\mathbf{v})) = 0$ si $\mathbf{u} \neq \mathbf{v}$) con $Var(\varepsilon(\mathbf{s})) = \sigma^2$.

Supongamos por el momento que el objetivo es la estimación eficiente de la tendencia, o lo que es lo mismo la estimación óptima de los parámetros de la *variación de gran escala* β , a partir de los datos observados en un conjunto de posiciones espaciales $\{\mathbf{s}_1, \dots, \mathbf{s}_n\}$. Bajo las hipótesis anteriores:

$$\mathbf{Z} = \mathbf{X}\beta + \varepsilon,$$

siendo $\mathbf{Z} = (Z(\mathbf{s}_1), \dots, Z(\mathbf{s}_n))^\top$, \mathbf{X} una matriz $n \times (p+1)$ con $X_{ij} = x_{j-1}(\mathbf{s}_i)$ y $\varepsilon = (\varepsilon(\mathbf{s}_1), \dots, \varepsilon(\mathbf{s}_n))^\top$; y el estimador lineal insesgado de β más eficiente resulta ser el estimador de mínimos cuadrados ordinarios (OLS, *ordinary least squares*):

$$\hat{\beta}_{ols} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Z}, \quad (1.2)$$

con

$$Var(\hat{\beta}_{ols}) = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}.$$

Sin embargo la suposición de que los errores son independientes e idénticamente distribuidos influye crucialmente en la inferencia. En el modelo anterior, en lugar de errores incorrelados, si suponemos que:

$$Var(\varepsilon) = \Sigma,$$

obtenemos el modelo lineal de regresión generalizado y en este caso el estimador lineal óptimo de β es el estimador de mínimos cuadrados generalizados (GLS, *generalized least squares*):

$$\hat{\beta}_{gls} = (\mathbf{X}^\top \Sigma^{-1} \mathbf{X})^{-1} \mathbf{X}^\top \Sigma^{-1} \mathbf{Z}. \quad (1.3)$$

Si $\Sigma = \sigma^2 \mathbf{I}_n$, siendo \mathbf{I}_n la matriz identidad $n \times n$, los estimadores (1.2) y (1.3) coinciden; pero en caso contrario las estimaciones basadas en el modelo anterior pueden llegar a ser altamente ineficientes. Puede verse fácilmente que en el caso general:

$$Var(\hat{\beta}_{gls}) = (\mathbf{X}^\top \Sigma^{-1} \mathbf{X})^{-1}, \quad Var(\hat{\beta}_{ols}) = (\mathbf{X}^\top \mathbf{X})^{-1} (\mathbf{X}^\top \Sigma \mathbf{X}) (\mathbf{X}^\top \mathbf{X})^{-1},$$

resultando además que $Var(\hat{\beta}_{ols}) - Var(\hat{\beta}_{gls})$ es una matriz semidefinida positiva (e.g. Searle, 1971, Sección 3.3).

En muchos casos el objetivo final es la predicción del proceso en una posición espacial \mathbf{s}_0 :

$$Z(\mathbf{s}_0) = \mathbf{x}^\top \beta + \varepsilon(\mathbf{s}_0),$$

donde $\mathbf{x} = (x_0(\mathbf{s}_0), \dots, x_p(\mathbf{s}_0))^\top$. Bajo las hipótesis del modelo clásico el predictor óptimo sería la estimación de la tendencia $\hat{\mu}(\mathbf{s}_0) = \mathbf{x}^\top \hat{\beta}_{ols}$ (el predictor óptimo de un error independiente sería cero). En el caso general, siguiendo esta aproximación, podríamos pensar en utilizar como predictor el estimador más eficiente de la tendencia:

$$\hat{Z}(\mathbf{s}_0) = \mathbf{x}^\top \hat{\beta}_{gls},$$

sin embargo no es el predictor lineal óptimo. Puede verse (e.g. Goldberger, 1962; Sección 4.X) que en este caso el mejor predictor lineal insesgado es:

$$\tilde{Z}(\mathbf{s}_0) = \mathbf{x}^\top \hat{\beta}_{gls} + \mathbf{c}^\top \Sigma^{-1} (\mathbf{Z} - \mathbf{X}\hat{\beta}_{gls}), \quad (1.4)$$

(el denominado predictor del *kriging universal*, Sección 4.X), siendo

$$\mathbf{c} = (Cov(\varepsilon(\mathbf{s}_1), \varepsilon(\mathbf{s}_0)), \dots, Cov(\varepsilon(\mathbf{s}_n), \varepsilon(\mathbf{s}_0)))^\top,$$

y la diferencia $Var(\tilde{Z}(\mathbf{s}_0)) - Var(Z(\mathbf{s}_0)) \geq 0$ puede ser significativamente mayor que cero² (si la dependencia espacial no es muy débil). Naturalmente, si se ignora por completo la dependencia y se emplea únicamente el estimador $\hat{\beta}_{ols}$ disminuye aún más la eficiencia de las predicciones.

Teniendo en cuenta los resultados anteriores podemos afirmar que al explotar la dependencia presente en los datos el incremento en eficiencia puede ser importante. Sin embargo el principal inconveniente es que en la práctica normalmente la matriz Σ y el vector \mathbf{c} son desconocidos. El procedimiento habitual, para evitar la estimación de $n + n(n+1)/2$ parámetros adicionales a partir del conjunto de n observaciones, suele ser la elección de un modelo paramétrico adecuado:

$$C(\mathbf{u}, \mathbf{v} | \theta) \equiv Cov(\varepsilon(\mathbf{u}), \varepsilon(\mathbf{v})),$$

i.e. suponer que $\Sigma \equiv \Sigma(\theta)$ y $\mathbf{c} \equiv \mathbf{c}(\theta)$. Una hipótesis natural es suponer que los datos cercanos en el espacio o en el tiempo están correlados y que la correlación disminuye al aumentar la separación entre ellos; por tanto es normal pensar en errores espacialmente correlados. Por ejemplo, podemos considerar:

$$C(\mathbf{u}, \mathbf{v} | \theta) = \sigma^2 \rho^{\|\mathbf{u}-\mathbf{v}\|},$$

con $\sigma^2 \geq 0$ y $0 < \rho < 1$. De esta forma, si $\hat{\theta}$ es un estimador de θ , podemos obtener por ejemplo una aproximación del predictor óptimo de $Z(\mathbf{s}_0)$ sustituyendo en (1.4) $\Sigma(\theta)$ por $\Sigma(\hat{\theta})$ y $\mathbf{c}(\theta)$ por $\mathbf{c}(\hat{\theta})$.

1.2.2 Ventajas de la aproximación espacial (y espacio-temporal)

Algunos de los beneficios de utilizar modelos espaciales para caracterizar y explotar la dependencia espacial de un conjunto de datos son los siguientes:

- Modelos más generales: en la mayoría de los casos, los modelos clásicos no espaciales son un caso particular de un modelo espacial.
- Estimaciones más eficientes: de la tendencia, de los efectos de variables explicativas, de promedios regionales...
- Mejora de las predicciones: más eficientes, con propiedades de extrapolación más estables...
- 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.

1.3 Procesos espaciales estacionarios

Supongamos que $\{Z(\mathbf{s}) : \mathbf{s} \in D \subset \mathbb{R}^d\}$ es un proceso geoestadístico. Este proceso aleatorio se puede caracterizar a través de las funciones de distribución finito-dimensionales:

$$F_{\mathbf{s}_1, \dots, \mathbf{s}_m}(z_1, \dots, z_m) = P(Z(\mathbf{s}_1) \leq z_1, \dots, Z(\mathbf{s}_m) \leq z_m)$$

²Por ejemplo, para un caso particular, Goldberger (1962, pp.374-375) observó que la mejora en la predicción puede llegar a ser del 50%. En Cressie (1993, Sección 1.3) se muestran también otros ejemplos del efecto de la presencia de correlación en la estimación.

(o de las funciones de densidad correspondientes $f_{\mathbf{s}_1, \dots, \mathbf{s}_m}(z_1, \dots, z_m)$). Por ejemplo, el proceso se dice normal (o gaussiano) si para cada posible conjunto de $m \in \mathbb{N}$ posiciones espaciales, $\{\mathbf{s}_1, \dots, \mathbf{s}_m\}$, su función de distribución $F_{\mathbf{s}_1, \dots, \mathbf{s}_m}$ es normal (gaussiana).

Como ya se comentó en la Sección 1.1, en general no se puede disponer de una realización completa del proceso $Z(\cdot)$ y solamente se observan valores en unas posiciones espaciales conocidas $\{\mathbf{s}_1, \dots, \mathbf{s}_n\}$ (que por lo general van a ser irregulares). Por tanto es necesario hacer algunas suposiciones acerca del proceso de forma que sea posible la inferencia sobre el mismo. Lo habitual es asumir algún tipo de estacionariedad del proceso (o del proceso de error, suponiendo que el proceso no tiene media constante y sigue el modelo general (1.1)).

El proceso $Z(\cdot)$ se dice *estrictamente estacionario* si al trasladar (en cualquier dirección) una configuración cualquiera de posiciones espaciales la distribución conjunta no varia:

$$F_{\mathbf{s}_1 + \mathbf{h}, \dots, \mathbf{s}_m + \mathbf{h}}(z_1, \dots, z_m) = F_{\mathbf{s}_1, \dots, \mathbf{s}_m}(z_1, \dots, z_m), \quad \forall \mathbf{h} \in D, \quad \forall m \geq 1.$$

El proceso $Z(\cdot)$ se dice *estacionario de segundo orden* (también proceso estacionario homogéneo o débilmente estacionario) si tiene media constante y la covarianza entre dos posiciones depende únicamente del salto entre ellas:

- $E(Z(\mathbf{s})) = \mu, \quad \forall \mathbf{s} \in D.$
- $Cov(Z(\mathbf{s}_1), Z(\mathbf{s}_2)) = C(\mathbf{s}_1 - \mathbf{s}_2), \quad \forall \mathbf{s}_1, \mathbf{s}_2 \in D.$

La función $C(\cdot)$ se denomina *covariograma* (también autocovariograma o función de covarianzas). Si además $C(\mathbf{h}) \equiv C(\|\mathbf{h}\|)$ (sólo depende de la magnitud y no de la dirección del salto) se dice que el covariograma es *isotrópico* (en caso contrario se dice que es *anisotrópico*; 3.2.2).

Si un proceso es estrictamente estacionario y $Var(Z(\mathbf{s}))$ es finita, entonces es estacionario de segundo orden. Además, como es bien conocido, en el caso de procesos normales ambas propiedades son equivalentes (ya que están caracterizados por su media y covarianza).

En algunos casos en lugar del covariograma se utiliza el correlograma:

$$\rho(\mathbf{h}) = \frac{C(\mathbf{h})}{C(\mathbf{0})} \in [-1, +1],$$

suponiendo que $C(\mathbf{0}) = Var(Z(\mathbf{s})) > 0$. Sin embargo lo habitual es modelar la dependencia espacial a través del variograma (principalmente por sus ventajas en la estimación; Sección 3.1), definido a continuación.

Se dice que el proceso es *intrínsecamente estacionario* (también proceso espacial de incrementos estacionarios u homogéneos) si:

- $E(Z(\mathbf{s})) = \mu, \quad \forall \mathbf{s} \in D.$
- $Var(Z(\mathbf{s}_1) - Z(\mathbf{s}_2)) = 2\gamma(\mathbf{s}_1 - \mathbf{s}_2), \quad \forall \mathbf{s}_1, \mathbf{s}_2 \in D.$

La función $2\gamma(\cdot)$ se denomina *variograma* y $\gamma(\cdot)$ *semivariograma*. Al igual que en el caso anterior, si además $\gamma(\mathbf{h}) \equiv \gamma(\|\mathbf{h}\|)$ (sólo depende de la distancia) se dice que el variograma es isotrópico.

La clase de procesos intrínsecamente estacionarios es más general que la clase de procesos estacionarios de segundo orden. Si un proceso estacionario de segundo orden tiene covariograma $C(\cdot)$, como:

$$\begin{aligned} Var(Z(\mathbf{s}_1) - Z(\mathbf{s}_2)) &= Var(Z(\mathbf{s}_1)) + Var(Z(\mathbf{s}_2)) - 2Cov(Z(\mathbf{s}_1), Z(\mathbf{s}_2)) \\ &= 2(C(\mathbf{0}) - C(\mathbf{s}_1 - \mathbf{s}_2)), \end{aligned}$$

entonces su semivariograma viene dado por:

$$\gamma(\mathbf{h}) = C(\mathbf{0}) - C(\mathbf{h}),$$

y por tanto es un proceso intrínsecamente estacionario. El reciproco en general no es cierto (por ejemplo el caso de un movimiento browniano), aunque sí se verifica en muchos casos. Normalmente cuando no se verifica es debido a que el proceso no tiene media constante y puede ser modelado como

una función de tendencia más un error estacionario de segundo orden (o cuando se consideran los errores del modelo general, la tendencia no está especificada correctamente).

Si el variograma está acotado y:

$$\lim_{\|\mathbf{h}\| \rightarrow \infty} \gamma(\mathbf{h}) = \sigma^2,$$

entonces³ podemos obtener el covariograma correspondiente como:

$$C(\mathbf{h}) = \sigma^2 - \gamma(\mathbf{h}).$$

A $\sigma^2 = C(\mathbf{0})$ se le denomina *umbral* (o *meseta*) del semivariograma. La relación entre el semivariograma y el covariograma se ilustra en la Figura 1.4.

Además del umbral, hay otras características geométricas del variograma (o del covariograma) de especial importancia⁴, entre ellas destacarían el *efecto pepita* (o *nugget*) y el *rango* (o *alcance*). La Figura 1.4 ilustra las distintas características del semivariograma.

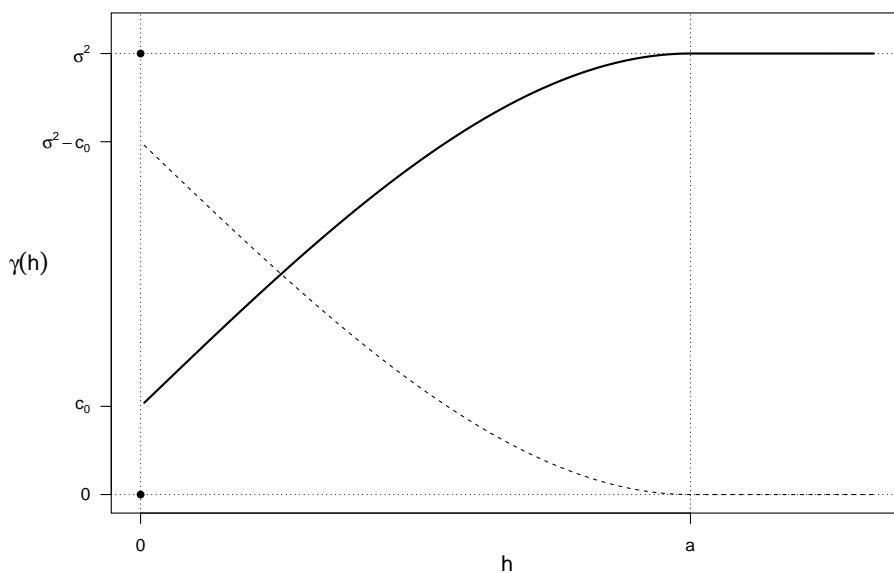


Figura 1.4: Relación entre el covariograma (línea discontinua) y el variograma (línea continua) en el caso unidimensional (o isotrópico), y principales características.

Siempre se verifica que $\gamma(\mathbf{0}) = 0$, sin embargo puede ser que:

$$\lim_{\mathbf{h} \rightarrow \mathbf{0}} \gamma(\mathbf{h}) = c_0 > 0.$$

entonces c_0 se denomina *efecto pepita* (o *nugget*)⁵. Además, si σ^2 es el umbral del semivariograma (suponiendo que existe), a $\sigma^2 - c_0$ se le denomina *umbral parcial*.

Las propiedades de continuidad (y derivabilidad) del variograma (o el covariograma) en el origen están relacionadas con las propiedades de continuidad (y diferenciabilidad) en media cuadrática del proceso $Z(\cdot)$ (ver e.g. Chilès y Delfiner, 1999, Sección 2.3.1). Por ejemplo, el proceso es continuo en media cuadrática si y sólo si su variograma (covariograma) es continuo en el origen. Entonces la presencia de efecto nugget indica que (en teoría) el proceso no es continuo y por tanto altamente irregular.

³Suponiendo que $\lim_{\|\mathbf{h}\| \rightarrow \infty} C(\mathbf{h}) = 0$.

⁴Además de poder interpretar su influencia en la predicción espacial (Sección 4.X), son utilizadas en la parametrización de la mayoría de los modelos de variogramas o covariogramas (Sección 3.2.1).

⁵El origen de esta denominación está relacionado con la terminología minera. En algunos yacimientos de metal, como por ejemplo en el caso del oro, el mineral suele obtenerse como pepitas de material puro y estas pepitas normalmente son más pequeñas que el tamaño de la unidad de muestreo (lo que produce una variabilidad adicional en la muestra).

La proporción del efecto nugget en el umbral total c_0/σ^2 proporciona mucha información acerca del grado de dependencia espacial presente en los datos. Por ejemplo, en el caso en que toda la variabilidad es efecto nugget (i.e. $\gamma(\mathbf{h}) = c_0, \forall \mathbf{h} \neq \mathbf{0}$) entonces $Z(\mathbf{s}_1)$ y $Z(\mathbf{s}_2)$ son incorrelados $\forall \mathbf{s}_1, \mathbf{s}_2 \in D$ independientemente de lo cerca que estén (el proceso $Z(\cdot)$ es ruido blanco). Por tanto podemos pensar en c_0/σ^2 como la proporción de “variabilidad independiente”, aunque en la práctica típicamente no se dispone de información sobre el variograma a distancias menores de $\min\{\|\mathbf{s}_i - \mathbf{s}_j\| : 1 \leq i < j \leq n\}$ (la estimación de c_0 se obtiene normalmente extrapolando un variograma experimental cerca del origen).

Si σ^2 es el umbral del semivariograma (suponiendo que existe), se define el *rango* (o alcance) del semivariograma en la dirección $\mathbf{e}_0 \in \mathbb{R}^d$ con $\|\mathbf{e}_0\| = 1$, como el mínimo salto en esa dirección en el que se alcanza el umbral:

$$a_0 = \min \{a : \gamma(a(1 + \varepsilon)\mathbf{e}_0) = \sigma^2, \forall \varepsilon > 0\}.$$

El rango en la dirección \mathbf{e}_0 puede interpretarse como el salto h a partir del cual no hay correlación entre $Z(\mathbf{s})$ y $Z(\mathbf{s} \pm h\mathbf{e}_0)$, por tanto está íntimamente ligado a la noción de “zona de influencia” (y tiene un papel importante en la determinación de criterios de vecindad). En los casos en los que el semivariograma alcanza el umbral asintóticamente (rango infinito), se suele considerar el *rango práctico*, definido como el mínimo salto en el que se alcanza el 95% del umbral.

El variograma y el covariograma son las funciones habitualmente consideradas en geoestadística para el modelado de la dependencia espacial (o espacio-temporal), y son consideradas como un parámetro (de especial interés) del proceso. En la práctica normalmente se suele utilizar el variograma, no sólo porque es más general (puede existir en casos en que el covariograma no), sino por las ventajas en su estimación (Sección 3.1; Cressie, 1993, Sección 2.4.1). No obstante, en muchos casos los modelos de variograma se obtienen a partir de modelos de covariograma.

1.3.1 Propiedades elementales del covariograma y del variograma

El variograma y el covariograma deben verificar ciertas propiedades que sus estimadores no siempre verifican, a continuación se detallan algunas de ellas.

Si $Z(\cdot)$ es un proceso estacionario de segundo orden con covariograma $C(\cdot)$, entonces se verifica que $C(\mathbf{0}) = \text{Var}(Z(\mathbf{s})) \geq 0$, es una función par $C(\mathbf{h}) = C(-\mathbf{h})$, y por la desigualdad de Cauchy-Schwarz $|C(\mathbf{h})| \leq C(\mathbf{0})$. Además, el covariograma debe ser semidefinido positivo, es decir:

$$\sum_{i=1}^m \sum_{j=1}^m a_i a_j C(\mathbf{s}_i - \mathbf{s}_j) \geq 0 \quad \forall m \geq 1, \quad \forall \mathbf{s}_i \in D, \quad \forall a_i \in \mathbb{R}; \quad i = 1, \dots, m,$$

ya que:

$$\sum_{i=1}^m \sum_{j=1}^m a_i a_j C(\mathbf{s}_i - \mathbf{s}_j) = \text{Var} \left\{ \sum_{i=1}^m a_i Z(\mathbf{s}_i) \right\}$$

La condición es necesaria y suficiente para que exista un proceso estacionario de segundo orden con covariograma $C(\cdot)$ (se puede construir un proceso normal multivariante con covarianzas definidas por $C(\cdot)$). Por tanto la clase de covariogramas válidos en \mathbb{R}^d es equivalente a la clase de funciones semidefinidas positivas en \mathbb{R}^d .

Algunas propiedades adicionales que verifican los covariogramas son las siguientes:

1. Si $C(\cdot)$ es un covariograma válido en \mathbb{R}^d , entonces $aC(\cdot)$, $\forall a \geq 0$, es también un covariograma válido en \mathbb{R}^d .
2. Si $C_1(\cdot)$ y $C_2(\cdot)$ son covariogramas válidos en \mathbb{R}^d , entonces $C_1(\cdot) + C_2(\cdot)$ es un covariograma válido en \mathbb{R}^d . Lo que equivale a suponer que el proceso $Z(\cdot)$ se obtiene como suma de dos procesos estacionarios de segundo orden independientes: $Z(\mathbf{s}) = Z_1(\mathbf{s})Z_2(\mathbf{s})$, con covariogramas $C_1(\cdot)$ y $C_2(\cdot)$ respectivamente.
3. Si $C_1(\cdot)$ y $C_2(\cdot)$ son covariogramas válidos en \mathbb{R}^d , entonces $C(\cdot) = C_1(\cdot)C_2(\cdot)$ es un covariograma válido en \mathbb{R}^d . Lo que equivale a suponer que el proceso se obtiene como producto de dos procesos estacionarios de segundo orden independientes.

4. Un covariograma isotrópico válido en \mathbb{R}^d es también un covariograma isotrópico válido en \mathbb{R}^m , $\forall m \leq d$ (el recíproco no es en general cierto, ver p.e. Cressie, 1993, p. 84).

Si $\gamma(\cdot)$ es el semivariograma de un proceso intrínsecamente estacionario $Z(\cdot)$, entonces se verifica que $\gamma(\mathbf{0}) = 0$, $\gamma(\mathbf{h}) \geq 0$ y $\gamma(\mathbf{h}) = \gamma(-\mathbf{h})$. El semivariograma debe ser además condicionalmente semidefinito negativo, es decir:

$$\sum_{i=1}^m \sum_{j=1}^m a_i a_j \gamma(\mathbf{s}_i - \mathbf{s}_j) \leq 0 \quad \forall m \geq 1, \forall \mathbf{s}_i \in D, \forall a_i \in \mathbb{R}; i = 1, \dots, m, \text{ tales que } \sum_{i=1}^m a_i = 0.$$

Esta condición es necesaria pero no suficiente (aunque pocas condiciones adicionales son necesarias para que el recíproco sea cierto; ver Cressie, 1993, Sección 3.5.2).

Algunas propiedades adicionales que verifica un variograma son las siguientes:

1. Si $\gamma(\cdot)$ es un semivariograma válido en \mathbb{R}^d , entonces $a\gamma(\cdot)$, $\forall a \geq 0$, es también un semivariograma válido en \mathbb{R}^d .
2. Si $\gamma_1(\cdot)$ y $\gamma_2(\cdot)$ son semivariogramas válidos en \mathbb{R}^d , entonces $\gamma_1(\cdot) + \gamma_2(\cdot)$, es también un semivariograma válido en \mathbb{R}^d . Lo que equivale a suponer que el proceso $Z(\cdot)$ se obtiene como suma de dos procesos intrínsecamente estacionarios independientes: $Z(\mathbf{s}) = Z_1(\mathbf{s}) + Z_2(\mathbf{s})$, con semivariogramas $\gamma_1(\cdot)$ y $\gamma_2(\cdot)$ respectivamente.
3. Un variograma isotrópico válido en \mathbb{R}^d es también un variograma isotrópico válido en \mathbb{R}^m , $\forall m \leq d$.

Se suelen emplear estas propiedades para la obtención de modelos de variograma válidos, como por ejemplo en el caso de la anisotropía zonal (Sección 3.2.2) o del modelo lineal de (co)regionalización (secciones 3.2.3 y 5.X).

1.3.2 Procesos agregados

En algunos casos los datos pueden ser agregaciones espaciales en lugar de observaciones puntuales (e incluso observaciones sobre distintos soportes) o, por ejemplo, puede ser de interés la estimación de medias espaciales a partir de datos puntuales. Estas agregaciones pueden ser modeladas como el promedio de un proceso puntual, lo que permite deducir fácilmente las relaciones entre covariogramas y variogramas vinculados a diferentes soportes.

Supongamos que el proceso espacial $Z(\cdot)$ definido sobre $D \subset \mathbb{R}^d$ es integrable en media cuadrática. Entonces, si $B \subset D$ es un subconjunto acotado e integrable con $|B| = \int_B d\mathbf{s} > 0$, se puede definir el proceso espacial agregado (también se denomina regularizado) como:

$$Z(B) \equiv \frac{1}{|B|} \int_B Z(\mathbf{s}) d\mathbf{s}.$$

Si por ejemplo el proceso puntual es intrínsecamente estacionario con semivariograma $\gamma(\cdot)$, entonces a partir del variograma puntual podemos obtener el variograma del proceso agregado:

$$\begin{aligned} Var(Z(B_1) - Z(B_2)) &= -\frac{1}{|B_1|^2} \int_{B_1} \int_{B_1} \gamma(\mathbf{s} - \mathbf{u}) d\mathbf{s} d\mathbf{u} \\ &\quad - \frac{1}{|B_2|^2} \int_{B_2} \int_{B_2} \gamma(\mathbf{s} - \mathbf{u}) d\mathbf{s} d\mathbf{u} \\ &\quad + \frac{1}{|B_1||B_2|} \int_{B_1} \int_{B_2} 2\gamma(\mathbf{s} - \mathbf{u}) d\mathbf{s} d\mathbf{u}. \end{aligned}$$

Aunque nos centraremos principalmente en el caso de soporte puntual, los métodos descritos en estos apuntes pueden ser extendidos para el caso de distintos soportes (por ejemplo el *block kriging* descrito en la Sección 4.X). Sin embargo, en la práctica pueden aparecer dificultades, especialmente al combinar observaciones en distintos soportes (esto es lo que se conoce como el problema de cambio de soporte, o el *modifiable areal unit problem*, MAUP). Para más detalles ver por ejemplo Cressie (1993, Sección 5.2) ó Chilès y Delfiner (1999, Sección 2.4).

1.4 Objetivos y pasos

A partir de los valores observados $\{Z(\mathbf{s}_1), \dots, Z(\mathbf{s}_n)\}$ (o $\{Z(B_1), \dots, Z(B_n)\}$), los objetivos suelen ser:

- Obtener predicciones (kriging) $\hat{Z}(\mathbf{s}_0)$ (o $\hat{Z}(B_0)$).
- Realizar inferencias (estimación, contrastes) sobre las componentes del modelo $\hat{\mu}(\cdot)$, $\hat{\gamma}(\cdot)$.
- Obtención de mapas de riesgo $P(Z(\mathbf{s}_0) \geq c)$.
- Realizar inferencias sobre la distribución (condicional) de la respuesta en nuevas localizaciones...

En cualquier caso en primer lugar habría que estimar las componentes del modelo: la tendencia $\mu(\mathbf{s})$ y el semivariograma $\gamma(\mathbf{h})$. La aproximación tradicional (paramétrica) para el modelado de un proceso geoestadístico consiste en los siguientes pasos:

1. Análisis exploratorio y formulación de un modelo paramétrico inicial (Capítulo 2).
2. Estimación de los parámetros del modelo (proceso iterativo; Capítulo 3):
 1. Estimar y eliminar la tendencia.
 2. Modelar la dependencia (ajustar un modelo de variograma) a partir de los residuos.
3. Validación del modelo (Sección 4.X) o reformulación del mismo.
4. Empleo del modelo aceptado (Capítulo 4).

Capítulo 2

Datos espaciales

En este capítulo se incluye una breve introducción a los tipos de datos espaciales (Sección 2.1) y a su manipulación en R con el paquete `sf` (secciones 2.2 y 2.4). La parte final se centra en el análisis exploratorio de datos espaciales (Sección 2.5).

2.1 Tipos de datos espaciales

En el campo de los datos espaciales se suele distinguir entre dos tipos de datos:

- *Datos vectoriales*: en los que se emplean coordenadas para definir las posiciones espaciales “exactas” de los datos. Entre ellos estarían los asociados a las geometrías habituales: puntos, líneas, polígonos y rejillas.
- *Datos ráster*: se utilizan habitualmente para representar una superficie continua. Un ráster no es más que una rejilla regular que determina un conjunto de rectángulos denominados celdas (o píxeles en el análisis de imágenes de satélite y teledetección) que tienen asociados uno o más valores. Este tipo de datos también se denominan *arrays* o *data cubes* espaciales (o espacio-temporales). El valor de una celda ráster suele ser el valor medio (o el total) de una variable en el área que representa (se trataría de observaciones de un proceso agregado, descritos en la Sección 1.3.2), aunque en algunos casos es el valor puntual correspondiente al centro de la celda (nodo de una rejilla vectorial).

En estos apuntes entenderemos que *ráster* hace referencia a agregaciones espaciales y nos centraremos principalmente en datos vectoriales (incluyendo rejillas de datos), aunque hoy en día cada vez es más habitual disponer de datos ráster gracias a la fotografía aérea y a la teledetección por satélite. Como se comentó en la Sección 1.3.2, muchos métodos geoestadísticos admiten datos en distintos soportes (por ejemplo el *block kriging* descrito en la Sección 4.X), aunque combinar datos en diferentes soportes puede presentar en la práctica serias dificultades (para más detalles ver referencias al final de la Sección 1.3.2).

Como ya se comentó en la Sección 1.1, dependiendo de las suposiciones sobre el soporte del proceso (índice espacial) se distingue entre distintos tipos de procesos espaciales. Sin embargo, aunque en principio los objetivos pueden ser muy distintos, en todos estos casos se trabaja con datos similares (espaciales y espacio-temporales):

- **Procesos geoestadísticos** (índice espacial continuo):
 - *Datos*: coordenadas y valores observados (puntos y datos), opcionalmente se pueden considerar los límites de una región de observación o de múltiples regiones (polígonos).
 - *Resultados*: superficie de predicción (rejilla), opcionalmente predicciones por área (polígonos y datos, o raster).
- **Procesos reticulares/regionales** (índice espacial discreto):

- *Datos*: límites de regiones y valores asociados (polígonos y datos, , o raster).
- *Resultados*: estimaciones por área (polígonos y datos, o raster).

- **Procesos puntuales** (indice espacial aleatorio):

- *Datos*: coordenadas (puntos), opcionalmente con valores asociados (procesos marcados; puntos y datos), límites región de observación (polígonos).
- *Resultados*: superficie de incidencia o probabilidad (rejilla).

Este es el principal motivo de que se hayan desarrollado paquetes de R para facilitar su manipulación (y permitiendo el intercambio de datos entre herramientas). Entre ellos destacan:

- **sp** (Classes and methods for spatial data, E. J. Pebesma y Bivand, 2005): se corresponde con Bivand et al. (2013) y emplea clases S4. Se complementa con los paquetes **rgdal** (interfaz a la *geospatial data abstraction library*, para la lectura y escritura de datos espaciales) y **rgeos** (interfaz a la librería *Geometry Engine Open Source*, para operaciones geométricas).
- **sf** (Simple Features for R, E. Pebesma, 2018): alternativa en desarrollo con objetos más simples S3 (compatible con **tidyverse** y que proporciona una interfaz directa a las librerías GDAL y RGEOS) que aspira a reemplazar el paquete **sp** a corto plazo. Se corresponde con E. Pebesma y Bivand (2021) (disponible online).

El paquete **sp** tiene un soporte limitado para datos ráster, este es uno de los motivos por los que surgió el paquete **raster**, que actualmente está siendo reemplazado por el paquete **terra** (información sobre estos paquetes está disponible en el manual online). El paquete **sf** no implementa datos ráster (y tiene un soporte muy limitado para rejillas de datos), para manejar este tipo de datos se complementa con el paquete **stolars** (Spatiotemporal Arrays: Raster and Vector Datacubes). Para detalles sobre la conversión entre datos ráster y datos vectoriales ver por ejemplo las secciones 7.5 y 7.7 de E. Pebesma y Bivand (2021).

En este capítulo emplearemos el paquete **sf** para la manipulación de datos espaciales, aunque en el Apéndice A se incluye una breve introducción a las clases **sp**, ya que este tipo de objetos siguen siendo ampliamente empleados en la actualidad (y, de momento, algunas de las herramientas disponibles en R solo admiten las clases de datos definidas en este paquete).

2.2 Introducción al paquete sf

El modelo de geometrías de *características simples* (o rasgos simples) es un estándar (ISO 19125) desarrollado por el Open Geospatial Consortium (OGC) para formas geográficas vectoriales, que ha sido adoptado por gran cantidad de software geográfico (entre otros por GeoJSON, ArcGIS, QGIS, PostGIS, MySQL Spatial Extensions, Microsoft SQL Server...). Como ya se comentó, este tipo de datos espaciales está implementado en R en el paquete **sf**.

Los objetos principales, del tipo **sf**, son extensiones de **data.frame** (o **tibble**) y como mínimo contienen una columna denominada *simple feature geometry list column* que contiene la geometría de cada observación (se trata de una columna tipo **list**). Cada fila, incluyendo la geometría y otras posibles variables (denominados atributos de la geometría), se considera una característica simple (SF).

```
library(sf)

## Linking to GEOS 3.9.1, GDAL 3.2.1, PROJ 7.2.1
nc <- st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)
nc <- nc[c(5, 9:15)]

## Simple feature collection with 100 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## Geodetic CRS:  NAD27
```

```

## First 10 features:
##      NAME BIR74 SID74 NWBIR74 BIR79 SID79 NWBIR79
## 1     Ashe  1091    1    10  1364    0    19
## 2   Alleghany  487    0    10  542    3    12
## 3     Surry 3188    5    208 3616    6   260
## 4   Currituck  508    1   123  830    2   145
## 5 Northampton 1421    9   1066 1606    3  1197
## 6   Hertford 1452    7   954 1838    5  1237
## 7    Camden  286    0   115  350    2   139
## 8     Gates  420    0   254  594    2   371
## 9    Warren  968    4   748 1190    2   844
## 10   Stokes 1612    1   160 2038    5   176
##                           geometry
## 1 MULTIPOLYGON (((-81.47276 3...
## 2 MULTIPOLYGON (((-81.23989 3...
## 3 MULTIPOLYGON (((-80.45634 3...
## 4 MULTIPOLYGON (((-76.00897 3...
## 5 MULTIPOLYGON (((-77.21767 3...
## 6 MULTIPOLYGON (((-76.74506 3...
## 7 MULTIPOLYGON (((-76.00897 3...
## 8 MULTIPOLYGON (((-76.56251 3...
## 9 MULTIPOLYGON (((-78.30876 3...
## 10 MULTIPOLYGON (((-80.02567 3...

str(nc)

## Classes 'sf' and 'data.frame': 100 obs. of 8 variables:
## $ NAME : chr "Ashe" "Alleghany" "Surry" "Currituck" ...
## $ BIR74 : num 1091 487 3188 508 1421 ...
## $ SID74 : num 1 0 5 1 9 7 0 0 4 1 ...
## $ NWBIR74 : num 10 10 208 123 1066 ...
## $ BIR79 : num 1364 542 3616 830 1606 ...
## $ SID79 : num 0 3 6 2 3 5 2 2 2 5 ...
## $ NWBIR79 : num 19 12 260 145 1197 ...
## $ geometry:sfc_MULTIPOLYGON of length 100; first list element: List of 1
## ..$ :List of 1
## ...$ : num [1:27, 1:2] -81.5 -81.5 -81.6 -81.6 -81.7 ...
## ...- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
## - attr(*, "sf_column")= chr "geometry"
## - attr(*, "agr")= Factor w/ 3 levels "constant", "aggregate", ...: NA NA NA NA NA NA NA
## ...- attr(*, "names")= chr [1:7] "NAME" "BIR74" "SID74" "NWBIR74" ...

```

El nombre de la columna de geometrías está almacenado en el atributo `"sf_column"` del objeto y se puede acceder a ella mediante la función `st_geometry()` (además de poder emplear los procedimientos habituales para acceder a los componentes de un `data.frame`). Esta columna es un objeto de tipo `sfc` (*simple feature geometry list column*), descritos más adelante.

```

# geom_name <- attr(nc, "sf_column")
# nc[, geom_name]; nc[[geom_name]]
# nc$geometry
st_geometry(nc)

## Geometry set for 100 features
## Geometry type: MULTIPOLYGON
## Dimension: XY
## Bounding box: xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## Geodetic CRS: NAD27
## First 5 geometries:

```

```
## MULTIPOLYGON (((-81.47276 36.23436, -81.54084 3...
## MULTIPOLYGON (((-81.23989 36.36536, -81.24069 3...
## MULTIPOLYGON (((-80.45634 36.24256, -80.47639 3...
## MULTIPOLYGON (((-76.00897 36.3196, -76.01735 36...
## MULTIPOLYGON (((-77.21767 36.24098, -77.23461 3...
```

En este paquete, todos los métodos y funciones que operan sobre datos espaciales comienzan por `st_` (*spatial type*; siguiendo la implementación de PostGIS):

```
methods(class="sf")

## [1] $<-
## [4] aggregate
## [7] coerce
## [10] filter
## [13] merge
## [16] rbind
## [19] st_agr
## [22] st_as_s2
## [25] st_boundary
## [28] st_centroid
## [31] st_coordinates
## [34] st_crs<-
## [37] st_geometry
## [40] st_interpolate_aw
## [43] st_is
## [46] st_line_merge
## [49] st_nearest_points
## [52] st_point_on_surface
## [55] st_reverse
## [58] st_set_precision
## [61] st_snap
## [64] st_triangulate
## [67] st_wrap_dateline
## [70] st_zm
## see '?methods' for accessing help and source code
```

Los objetos geométricos básicos son del tipo `sfg` (*simple feature geometry*) que contienen la geometría de una única característica definida a partir de puntos en dos (XY), tres (XYZ, XYM) o cuatro dimensiones (XYZM). Admite los 17 tipos de geometrías simples del estándar, pero de forma completa los 7 tipos básicos:

Tipo	Description	Creación
POINT,	Punto o conjunto de puntos	<code>st_point()</code> ,
MULTIPOINT		<code>st_multipoint()</code>
LINESTRING,	Línea o conjunto de líneas	<code>st_linestring()</code> ,
MULTILINESTRING		<code>st_multilinestring()</code>
POLYGON,	Polígono ¹ o conjunto de polígonos	<code>st_polygon()</code> ,
MULTIPOLYGON		<code>st_multipolygon()</code>
GEOMETRYCOLLECTION	Conjunto de geometrías de los tipos anteriores	<code>st_geometrycollection()</code>

¹ Secuencia de puntos que forma un anillo cerrado, que no se interseca; el primero anillo definen el anillo exterior, anillos posteriores definen agujeros. Según la norma, los puntos del anillo exterior deben especificarse en sentido contrario a las agujas del reloj y los de los agujeros en sentido de las agujas del reloj.

Las geometrías se imprimen empleando la representación *well-known text* (WKT) del estándar (se exportan empleando la representación *well-known binary*, WKB).

```
nc$geometry[[1]]
```

```
## MULTIPOLYGON (((-81.47276 36.23436, -81.54084 36.27251, -81.56198 36.27359, -81.63306 36.34069,
```

Los objetos básicos `sfg` (normalmente del mismo tipo) se pueden combinar en un objeto `sfc` (*simple feature geometry list column*) mediante la función `st_sfg()`. Estos objetos pueden incorporar un sistema de referencia de coordenadas (por defecto `NA_crs_`), descritos en la Sección 2.2.1. Posteriormente se puede crear un objeto `sf` mediante la función `st_sf()`.

```
p1 <- st_point(c(-8.395835, 43.37087))
p2 <- st_point(c(-7.555851, 43.01208))
p3 <- st_point(c(-7.864641, 42.34001))
p4 <- st_point(c(-8.648053, 42.43362))
sfc <- st_sfc(list(p1, p2, p3, p4))
cprov <- st_sf(names = c('Coruña (A)', 'Lugo', 'Ourense', 'Pontevedra'),
geom = sfc)
cprov
```

```
## Simple feature collection with 4 features and 1 field
## Geometry type: POINT
## Dimension: XY
## Bounding box: xmin: -8.648053 ymin: 42.34001 xmax: -7.555851 ymax: 43.37087
## CRS: NA
##           names          geom
## 1 Coruña (A) POINT (-8.395835 43.37087)
## 2      Lugo POINT (-7.555851 43.01208)
## 3    Ourense POINT (-7.864641 42.34001)
## 4 Pontevedra POINT (-8.648053 42.43362)
```

Esta forma de proceder puede resultar de interés cuando se construyen geometrías tipo líneas o polígonos, pero en el caso de datos puntuales (las observaciones habituales en geoestadística), resulta mucho más cómodo emplear un `data.frame` que incluya las coordenadas en columnas y convertirlo a un objeto `sf` mediante la función `st_as_sf()`.

Ejercicio 2.1 (Creación de una columna de geometrías). Crear una geometría (un objeto `sfc`) formada por: dos puntos en las posiciones (1,5) y (5,5), una línea entre los puntos (1,1) y (5,1), y un polígono, con vértices {(0,0), (6,0), (6,6), (0,6), (0,0)} y con un agujero con vértices {(2,2), (2,4), (4,4), (4,2), (2,2)} (NOTA: consultar la ayuda `?st`, puede resultar cómodo emplear `matrix(..., ncol = 2, byrow = TRUE)`).

Como ejemplo consideraremos el conjunto de datos `meuse` del paquete `sp` que contiene concentraciones de metales pesados, junto con otras variables del terreno, en una zona de inundación del río Meuse (cerca de Stein, Holanda)² (ver Figura 2.1).

```
data(meuse, package="sp")
str(meuse)

## 'data.frame': 155 obs. of 14 variables:
## $ x      : num 181072 181025 181165 181298 181307 ...
## $ y      : num 333611 333558 333537 333484 333330 ...
## $ cadmium: num 11.7 8.6 6.5 2.6 2.8 3 3.2 2.8 2.4 1.6 ...
## $ copper : num 85 81 68 81 48 61 31 29 37 24 ...
## $ lead   : num 299 277 199 116 117 137 132 150 133 80 ...
```

²Empleado en la viñeta del paquete `gstat` con el paquete `sp`.

```

## $ zinc    : num  1022 1141 640 257 269 ...
## $ elev    : num  7.91 6.98 7.8 7.66 7.48 ...
## $ dist    : num  0.00136 0.01222 0.10303 0.19009 0.27709 ...
## $ om      : num  13.6 14 13 8 8.7 7.8 9.2 9.5 10.6 6.3 ...
## $ ffreq   : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
## $ soil    : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 1 1 2 ...
## $ lime    : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 1 1 ...
## $ landuse: Factor w/ 15 levels "Aa","Ab","Ag",...: 4 4 4 11 4 11 4 2 2 15 ...
## $ dist.m  : num  50 30 150 270 380 470 240 120 240 420 ...
# ?meuse
# Sistema de coordenadas Rijksdriehoek (RDH) (Netherlands topographical)
# https://epsg.io/28992 # EPSG:28992
meuse_sf <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992, agr = "constant")

# Rio Meuse
data(meuse.riv, package="sp")
str(meuse.riv)

## num [1:176, 1:2] 182004 182137 182252 182315 182332 ...
meuse_riv <- st_sfc(st_polygon(list(meuse.riv)), crs = 28992)

# Rejilla
data(meuse.grid, package="sp")
str(meuse.grid)

## 'data.frame': 3103 obs. of 7 variables:
## $ x      : num 181180 181140 181180 181220 181100 ...
## $ y      : num 333740 333700 333700 333700 333660 ...
## $ part.a: num 1 1 1 1 1 1 1 1 1 ...
## $ part.b: num 0 0 0 0 0 0 0 0 0 ...
## $ dist   : num 0 0 0.0122 0.0435 0 ...
## $ soil   : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 ...
## $ ffreq  : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 ...
meuse_grid <- st_as_sf(meuse.grid, coords = c("x", "y"),
                       crs = 28992, agr = "constant")

# Almacenar
# save(meuse_sf, meuse_riv, meuse_grid, file = "datos/st_meuse.RData")

# Representar
plot(meuse_sf["zinc"], pch = 16, cex = 1.5, main = "",
      breaks = "quantile", key.pos = 4, reset = FALSE)
plot(meuse_riv, col = "lightblue", add = TRUE)
plot(st_geometry(meuse_grid), pch = 3, cex = 0.2, col = "lightgray", add = TRUE)

```

Ejercicio 2.2 (Creación y representación de datos espaciales). Cargar los datos del acuífero Wolfcamp (*aquifer.RData*), generar el correspondiente objeto **sf** y representarlo mostrando los ejes.

2.2.1 Sistemas de referencia de coordenadas

El sistema de referencia de coordenadas (CRS) especifica la correspondencia entre valores de las coordenadas y puntos concretos en la superficie de la Tierra (o del espacio), y resulta fundamental cuando se combinan datos espaciales. En general se consideran dos tipos de CRS:

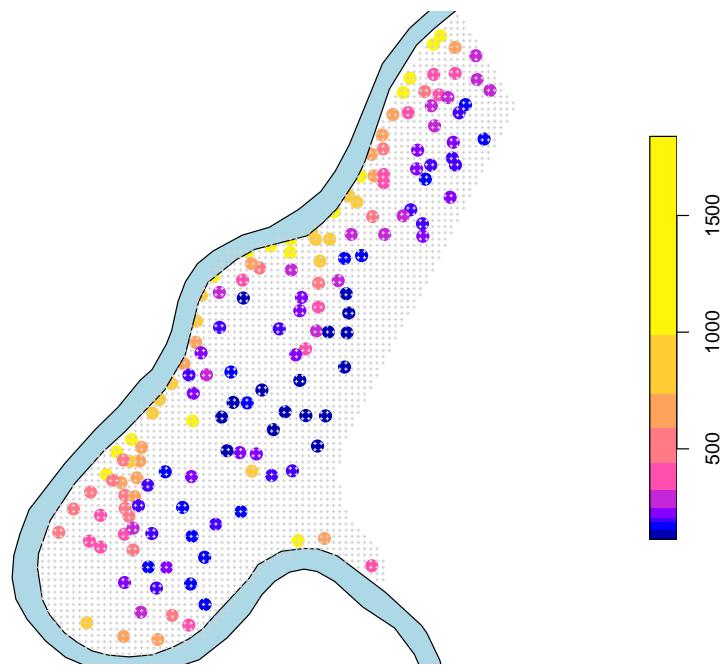


Figura 2.1: Concentración de zinc (ppm) en el entorno del río Meuse (datos ‘sp::meuse’).

- Geodésico: las coordenadas en tres dimensiones (latitud, longitud y altura) se basan en un elipsode de referencia (global o local) que sirve como aproximación del globo terrestre (se tiene en cuenta que no es una esfera perfecta e incluso que puede haber variaciones locales). Este elipsode, junto con información adicional sobre como interpretar las coordenadas (incluyendo el orden y el origen), define el denominado *datum*. Normalmente se asume que las coordenadas son en la superficie terrestre y solo se consideran:
 - latitud: ángulo entre el plano ecuatorial y la línea que une la posición con el centro de la Tierra. Varía desde -90 (polo sur) hasta 90 (polo norte). Un grado equivale aproximadamente a 110.6 km. Los paralelos son las líneas en la superficie terrestre correspondientes a la misma latitud (siendo el 0 el ecuador).
 - longitud: ángulo (paralelo al plano ecuatorial) entre un meridiano de referencia (arco máximo que une los polos pasando por una determinado punto, normalmente el observatorio de Greenwich) y la línea que une la posición con el centro de la Tierra. Varía desde -180 (oeste) hasta 180 (este). Un grado en el ecuador equivale a aproximadamente a 111.3 km. Los meridianos son las líneas en la superficie terrestre correspondientes a la misma longitud (siendo el 0 el meridiano de Greenwich y -180 o 180 el correspondiente antimeridiano).

La rejilla correspondiente a un conjunto de paralelos y meridianos se denomina *graticula* (ver `st_graticule()`).

Uno de los CRS más empleados es el WGS84 (*World Geodetic System 1984*) en el que se basa el *Sistema de Posicionamiento Global* (GPS).

- Proyectado (cartesiano): sistema (local) en dos dimensiones que facilita algún tipo de cálculo (normalmente distancias o áreas). Por ejemplo el UTM (*Universal Transverse Mercator*), que emplea coordenadas en metros respecto a una cuadrícula de referencia (se divide la tierra en 60 husos de longitud, numerados, y 20 bandas de latitud, etiquetadas con letras; por ejemplo Galicia se encuentra en la cuadricula 29T). Se define relacionando estas coordenadas cartesianas con coordenadas geodésicas con un determinado datum.

En `sf` se emplea la librería PROJ para definir el CRS y convertir coordenadas en distintos sistemas³.

³El paquete `sf` admite las últimas versiones PROJ 5 y 6, incluyendo el formato WKT-2 de 2019, mientras que el

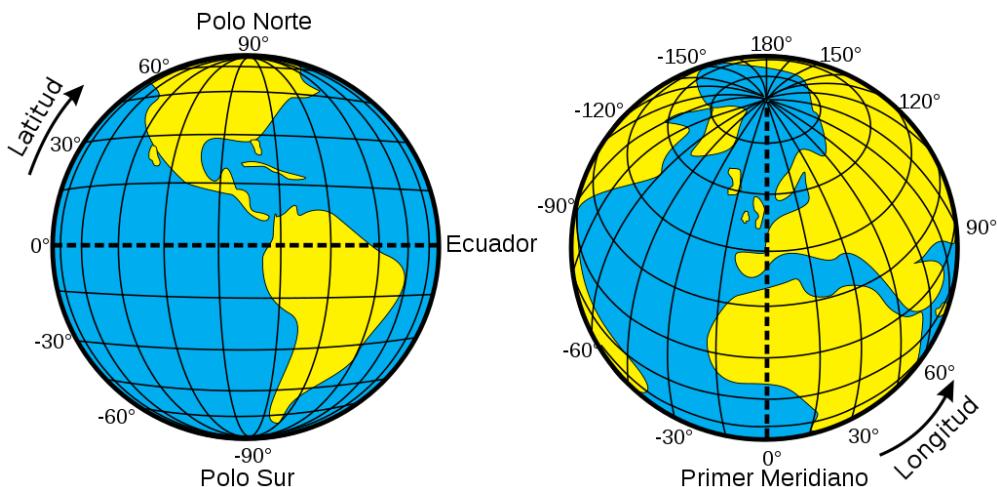


Figura 2.2: Coordenadas geográficas en la superficie terrestre (Fuente Wikimedia Commons).

Para obtener o establecer el CRS se puede emplear la función `st_crs()`. Se puede especificar mediante una cadena de texto que admite GDAL (por ejemplo "WGS84", que se corresponde con el *World Geodetic System 1984*), que típicamente es de la forma ESTÁNDAR:CÓDIGO. El estándar más empleado es el EPSG (*European Petroleum Survey Group*), y es que que da por hecho el paquete `sf` cuando se especifica el CRS mediante un número. También admite el estándar OGC WKT (*Open Geospatial Consortium well-known text*) que es el que emplea internamente, pero resulta complicado manejar en la práctica.

```
st_crs("WGS84")
```

```
## Coordinate Reference System:
##   User input: WGS84
##   wkt:
## GEOCRS["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]],
##        ID["EPSG",4326])
all.equal(st_crs(4326), st_crs("EPSG:4326"), st_crs("WGS84"))
```

```
## [1] TRUE
st_crs(nc)
```

```
## Coordinate Reference System:
##   User input: NAD27
##   wkt:
## GEOCRS["NAD27",
```

paquete `sp` está diseñado para cadenas `PROJ.4 string` que se recomiendan abandonar (las últimas versiones permiten añadir una cadena WKT2 como `comment`).

```

##   DATUM["North American Datum 1927",
##         ELLIPSOID["Clarke 1866",6378206.4,294.978698213898,
##                    LENGTHUNIT["metre",1]],
##         PRIMEM["Greenwich",0,
##                  ANGLEUNIT["degree",0.0174532925199433]],
##         CS[ellipsoidal,2],
##            AXIS["latitude",north,
##                  ORDER[1],
##                  ANGLEUNIT["degree",0.0174532925199433]],
##            AXIS["longitude",east,
##                  ORDER[2],
##                  ANGLEUNIT["degree",0.0174532925199433]],
##            ID["EPSG",4267]

```

En spatialreference.org se puede obtener información detallada sobre una gran cantidad de proyecciones (y permite realizar búsquedas). También puede ser de utilidad epsg.io o este listado con detalles de los parámetros.

El CRS ideal dependerá del tipo de problema y de la zona cubierta por los datos (ver e.g Lovelace et al, 2021, Sección 6.3, para más información). En general en estadística espacial nos interesaría trabajar con coordenadas proyectadas, de forma que tenga sentido emplear la distancia euclídea (algo que puede ser poco o nada razonable si se trabaja con coordenadas geodésicas en una zona muy amplia del globo o cerca de los polos). En el caso de coordenadas sin proyectar (latitud/longitud) puede ser preferible trabajar con distancias ortodrómicas (longitud del arco del círculo máximo que une los puntos, *great circle distances*)⁴. Es importante destacar que cambiar el CRS no reproyecta los datos, hay que emplear `st_transform()` para hacerlo, como se describe en la Sección 2.4.

Finalmente hay que insistir también en que el campo de aplicación de la estadística espacial no se restringe al análisis de datos geográficos (por ejemplo nos puede interesar analizar el desgaste en la pared de un crisol empleado en fundición) y en estos casos los CRS geográficos carecen de sentido. De todos modos habrá que emplear un sistema de coordenadas que permita calcular algún tipo de salto o distancia entre puntos (aunque siempre se pueden considerar coordenadas espaciales tres dimensiones con la distancia euclídea).

2.2.2 Integración con el ecosistema tidyverse

El paquete `sf` es compatible con `tidyverse` y proporciona métodos para interactuar con los paquetes `dplyr`, `tidyr` y `ggplot2`.

Algunos de los métodos de interés para manipular datos espaciales con el paquete `dplyr` son:

- `filter()`, `select()`, `mutate()`, `summarise(..., do_union = TRUE, is_coverage = FALSE)`, `group_by()`, `ungroup()`, etc.
- `inner_join()`, `left_join()`, `right_join()`, `full_join()`, `semi_join()`, `anti_join()`, `st_join()`.
- `st_drop_geometry()`, `st_set_crs()`.

Para detalles ver la referencia.

En el caso del paquete `ggplot2` se puede consultar la referencia y el tutorial *Drawing beautiful maps programmatically with R, sf and ggplot2*:

- Part 1: Basics (General concepts illustrated with the world Map).
- Part 2: Layers (Adding additional layers: an example with points and polygons).
- Part 3: Layouts (Positioning and layout for complex maps).

Por ejemplo, se puede generar un gráfico similar al de la Figura 1.2 (porcentaje de incremento de las defunciones en el año 2020 respecto al 2019 en las CCAA españolas; datos provisionales INE), con el siguiente código:

⁴Algo que ya hace de forma automática el paquete `gstat`.

```

library(dplyr)
library(mapSpain)
mortalidad <- read.csv2("datos/mortalidad.csv")
CCAA_sf <- esp_get_ccaa() %>% left_join(mortalidad) %>%
  mutate(incremento = 100*(mort.2020 - mort.2019)/mort.2019)

library(ggplot2)
ggplot(CCAA_sf) +
  geom_sf(aes(fill = incremento), color = "grey70") +
  scale_fill_gradientn(colors = hcl.colors(10, "Blues", rev = TRUE)) +
  geom_sf_label(aes(label = paste0(round(incremento, 1), "%")), alpha = 0.5) +
  geom_sf(data = esp_get_can_box(), color = "grey70") +
  theme_void()

```

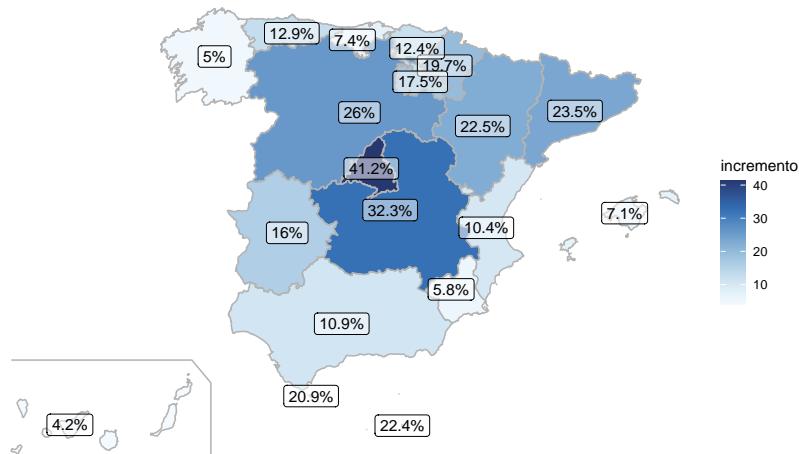


Figura 2.3: Ejemplo de gráfico generado empleando los paquetes ‘dplyr’ y ‘ggplot2’.

[Figura 2.3]

Sin embargo, en estos apuntes se supone que no se está familiarizado con estas herramientas y se evitara su uso (aunque pueden resultar más cómodas después de su aprendizaje). Para una introducción a `dplyr`, ver por ejemplo la viñeta `Introduction to dplyr`, el Capítulo 5 del libro `R for Data Science` o el Capítulo 4 de los apuntes `Prácticas de Tecnologías de Gestión y Manipulación de Datos`.

No obstante, en ciertas ocasiones emplearemos el operador `pipe %>%` (tubería, redirección) por comodidad. Este operador permite canalizar la salida de una función a la entrada de otra. Por ejemplo `segundo(primeros(datos))` se traduce en `datos %>% primero %>% segundo` (facilitando la lectura de expresiones de izquierda a derecha).

2.3 Representación de datos espaciales

El paquete `sf` implementa métodos `plot()` para la representación de objetos espaciales (ver `?plot.sf`). Estos métodos suelen ser la forma más rápida de generar gráficos básicos (estáticos), pero también se pueden emplear otros paquetes como `ggplot2` (Sección 2.2.2), `tmap`, `maps`, `leaflet`, `mapview`, `mapdeck` o `ggmap`, para generar mapas más avanzados, incluyendo mapas dinámicos. Para una introducción a las posibilidades gráficas con el paquete `sf` se puede consultar la viñeta `Plotting Simple Features`.

El método `plot()` es de la forma:

```
plot(x, ..., max.plot, pal = NULL, nbreaks, breaks = "pretty",
      key.pos, key.length, key.width, extent = x, axes = FALSE,
      graticule = NA_crs_, col_graticule = "grey", border, reset = TRUE)
```

- `x`: objeto de tipo `sf` o `sfc`.
- `max.plot`: número máximo de atributos que se representarán.
- `pal`: función que genera la paleta de colores (ver e.g. `?rainbow`), por defecto `sf.colors` (ver Figura 2.4).
- `nbreaks`: número de puntos de corte para la clave de color.
- `breaks`: vector de puntos de corte o cadena de texto válida para el argumento `style` de `classIntervals` (ver figuras: 2.1, 2.4).
- `key.pos`: posición de la leyenda, -1 = automática, 0 = error?, 1 = abajo, 2 = izquierda, 3 = arriba, 4 = derecha, `NULL` = omitir. Cuando se representan múltiples atributos se añade una única leyenda común únicamente si se establece (ver figuras: 2.4, 2.7).
- `key.length, key.width`: dimensiones de la leyenda (proporción de espacio).
- `extent`: objeto con método `st_bbox()` para definir los límites (sustituyendo a `xlim` e `ylim`).
- `axes`: lógico; `TRUE` para dibujar los ejes.
- `graticule`: lógico, objeto de clase `crs` (`st_crs()`) u objeto creado por `st_graticule`; `TRUE` representará la graticula `st_graticule(x)` (ver Figura 2.7).
- `col_graticule`: color de la graticula.
- `border`: color de los bordes de polígonos.
- `reset`: lógico; si el gráfico contiene una leyenda se modifican los parámetros gráficos y por defecto los restaura (`reset = TRUE`). Solo en ese caso es necesario establecer `reset = FALSE` para continuar añadiendo elementos, con `add = TRUE` (para restaurarlos hay que ejecutar `dev.off()`) (ver figuras: 2.1, 2.4).
-: otros parámetros gráficos (ver `?plot.default` y `?par`).

Ejemplo:

```
library(viridis)
plot(nc[c("SID74", "SID79")], pal = viridis, border = 'grey70', logz = TRUE,
      breaks = seq(0, 2, len = 9), at = c(0, 0.5, 1, 1.5, 2),
      key.pos = 1, key.width = lcm(1.2), key.length = 0.8)
```

El paquete `tmap` permite generar mapas temáticos con una gramática similar a la de `ggplot2` pero enfocada a mapas. Por defecto crea mapas estáticos (`tmap_mode("plot")`):

```
library(tmap)
tm_shape(nc) + tm_polygons("SID79")
```

Aunque puede crear mapas interactivos, en páginas html, utilizando el paquete `leaflet` (interfaz a la librería JavaScript Leaflet), implementando también leyendas, ventanas emergentes al pulsar con el ratón en las características y soporte para datos rasterizados.

```
tmap_mode("view")
tmap_last()
# Error en bookdown
```

Para más información ver el capítulo Making maps with R del libro Geocomputation with R, la viñeta del paquete, o el borrador del libro Elegant and informative maps with tmap.

El paquete `mapview` también permite crear mapas interactivos utilizando el paquete `leaflet` (con funcionalidades añadidas) o el paquete `mapdeck` (diseñado para grandes conjuntos de datos espaciales).

```
library(mapview)
mapview(nc, zcol = "SID79")
# Error en bookdown
```

Para más información ver las viñetas del paquete.

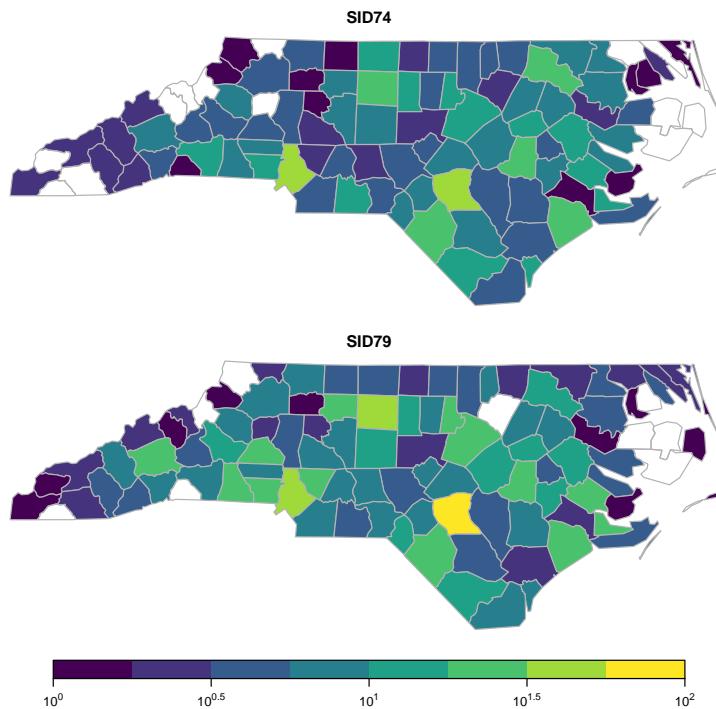


Figura 2.4: Ejemplo de gráfico con múltiples atributos (con colores personalizados y leyenda común, en escala logarítmica personalizada).

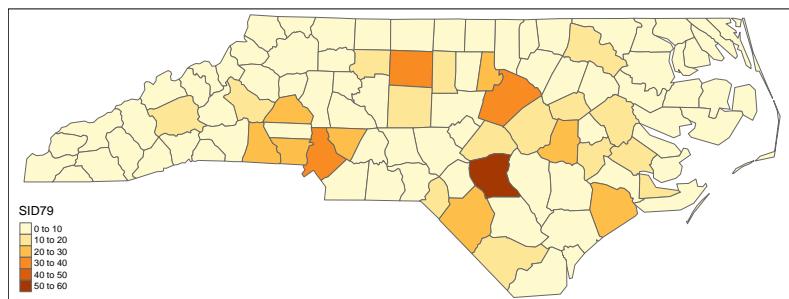


Figura 2.5: Ejemplo de mapa estático creado con ‘tmap’.

2.4 Operaciones con datos espaciales

A continuación se describe una selección de las herramientas disponibles para datos espaciales. Para un listado completo de las funciones implementadas en el paquete `sf` se puede consultar la referencia (o la “chuleta”, aunque puede contener algunos errores).

Puede que algunas herramientas (o recursos) admitan únicamente objetos `Spatial*` del paquete `sp`, aunque siempre se pueden emplear las funciones para convertir tipos de objetos:

- `st_as_sf(x, ...)`: convierte `x` a un objeto `sf` (por ejemplo objetos `Spatial*`).
- `as(x, "Spatial")`: convierte `x` a un objeto `Spatial*`.

2.4.1 Importación y exportación de datos espaciales

El paquete `sf` permite importar y exportar una gran cantidad de formatos de datos espaciales, almacenados en ficheros o en bases de datos, mediante las funciones `st_read()` y `st_write()`. Como se mostró al principio de la Sección 2.2, estas funciones deducen el formato automáticamente a partir de la extensión del archivo:

```
dir <- system.file("shape", package="sf")
list.files(dir, pattern="^*[nc]*")

## [1] "nc.dbf" "nc.prj" "nc.shp" "nc.shx"
# ESRI Shapefile, consta por lo menos de 3 ficheros, el principal .shp
file <- paste0(dir, "/nc.shp")
file

## [1] "C:/Program Files/R/R-4.1.1/library/sf/shape/nc.shp"
nc_sf <- st_read(file)

## Reading layer `nc' from data source
##   `C:\Program Files\R\R-4.1.1\library\sf\shape\nc.shp' using driver `ESRI Shapefile'
## Simple feature collection with 100 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## Geodetic CRS:  NAD27
```

Se admiten los formatos de datos vectoriales soportados por GDAL (que emplea internamente), se puede obtener un listado con la función `st_drivers()`:

```
drivers <- st_drivers()
str(drivers)

## 'data.frame': 89 obs. of 7 variables:
## $ name     : chr "ESRIC" "FITS" "PCIDSK" "netCDF" ...
## $ long_name: chr "Esri Compact Cache" "Flexible Image Transport System" "PCIDSK Database File"
## $ write    : logi FALSE TRUE TRUE TRUE TRUE ...
## $ copy     : logi FALSE FALSE FALSE TRUE TRUE TRUE ...
## $ is_raster: logi TRUE TRUE TRUE TRUE TRUE ...
## $ is_vector: logi TRUE TRUE TRUE TRUE TRUE ...
## $ vsi      : logi TRUE FALSE TRUE FALSE TRUE ...
```

Además, se han desarrollado una gran cantidad de paquetes de R que permiten acceder directamente desde R a datos espaciales. Muchos incluyen conjuntos de datos espaciales y otros implementan interfaces a bases de datos espaciales o geoportales disponibles en Internet. Algunos de ellos son los siguientes:

- `rnatuearth`: permite importar una gran cantidad de datos vectoriales y rasterizados de Natural Earth, incluyendo datos administrativos/culturales (fronteras de países, aeropuertos, carreteras, vías férreas...) y físicos (costas, lagos...).
- `giscoR`: permite importar datos de Eurostat - GISCO (*Geographic Information System of the Commission*).
- `mapSpain`: permite importar límites administrativos de España (CCAA, provincias, municipios...).
- `osmdata`: permite importar “pequeños” conjuntos de datos de OpenStreetMap (OSM).
- `osmextract`: permite importar grandes conjuntos de datos de OSM.
- `ows4R`: (en desarrollo) proporciona una interfaz para *OGC standard Web-Services* (OWS).
- `openeo`: permite importar datos de servidores openEO (*Open Earth Observation data*).
- `rnoaa`: permite importar datos climáticos de la National Oceanic and Atmospheric Administration (NOAA).

- **climaemet**: permite importar datos climáticos proporcionados por la Agencia Estatal de Meteorología de España (AEMET).
- **meteoForecast**: permite importar resultados de los modelos numéricos de predicción meteorológica GFS, MeteoGalicia, NAM y RAP.
- **saqgetr**: permite importar datos de calidad del aire de Europa.
- **RGISTools**: permite importar datos de imágenes de satélite de Landsat, MODIS y Sentinel.
- **maptools,spData,spDataLarge,getlandsat...**

```
library(osmdata)
# Cuidado: descarga mucha información
# https://nominatim.openstreetmap.org/ui/search.html
# https://wiki.openstreetmap.org/wiki/Map_features
osm_coru <- opq('A Coruña') %>%
  add_osm_feature(key = 'highway') %>%
  osmdata_sf()
plot(st_geometry(osm_coru$osm_lines), main = "",
  xlim = c(-8.45, -8.38), ylim = c(43.32, 43.39))
```

[Figura ??]



Figura 2.6: Representación de las carreteras, calles y caminos en A Coruña (generado con el paquete ‘osmdata’).

También están disponibles una gran cantidad de páginas web y geoportales desde donde es posible descargar datos espaciales (algo que se puede hacer directamente desde R). Algunas de ellas son:

- CGADM database of Global Administrative Areas: permite descargar límites administrativos a distintos niveles (e.g. 0 = país, 1 = CCAA, 2 = provincias, 3 = comarcas, 4 = ayuntamientos).
- NASA Earth Science Data.
- INSPIRE Geoportal: *Enhancing access to European spatial data*.
- Copernicus Open Access Hub: *Europe’s eyes on Earth*.

2.4.2 Operaciones con geometrías

Operaciones unarias (operan sobre un único conjunto de geometrías simples, el primer argumento) con resultado geométrico:

- `st_geometry()`: devuelve (o establece) la columna `sfc` de un objeto `sf`.
- `st_transform(x, crs, ...)`: transforma o convierte las coordenadas de `x` a un nuevo sistema de referencia.
- `st_cast(x, to, ...)`: cambia la geometría `x` a otro tipo de geometría.
- `st_centroid()`: devuelve los centroides de las geometrías.
- `st_buffer()`: crea un buffer en torno a la geometría o a cada geometría.
- `st_boundary()`: devuelve la frontera de la geometría.
- `st_convex_hull()`: crea el envoltorio convexo de un conjunto de puntos.
- `st_voronoi()`: crea una teselación de Voronoi.
- `st_make_grid(x, cellsize, offset, n, what = c("polygons", "corners", "centers"))`: genera una rejilla rectangular (o exagonal) de geometrías (`what`) que cubre los límites de `x`.

Como ya se comentó en la Sección 2.2.1, nos puede interesar transformar las coordenadas a un nuevo sistema de referencia (algo necesario para poder combinar conjuntos de datos espaciales con distintos CRS). Por ejemplo podemos utilizar la proyección de Mollweide para representar datos globales (en este caso estimaciones de la población de países; Figura 2.7 derecha).

```
library(rnaturalearth)
par_old <- par(mfrow = c(1, 2), mar = c(bottom = 0, left = 0, top = 0, right = 0))
# NOTA: plot.sf() con escala no es compatible con mfrow
world_pop <- ne_countries(returnclass = "sf")["pop_est"]
plot(world_pop, logz = TRUE, main = "", key.pos = NULL, reset = FALSE)
grat <- st_graticule(crs=st_crs("WGS84"), lon = seq(-180, 180, by = 20), lat = seq(-90, 90, by = 10))
plot(grat[1], col = 'darkgray', add = TRUE)
# https://spatialreference.org/ref/esri/54009/
world_pop2 <- st_transform(world_pop, "ESRI:54009")
plot(world_pop2, logz = TRUE, main = "", key.pos = NULL, reset = FALSE)
grat <- st_graticule(world_pop2, lon = seq(-180, 180, by = 20), lat = seq(-90, 90, by = 10))
plot(grat[1], col = 'darkgray', add = TRUE)
```

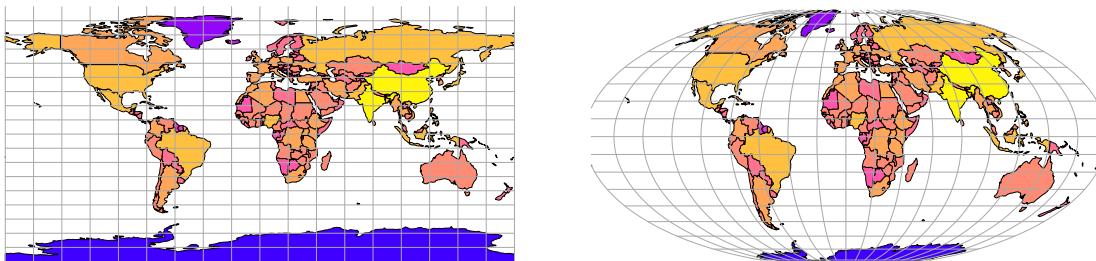


Figura 2.7: Mapa de la población estimada por países (en escala logarítmica), datos sin proyectar (izquierda) y con proyección de Mollweide (derecha).

```
par(par_old)
```

Operaciones binarias (operan sobre dos conjuntos de geometrías simples) con resultado geométrico:

- `st_union(x, y, ..., by_feature)`: une varias geometrías.
- `st_intersection(x, y, ...)`: intersección de pares de geometrías.
- `st_crop(x, y, ..., xmin, ymin, xmax, ymax)`: intersección con rectángulo delimitador o especificado.
- `st_difference(x, y, ...)`: diferencia de pares de geometrías.

- `st_sym_difference(x, y, ...)`: diferencia simétrica (xor) de pares de geometrías.
- `st_nearest_points(x, y, ...)`: obtiene los puntos más cercanos entre pares de geometrías.

Operaciones unarias con resultado numérico o lógico:

- `st_coordinates(x)`: devuelve una matriz con las coordenadas.
- `st_bbox(obj, ...)`: devuelve los límites del conjunto de geometrías.
- `st_area(x, ...)`: devuelve el área de polígonos.
- `st_length(x, ...)`: devuelve la longitud de líneas.
- `st_is(x, type)`: verifica si la geometría es de un determinado tipo o conjunto de clases.

Operaciones binarias con resultado numérico o lógico:

- `st_distance(x, y, ..., by_element, which)`: devuelve la matriz de distancias mínimas entre geometrías.
- `st_nearest_feature(x, y)`: devuelve el índice de la geometría de y más cercana a cada geometría de x .
- `st_intersects(x, y, ...)`: determina si las geometrías se solapan o tocan.
- `st_disjoint(x, y, ...)`: determina si las geometrías no se solapan o tocan.
- `st_touches(x, y, ...)`: determina si las geometrías se tocan.
- `st_overlaps(x, y, ...)`: determina si las geometrías se solapan, pero no están completamente contenidas la una en la otra.
- `st_crosses(x, y, ...)`: determina si las geometrías se cruzan, pero no se tocan.
- `st_within(x, y, ...)`: determina si x está en y .
- `st_contains(x, y, ...)`: determina si y está en x .
- `st_covers(x, y, ...)`: determina si todos los puntos de y están dentro de x .
- `st_covered_by(x, y, ...)`: determina si todos los puntos de x están dentro de y .
- `st_equals(x, y, ...)`: determina si x es geométricamente igual a y .
- `st_equals_exact(x, y, par, ...)`: determina si x es igual a y con cierta tolerancia.

El resultado de las operaciones lógicas es una matriz dispersa (de clase `sgbp`, *sparse geometry binary predicate*), que se puede convertir a una matriz densa con `as.matrix()`.

Ejercicio 2.3 (Creación de rejilla de predicción). Continuando con los datos del Ejercicio 2.2, generar un buffer (`st_buffer()`) de radio 40 en torno a las posiciones espaciales, a partir de él crear una rejilla vectorial (`st_make_grid(..., what = "centers")`) de dimensiones 50 por 50 e intersecarla con el buffer. Representar todos los objetos.

2.5 Análisis exploratorio de datos espaciales

Como se comentó en la Sección 1.4, el primer paso para estimar las componentes del modelo, la tendencia $\mu(\mathbf{s})$ y el semivariograma $\gamma(\mathbf{h})$, es realizar un análisis exploratorio de los datos.

Normalmente comenzaremos por un análisis descriptivo de la respuesta. Sería deseable que su distribución fuese aproximadamente simétrica (de forma que los métodos basados en mínimos cuadrados sean adecuados). Si además la distribución es aproximadamente normal (después de eliminar la tendencia) tendría sentido emplear métodos basados en máxima verosimilitud (Sección 3.3.2). Si su distribución es muy asimétrica se puede pensar en transformarla como punto de partida (aunque podría cambiarse posteriormente dependiendo del modelo final para la tendencia).

```
load("datos/aquifer.RData")
str(aquifer)

## 'data.frame': 85 obs. of 3 variables:
## $ lon : num 42.78 -27.4 -1.16 -18.62 96.47 ...
## $ lat : num 127.6 90.8 84.9 76.5 64.6 ...
## $ head: num 1464 2553 2158 2455 1756 ...
```

```

library(sf)
aquifer_sf <- st_as_sf(aquifer, coords = c("lon", "lat"), agr = "constant")
z <- aquifer_sf$head/100
summary(z)

##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##    10.24   15.48   17.97   20.02   25.40   35.71

hist(z, xlab = "piezometric-head", main = "", freq = FALSE)
lines(density(z), col = 'blue')

```

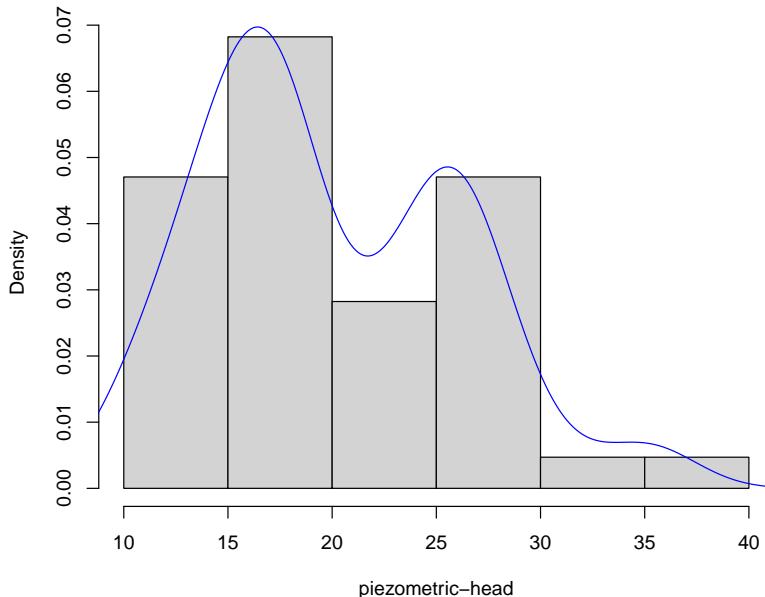


Figura 2.8: Distribución del nivel del agua subterránea en el acuífero Wolfcamp.

En un segundo paso se podría tener en cuenta las coordenadas espaciales. Por ejemplo, podríamos generar un gráfico de dispersión para ver si se observa algún patrón claro (lo que nos haría sospechar que la tendencia no es constante).

```
plot(aquifer_sf, pch = 20, cex = 3, breaks = "quantile", nbreaks = 4)
```

Gráficos de dispersión de la respuesta frente a las coordenadas nos pueden ayudar a determinar si hay una tendencia (al estilo de las funciones `geoR::plot.geodata()` o `npsp::scattersplot()`):

```

coord <- st_coordinates(aquifer_sf)
old.par <- par(mfrow = c(1, 2), oma = c(0.05, 0.95, 0.01, 0.95))
plot(coord[, 1], z, xlab = "x", ylab = "z")
lines(lowess(coord[, 1], z), lty = 2, lwd = 2, col = 'blue')
plot(coord[, 2], z, xlab = "y", ylab = "z")
lines(lowess(coord[, 2], z), lty = 2, lwd = 2, col = 'blue')

par(old.par)

```

En este caso concreto parece que una tendencia lineal es adecuada.

Ejercicio 2.4 (Creación de rejilla de predicción). Realizar un análisis descriptivo del conjunto de

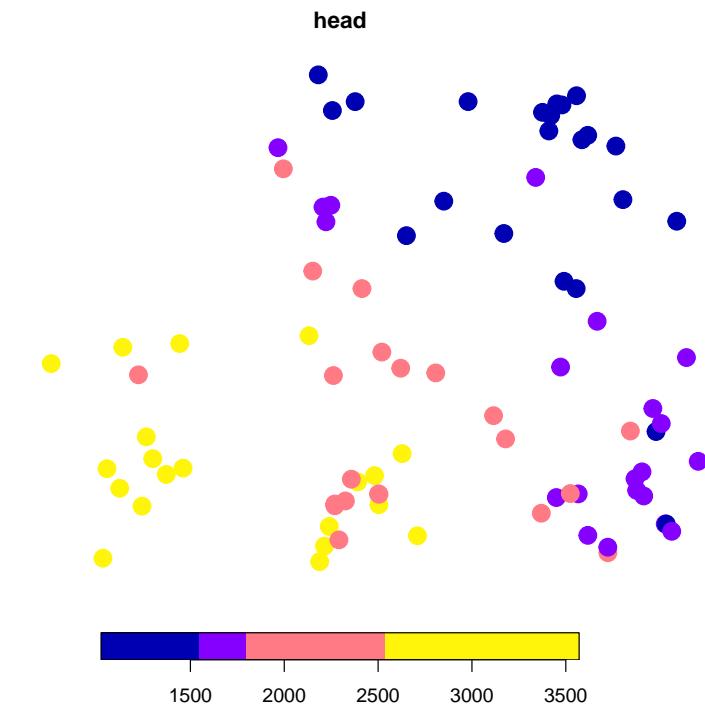


Figura 2.9: Distribución espacial de las observaciones del nivel del agua subterránea en el acuífero Wolfcamp.

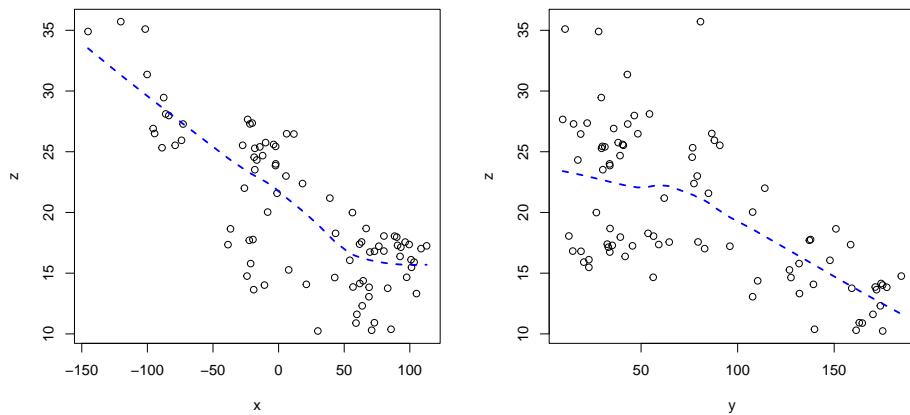


Figura 2.10: Gráficos de dispersión del nivel del agua subterránea frente a coordenadas (acuífero Wolfcamp).

datos `s100` del paquete `geoR` (que contiene una simulación de un proceso espacial estacionario, sin tendencia; ver Sección 3.1).

Realizar un análisis descriptivo del conjunto de datos `meuse_sf` (almacenado en el archivo `st_meuse.RData`) considerando únicamente las variables que comparte con la rejilla `meuse_grid`.

Para el análisis descriptivo de la dependencia se suelen emplear las semivarianzas o los estimadores experimentales del variograma, como se describe en la Sección 3.1.

2.6 El paquete **gstat**

El paquete **gstat** permite la modelización geoestadística (univariante, Capítulo 3, y multivariante, Capítulo 5), espacial y espacio-temporal (Capítulo 6), incluyendo predicción y simulación (Capítulo 4 y secciones 5.X y 6.X).

```
library(gstat)
```

Este paquete implementa su propia estructura de datos (S3, basada en **data.frame**) pero también es compatible con los objetos **Spatial*** del paquete **sp** y los objetos de datos de los paquetes **sf** y **stars**.

Para más información se pueden consultar la referencia, las viñetas del paquete:

- The meuse data set: a tutorial for the gstat R package,
- Spatio-Temporal Geostatistics using gstat,
- Introduction to Spatio-Temporal Variography,

el blog r-spatial o las correspondientes publicaciones (Pebesma, 2004; Gräler, Pebesma y Heuvelink, 2016).

Este paquete de R es una evolución de un programa independiente anterior con el mismo nombre (Pebesma y Wesseling, 1998; basado en la librería GSLIB, Deutsch y Journel, 1992). Puede resultar de interés consultar el manual original para información adicional sobre los detalles computacionales.

Capítulo 3

Modelado de procesos geoestadísticos

En preparación...

Como se comentó en la Sección 1.4 la aproximación tradicional (paramétrica) para el modelado de un proceso geoestadístico, es decir, estimar la tendencia $\mu(\mathbf{s})$ y el semivariograma $\gamma(\mathbf{h})$, consiste en los siguientes pasos:

1. Análisis exploratorio y formulación de un modelo paramétrico (inicial).
2. Estimación de los parámetros del modelo:
 1. Estimar y eliminar la tendencia (suponiendo que no es constante).
 2. Modelar la dependencia (ajustar un modelo de variograma) a partir de los residuos (o directamente de las observaciones si la tendencia se supone constante).
3. Validación del modelo o reformulación del mismo.
4. Empleo del modelo aceptado.

El procedimiento habitual para el modelado de la dependencia en el paso 2 (también denominado *análisis estructural*) consiste en obtener una estimación inicial del semivariograma utilizando algún tipo de estimador experimental (Sección 3.1) y posteriormente ajustar un modelo paramétrico válido de semivariograma a las estimaciones “piloto” obtenidas en el primer paso (secciones 3.2 y 3.3).

El principal problema con esta aproximación aparece cuando no se puede asumir que la tendencia es constante, ya que los estimadores muestrales descritos en la siguiente sección solo son adecuados para procesos estacionarios. En este caso, como la media es constante, entonces:

$$E(Z(\mathbf{s}_1) - Z(\mathbf{s}_2))^2 = 2\gamma(\mathbf{s}_1 - \mathbf{s}_2), \quad \forall \mathbf{s}_1, \mathbf{s}_2 \in D. \quad (3.1)$$

Sin embargo, cuando la tendencia no es constante:

$$E(Z(\mathbf{s}_1) - Z(\mathbf{s}_2))^2 = 2\gamma(\mathbf{s}_1 - \mathbf{s}_2) + (\mu(\mathbf{s}_1) - \mu(\mathbf{s}_2))^2, \quad (3.2)$$

y no es necesariamente función de $\mathbf{s}_1 - \mathbf{s}_2$, ni tiene por qué verificar las propiedades de un variograma. Por este motivo, estos estimadores no deben ser utilizados mientras que no se elimine la tendencia de los datos.

Si no se puede asumir que la tendencia es constante, para poder estimarla de forma eficiente sería necesario conocer la dependencia (i.e. conocer $\gamma(\cdot)$). Este problema circular se suele resolver en la práctica realizando el paso 2 de forma iterativa, como se describe en la Sección 3.4. Otra alternativa sería asumir normalidad y estimar ambos componentes de forma conjunta empleando alguno de los métodos basados en máxima verosimilitud descritos en la Sección 3.3.

Finalmente, en el paso 3, para verificar si el modelo (de tendencia y variograma) describe adecuadamente la variabilidad espacial de los datos (y para comparar modelos), se emplea normalmente la técnica de validación cruzada, descrita en la Sección 4.X del siguiente capítulo (en el que también se describe los principales métodos empleados en el paso 4).

3.1 Estimadores muestrales del semivariograma

Suponiendo que el proceso es intrísecamente estacionario, a partir de (3.1), empleando el método de los momentos, se obtiene el denominado *estimador empírico* (o clásico) del semivariograma (Matheron, 1962):

$$\hat{\gamma}(\mathbf{h}) = \frac{1}{2|N(\mathbf{h})|} \sum_{N(\mathbf{h})} (Z(\mathbf{s}_i) - Z(\mathbf{s}_j))^2, \quad \mathbf{h} \in \mathbb{R}^d,$$

donde:

$$N(\mathbf{h}) = \{(i, j) : \mathbf{s}_i - \mathbf{s}_j \in Tol(\mathbf{h}); i, j = 1, \dots, n\},$$

$Tol(\mathbf{h}) \subset \mathbb{R}^d$ es una región de tolerancia en torno a \mathbf{h} y $|N(\mathbf{h})|$ es el número de pares distintos en $N(\mathbf{h})$. La región de tolerancia debería ser lo suficientemente grande como para que no aparezcan inestabilidades, por lo que se recomienda (Journel y Huijbregts 1978, p.194) que el numero de aportaciones a la estimación en un salto \mathbf{h} sea por lo menos de treinta (i.e. $|N(\mathbf{h})| \geq 30$).

De forma análoga, suponiendo estacionariedad de segundo orden, se obtiene el estimador clásico del covariograma:

$$\hat{C}(\mathbf{h}) = \frac{1}{|N(\mathbf{h})|} \sum_{N(\mathbf{h})} (Z(\mathbf{s}_i) - \bar{Z})(Z(\mathbf{s}_j) - \bar{Z}), \quad \mathbf{h} \in \mathbb{R}^d,$$

siendo $\bar{Z} = \frac{1}{n} \sum_{i=1}^n Z(\mathbf{s}_i)$ la media muestral. El principal problema con este estimador es la necesidad de estimar la media μ del proceso, lo que produce que sea sesgado. Por otra parte, además de que la suposición de estacionariedad de segundo orden es menos general (Sección 1.3, si el proceso es intrísecamente estacionario el estimador del variograma es insesgado y también tiene mejores propiedades cuando la estimación se basa en residuos (aunque en este caso ambos estimadores son sesgados). Más información sobre la distribución y propiedades de estos estimadores se tienen por ejemplo en Cressie (1993, pp. 71-74). Estos resultados justifican que el modelado de la dependencia espacial se realice a través del semivariograma.

Uno de los problemas con el estimador empírico del semivariograma es su falta de robustez frente a observaciones atípicas. Por este motivo se han propuesto numerosas alternativas robustas. Hawkins y Cressie (1984) sugirieron promediar la raíz cuadrada de las diferencias en valor absoluto¹ y posteriormente transformar el resultado a la escala original tratando de obtener un estimador aproximadamente insesgado (utilizando del método delta), obteniéndose el estimador:

$$2\tilde{\gamma}(\mathbf{h}) = \left(\frac{1}{|N(\mathbf{h})|} \sum_{N(\mathbf{h})} |Z(\mathbf{s}_i) - Z(\mathbf{s}_j)|^{\frac{1}{2}} \right)^4 / \left(0.457 + \frac{0.494}{|N(\mathbf{h})|} + \frac{0.045}{|N(\mathbf{h})|^2} \right).$$

Los estimadores locales tipo núcleo son herramientas frecuentemente utilizadas en la estimación de curvas y superficies. Entre los más conocidos podemos señalar los estimadores tipo Nadaraya-Watson y los polinómicos locales (e.g. Fan y Gijbels,1996). Recientemente se han considerado también estas ideas para la estimación del covariograma (e.g. Hall et al., 1994) y del semivariograma. La expresión general de un estimador no paramétrico de un semivariograma isotrópico es de la forma:

$$\hat{\gamma}(r) = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n \omega_{ij}(r) (Z(\mathbf{s}_i) - Z(\mathbf{s}_j))^2}{2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n \omega_{ij}(r)},$$

¹Si el proceso $Z(\cdot)$ es normal entonces $(Z(\mathbf{s}) - Z(\mathbf{s}+\mathbf{h}))^2$ sigue una distribución $2\gamma(\mathbf{h})\chi_1^2$, sin embargo esta distribución es muy asimétrica y la transformación de potencia que hace que se aproxime más a la simetría (normalidad) es la raíz cuarta. Otra ventaja de utilizar la raíz cuadrada de las diferencias es que, en general, están menos correladas que las diferencias al cuadrado (ver p.e. Cressie, 1993, p. 76).

donde $\omega_{ij}(r) \geq 0, \forall i, j$ y $\sum_{i,j} \omega_{ij}(r) > 0$. Dependiendo de la elección de estos pesos obtenemos distintos estimadores:

- $\omega_{ij}(r) = \mathcal{I}_{Tol(r)}(\|\mathbf{s}_i - \mathbf{s}_j\|)$, siendo $Tol(r) \subset \mathbb{R}$ una región de tolerancia en torno a r (y denotando por $\mathcal{I}_A(\cdot)$ función indicadora del conjunto A), obtenemos el estimador clásico del semivariograma.
- $\omega_{ij}(r) = K\left(\frac{\|\mathbf{s}_i - \mathbf{s}_j\|-r}{h}\right)$, es el estimador Nadaraya-Watson (Hall et al., 1994).
- $\omega_{ij}(r) = K\left(\frac{\|\mathbf{s}_i - \mathbf{s}_j\|-r}{h}\right) \times \sum_k \sum_l K\left(\frac{\|\mathbf{s}_k - \mathbf{s}_l\|-r}{h}\right) (\|\mathbf{s}_k - \mathbf{s}_l\| - r) (\|\mathbf{s}_k - \mathbf{s}_l\| - \|\mathbf{s}_i - \mathbf{s}_j\|)$ se obtiene el estimador lineal local del semivariograma (García-Soidán et al., 2003).

La función `variogram()` del paquete `gstat`:

```
variogram(formula, locations = coordinates(data), data, cutoff, width = cutoff/15,
          cressie = FALSE, cloud = FALSE, covariogram = FALSE, ...)
```

permite obtener la nube de semivarianzas (`cloud = TRUE`) y las estimaciones empíricas o robustas (`cressie = TRUE`), además de otras posibilidades.

En primer lugar emplearemos el conjunto de datos `s100` del paquete `geoR`, que contiene una simulación de un proceso espacial estacionario (sin tendencia).

```
# Cargamos los datos y los transformamos a un objeto `sf`
library(sf)
```

```
## Linking to GEOS 3.9.1, GDAL 3.2.1, PROJ 7.2.1
data(s100, package = "geoR")
datos <- st_as_sf(data.frame(s100$coords, z = s100$data),
                  coords = 1:2, agr = "constant")

library(gstat)
vario.cloud <- variogram(z ~ 1, datos, cloud = TRUE, cutoff = 0.6)
vario <- variogram(z ~ 1, datos, cloud = FALSE, cutoff = 0.6)
# Si se quiere tomar el 50% del máximo salto (possible) cutoff = maxlag
# maxlag <- 0.5*sqrt(sum(diff(apply(s100$coord, 2, range))~2))
# maxlag <- 0.5*sqrt(sum(diff(matrix(st_bbox(datos)), nrow = 2, byrow = TRUE))~2))
names(vario)
```

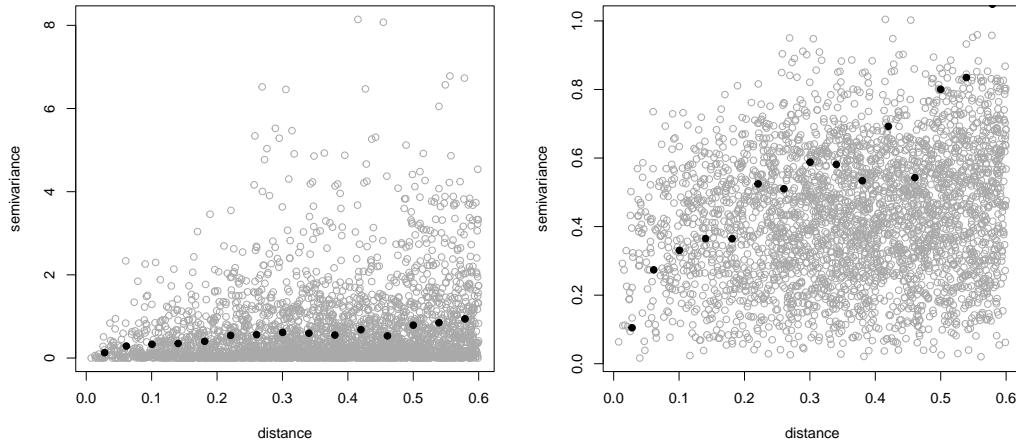
```
## [1] "np"      "dist"     "gamma"    "dir.hor"  "dir.ver"  "id"
```

NOTA: La componente `dist` contiene los saltos, `gamma` las estimaciones del semivariograma (semivarianzas) y `np` el número de aportaciones.

```
rvario.cloud <- variogram(z ~ 1, datos, cloud = TRUE, cressie = TRUE, cutoff = 0.6)
rvario <- variogram(z ~ 1, datos, cloud = FALSE, cressie = TRUE, cutoff = 0.6)

oldpar <- par(mfrow = c(1, 2))
with(vario.cloud, plot(dist, gamma, col = "darkgray",
                       xlab = "distance", ylab = "semivariance"))
with(vario, points(dist, gamma, pch = 19))

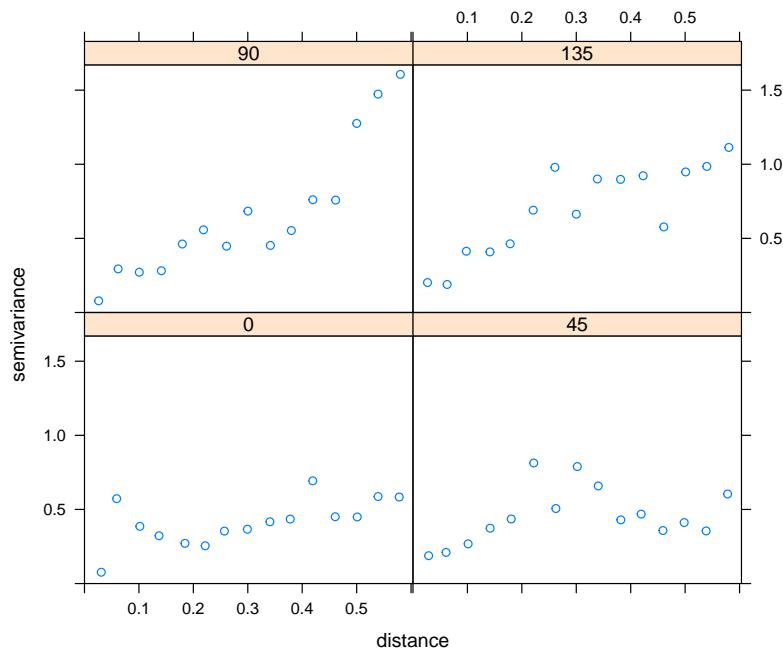
with(rvario.cloud, plot(dist, gamma, col = "darkgray",
                       xlab = "distance", ylab = "semivariance"))
with(rvario, points(dist, gamma, pch = 19))
```



```
par(oldpar)
```

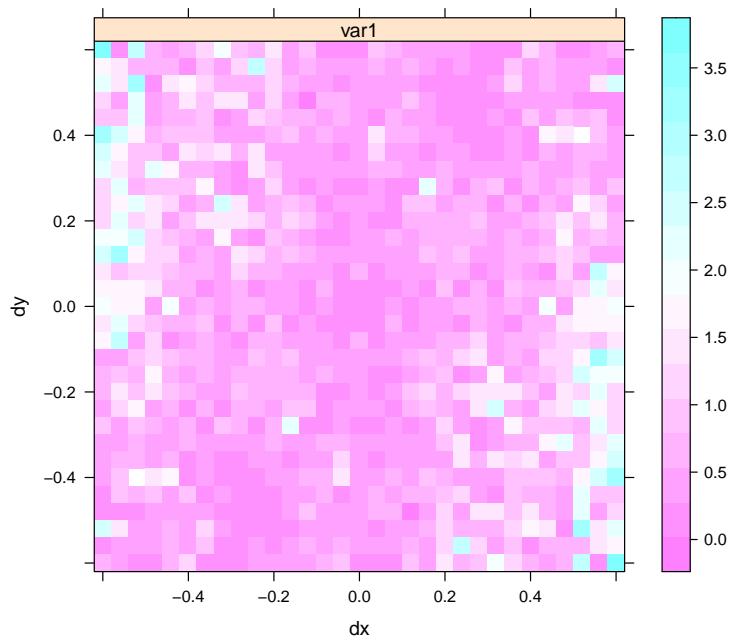
Para un análisis exploratorio de la anisotropía, podemos obtener variogramas direccionales indicando el ángulo y los grados de tolerancia en cada eje:

```
plot(variogram(z ~ 1, datos, cutoff = 0.6, alpha = c(0, 45, 90, 135)))
```



Complementariamente, se puede obtener un mapa de semivarianzas discretizadas en dos dimensiones:

```
variogram.map <- variogram(z ~ 1, datos, cutoff = 0.6, width = 0.6 / 15, map = TRUE)
plot(variogram.map)
```

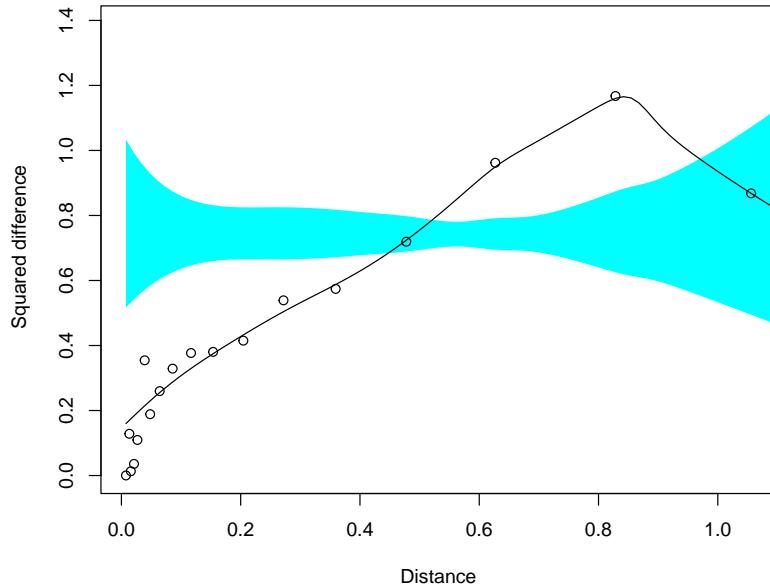


Para estudiar si hay dependencia espacial (estadísticamente significativa) se puede emplear la rutina `sm.variogram` del paquete `sm`. Estableciendo `model = "independent"` devuelve un p-valor para contrastar la hipótesis nula de independencia (i.e. se acepta que hay una dependencia espacial si $p \leq \alpha = 0.05$) y un gráfico en el que se muestra el estimador empírico robusto, un estimador suavizado y una región de confianza para el variograma suponiendo que el proceso es independiente (i.e. consideraríamos que hay dependencia espacial si el variograma suavizado no está contenido en esa región).

```
library(sm)
```

```
## Package 'sm', version 2.2-5.6: type help(sm) for summary information
sm.variogram(s100$coords, s100$data, model = "independent")
```

```
## Test of spatial independence: p = 0.024
```



También se puede realizar contrastes adicionales estableciendo el parámetro `model` a "isotropic" o "stationary".

3.2 Modelos de semivariogramas

Los variogramas deben ser condicionalmente semidefinidos negativos, una propiedad que los estimadores tradicionales normalmente no poseen. Tradicionalmente esto se remedia ajustando un modelo paramétrico válido al estimador muestral (Sección 3.3). En la Sección 3.2.1 se presentan algunos de los modelos isotrópicos tradicionalmente utilizados en geoestadística. Estos modelos son empleados también en ciertos casos como estructuras básicas a partir de las cuales se construyen modelos más complejos, como modelos anisotrópicos (Sección 3.2.2) o los denominados modelos lineales de regionalización (Sección 3.2.3).

3.2.1 Modelos paramétricos isotrópicos

A continuación se presentan algunos de los modelos isotrópicos de semivariograma más utilizados en geoestadística (una revisión más completa se tiene por ejemplo en Chilès y Delfiner, 1999, sección 2.5.1). En la notación utilizada en las parametrizaciones $c_0 \geq 0$ representa el efecto nugget, $c_1 \geq 0$ el umbral parcial (en el caso de variogramas acotados, con $\sigma^2 = c_0 + c_1$) y $a > 0$ el rango (si existe) o el parámetro de escala. En el caso de semivariogramas acotados que alcanzan el umbral asintóticamente (rango infinito), el parámetro a representa el rango práctico, definido como la distancia en la que el valor del semivariograma es el 95% del umbral parcial. En la Figura 3.1 se tienen algunos ejemplos de las formas de algunos de estos semivariogramas.

- Modelo esférico:

$$\gamma(\mathbf{h} | \theta) = \begin{cases} 0 & \text{si } \|\mathbf{h}\| = 0 \\ c_0 + c_1 \left\{ \frac{3}{2} \frac{\|\mathbf{h}\|}{a} - \frac{1}{2} \left(\frac{\|\mathbf{h}\|}{a} \right)^3 \right\} & \text{si } 0 < \|\mathbf{h}\| \leq a \\ c_0 + c_1 & \text{si } \|\mathbf{h}\| > a \end{cases}$$

válido en \mathbb{R}^d , $d = 1, 2, 3$.

- Modelo exponencial:

$$\gamma(\mathbf{h} | \theta) = \begin{cases} 0 & \text{si } \mathbf{h} = \mathbf{0} \\ c_0 + c_1 \left(1 - \exp \left(-\frac{3\|\mathbf{h}\|}{a} \right) \right) & \text{si } \mathbf{h} \neq \mathbf{0} \end{cases}$$

válido en \mathbb{R}^d , $\forall d \geq 1$.

- Modelo racional cuadrático:

$$\gamma(\mathbf{h} | \theta) = \begin{cases} 0 & \text{si } \mathbf{h} = \mathbf{0} \\ c_0 + c_1 \frac{\|\mathbf{h}\|^2}{\frac{1}{19}a^2 + \|\mathbf{h}\|^2} & \text{si } \mathbf{h} \neq \mathbf{0} \end{cases}$$

válido en \mathbb{R}^d , $\forall d \geq 1$.

- Modelo potencial:

$$\gamma(\mathbf{h} | \theta) = \begin{cases} 0 & \text{si } \mathbf{h} = \mathbf{0} \\ c_0 + a \|\mathbf{h}\|^\lambda & \text{si } \mathbf{h} \neq \mathbf{0} \end{cases}$$

con $0 \leq \lambda < 2$ y válido en \mathbb{R}^d , $\forall d \geq 1$. En el caso de $\lambda = 1$ se obtiene el conocido modelo lineal.

- Modelo exponencial-potencial:

$$\gamma(\mathbf{h} | \theta) = \begin{cases} 0 & \text{si } \mathbf{h} = \mathbf{0} \\ c_0 + c_1 \left(1 - \exp \left(-3 \left(\frac{\|\mathbf{h}\|}{a} \right)^\lambda \right) \right) & \text{si } \mathbf{h} \neq \mathbf{0} \end{cases}$$

con $0 \leq \lambda \leq 2$ y válido en \mathbb{R}^d , $\forall d \geq 1$. Cuando $\lambda = 2$ es denominado modelo gausiano; este modelo sin embargo no debería ser utilizado en la predicción espacial debido a las inestabilidades numéricas que produce en los algoritmos kriging (especialmente cuando el efecto nugget es grande; ver p.e. Wackernagel, 1998, pp. 120-123). El modelo exponencial se obtiene también como caso particular cuando $\lambda = 1$.

- Modelo oscilatorio:

$$\gamma(\mathbf{h} | \theta) = \begin{cases} 0 & \text{si } \mathbf{h} = \mathbf{0} \\ c_0 + c_1 \left(1 - \frac{a}{\|\mathbf{h}\|} \sin \left(\frac{\|\mathbf{h}\|}{a} \right) \right) & \text{si } \mathbf{h} \neq \mathbf{0} \end{cases}$$

válido en \mathbb{R}^d , $d = 1, 2, 3$. Este modelo con forma de onda (hay correlaciones negativas) alcanza su valor máximo ($c_0 + 1.218c_1$) cuando $\|\mathbf{h}\| \simeq 4.5a$, siendo a el parámetro de escala.

- Modelo de Matérn (o K-Bessel):

$$\gamma(\mathbf{h} | \theta) = \begin{cases} 0 & \text{si } \mathbf{h} = \mathbf{0} \\ c_0 + c_1 \left(1 - \frac{1}{2^{\nu-1}\gamma(\nu)} \left(\frac{\|\mathbf{h}\|}{a} \right)^\nu K_\nu \left(\frac{\|\mathbf{h}\|}{a} \right) \right) & \text{si } \mathbf{h} \neq \mathbf{0} \end{cases}$$

siendo $\nu \geq 0$ (un parámetro de suavizado) y K_ν la función de Bessel modificada de tercera clase de orden ν (ver p.e. Abramowitz y Stegun, 1965, pp. 374-379). Este modelo es válido en \mathbb{R}^d , $\forall d \geq 1$. El modelo exponencial se obtiene como caso particular cuando $\nu = \frac{1}{2}$ y en el límite $\nu \rightarrow \infty$ el modelo gausiano.

En `gstat` se emplea la función `vgm()` (*Variogram Model*) para definir un modelo de variograma:

```
vgm(psill = NA, model, range = NA, nugget, add.to, anis, kappa = 0.5, ...)
```

- `psill`: umbral parcial (c_1).
- `model`: cadena de texto que identifica el modelo (e.g. "Exp", "Sph", "Gau", "Mat" ...).
- `range`: rango o parámetro de escala (proporcional a a).
- `nugget`: efecto nugget (c_0).

- `kappa`: parámetro de suavizado (ν en el modelo de Matérn).
- `add.to`: permite combinar modelos (Sección 3.2.3).
- `anis`: parámetros de anisotropía (Sección 3.2.2).

Lo habitual es definir un modelo para posteriormente estimar sus parámetros utilizando los empleados en la definición como valores iniciales. También se puede llamar a esta función con el modelo como primer y único argumento, indicando que los parámetros son desconocidos (o que tome los valores por defecto en el ajuste). Si se ejecuta sin argumentos devuelve un listado de todos los modelos:

```
vgm()
```

## short	long
## 1 Nug	Nug (nugget)
## 2 Exp	Exp (exponential)
## 3 Sph	Sph (spherical)
## 4 Gau	Gau (gaussian)
## 5 Exc	Exclass (Exponential class/stable)
## 6 Mat	Mat (Matern)
## 7 Ste	Mat (Matern, M. Stein's parameterization)
## 8 Cir	Cir (circular)
## 9 Lin	Lin (linear)
## 10 Bes	Bes (bessel)
## 11 Pen	Pen (pentaspherical)
## 12 Per	Per (periodic)
## 13 Wav	Wav (wave)
## 14 Hol	Hol (hole)
## 15 Log	Log (logarithmic)
## 16 Pow	Pow (power)
## 17 Spl	Spl (spline)
## 18 Leg	Leg (Legendre)
## 19 Err	Err (Measurement error)
## 20 Int	Int (Intercept)

La función `show.vgms()` genera gráficos con los distintos modelos (por defecto los 17 primeros):

```
show.vgms()
```

```
show.vgms(kappa.range = c(0.1, 0.5, 1, 5, 10), max = 10)
```

```
v1 <- vgm(psill = 1, model = "Exp", range = 0.5, nugget = 0)
v1
```

```
##   model psill range
## 1   Nug     0   0.0
## 2   Exp     1   0.5
plot(v1, cutoff = 3)
```

3.2.2 Modelado de anisotropía

La hipótesis de isotropía simplifica notablemente el modelado de la dependencia espacial por lo que la mayoría de los modelos (básicos) de semivariogramas considerados en geoestadística son isotrópicos (Sección XX). Sin embargo, en muchos casos no se puede asumir que la dependencia es igual en cualquier dirección (uno de los ejemplos más claros es el caso espacio-temporal, donde en principio no es razonable pensar que un salto espacial es equivalente a un salto temporal). En esos casos se suelen considerar ligeras variaciones de la hipótesis de isotropía para modelar la dependencia espacial. En esta sección se comentan brevemente las distintas aproximaciones tradicionalmente consideradas en geoestadística (para más detalles ver p.e. Chilès y Delfiner, 1999, sección 2.5.2, o Goovaerts, 1997, sección 4.2.2), otras aproximaciones adicionales se tratarán en el Capítulo 7 (caso espacio-temporal).

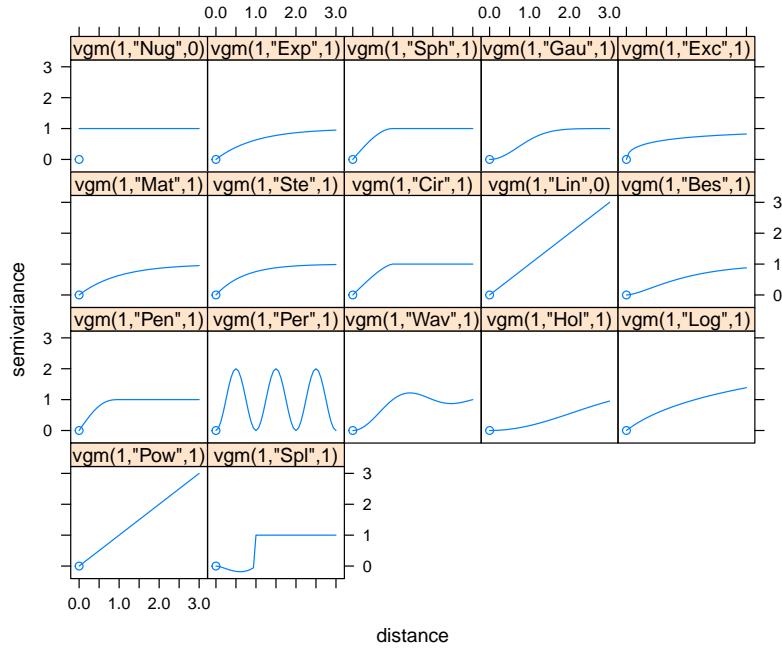


Figura 3.1: Representaciones de los modelos paramétricos isotrópicos de semivariogramas implementados en el paquete ‘gstat’.

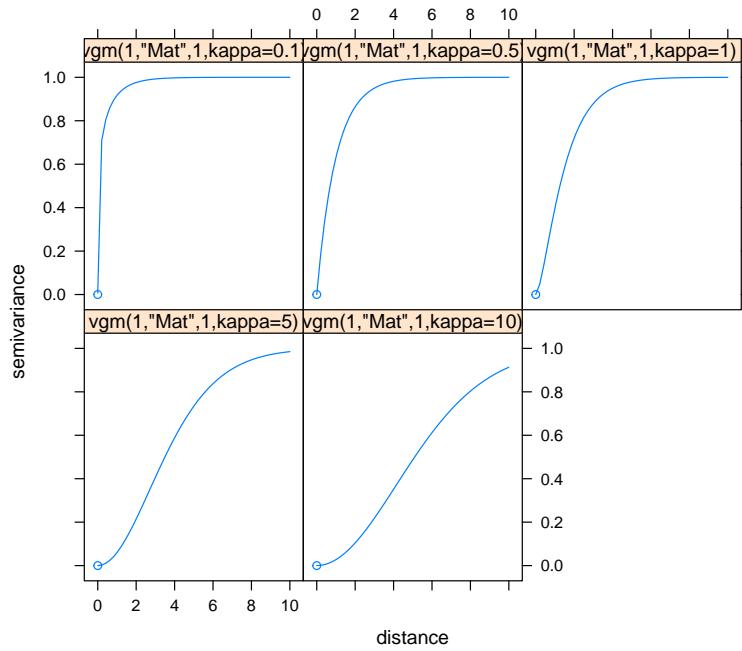


Figura 3.2: Modelo de Matérn con distintos valores del parámetro de suavizado.

Cuando el variograma es función de la dirección además de la magnitud del salto, se dice que el variograma es anisotrópico (no isotrópico). Los tipos de anisotropía habitualmente considerados son:

- *Anisotropía geométrica*: cuando el umbral permanece constante mientras que el rango varía con la dirección.

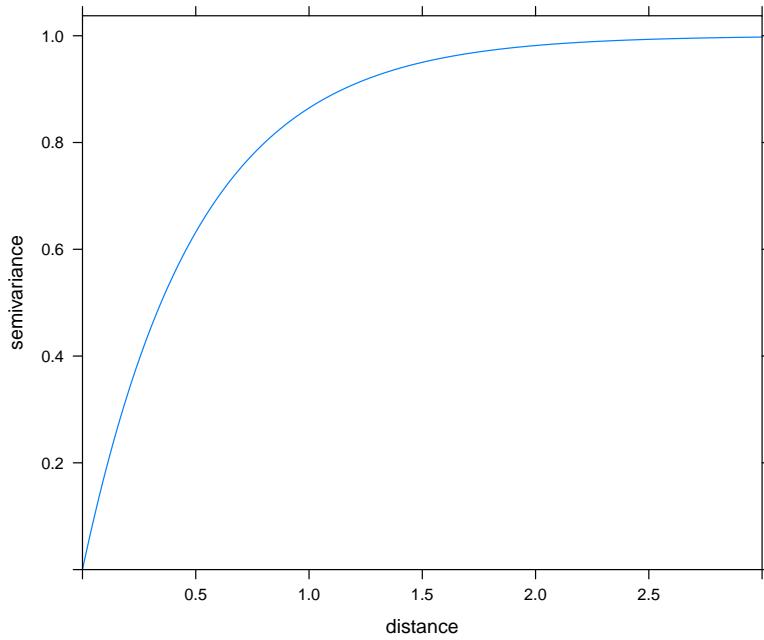


Figura 3.3: Ejemplo de modelo exponencial.

- *Anisotropía zonal*: cuando el umbral del semivariograma varía con la dirección (también se denomina anisotropía estratificada).
- Combinación de las anteriores.

La anisotropía geométrica se puede corregir mediante una transformación lineal del vector de salto \mathbf{h} :

$$\gamma(\mathbf{h}) = \gamma^0(\|\mathbf{Ah}\|), \forall \mathbf{h} \in \mathbb{R}^d,$$

siendo \mathbf{A} una matriz cuadrada $d \times d$ y $\gamma^0(\cdot)$ un semivariograma isotrópico². En este caso se dice que el variograma es *geométricamente anisotrópico*. Por ejemplo, en el caso bidimensional, se suelen considerar una matriz de la forma:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & a_2/a_1 \end{pmatrix} \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix},$$

que se corresponde con las direcciones principales de anisotropía θ y $\theta + \frac{\pi}{2}$ (normalmente se toma θ igual a la dirección de máximo rango). Esto puede extenderse fácilmente para el caso tridimensional (ver p.e. Chilès y Delfiner, 1999, pp. 94-95).

En `gstat` se puede definir anisotropía mediante el argumento `anis` de la función `vgm()`. En dos dimensiones es un vector con dos componentes `aniso = c(alpha, ratio)`, `alpha` es el ángulo para la dirección principal de variabilidad (medido en el sentido del reloj partiendo de la dirección norte) y `ratio` la relación entre el rango mínimo y máximo ($0 \leq ratio \leq 1$).

Ejemplo:

```
v <- vgm(1, "Exp", 5, anis = c(30, 0.1))
str(v)
```

```
## Classes 'variogramModel' and 'data.frame': 1 obs. of 9 variables:
## $ model: Factor w/ 20 levels "Nug","Exp","Sph",...: 2
```

²Esta idea (que el espacio euclídeo no es apropiado para medir distancias entre posiciones espaciales pero una transformación lineal de él sí) ha sido también generalizada para el caso de deformaciones no lineales del espacio. Por ejemplo, Sampson y Guttman (1992) consideraron transformaciones no lineales obtenidas mediante técnicas de escalamiento óptimo multidimensional.

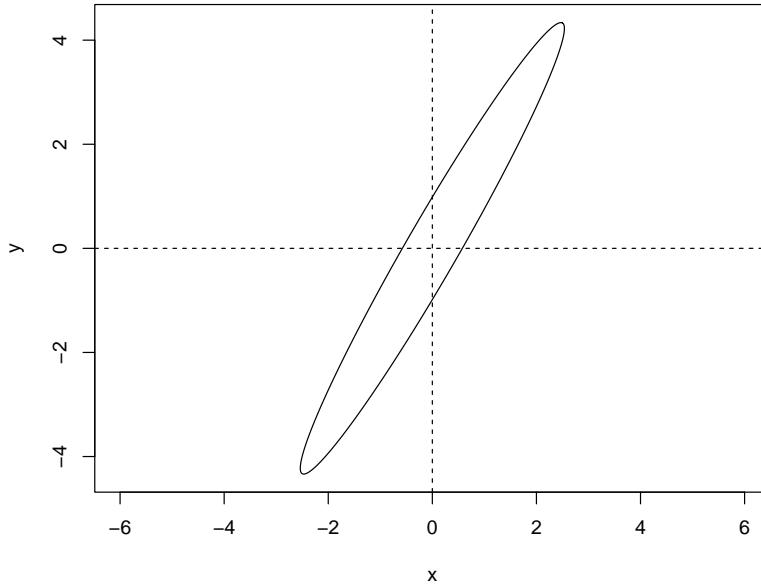
```

## $ psill: num 1
## $ range: num 5
## $ kappa: num 0.5
## $ ang1 : num 30
## $ ang2 : num 0
## $ ang3 : num 0
## $ anis1: num 0.1
## $ anis2: num 1

plot_ellipse_2d <- function(xc = 0, yc = 0, l1 = 10, l2 = 1, phi = pi/3,
                             by = 0.01, asp = 1, ...) {
  # xc, yc: centro
  # l1, l2: longitud ejes
  # phi: angulo del eje 1 respecto al eje x
  t <- seq(0, 2*pi, by)
  x <- xc + l1*cos(t)*cos(phi) - l2*sin(t)*sin(phi)
  y <- yc + l1*cos(t)*sin(phi) + l2*sin(t)*cos(phi)
  plot(x, y, type = "l", asp = asp, ...)
}

with(v, plot_ellipse_2d(l1 = range, l2 = range*anis1,
                       phi = (90 - ang1)*pi/180))
abline(h = 0, lty = 2)
abline(v = 0, lty = 2)

```



En el caso de la anisotropía zonal se suele considerar una combinación de un semivariograma isotrópico más otros “zonales” que depende solamente de la distancia en ciertas direcciones (o componentes del vector de salto). Por ejemplo, en el caso bidimensional, si ϕ es la dirección de mayor varianza se suele considerar una combinación de la forma:

$$\gamma(\mathbf{h}) = \gamma_1(\|\mathbf{h}\|) + \gamma_2(h_\phi),$$

siendo $\gamma_1(\cdot)$ y $\gamma_2(\cdot)$ semivariogramas isotrópicos, y $h_\phi = \cos(\phi)h_1 + \sin(\phi)h_2$ el salto en la dirección ϕ , para $\mathbf{h} = (h_1, h_2) \in \mathbb{R}^2$. Es importante destacar que este tipo de anisotropías pueden causar la aparición de problemas al realizar predicción espacial (ver p.e. Myers y Journel, 1990; y Rouhani y

Myers, 1990), como por ejemplo dar lugar a sistemas kriging no válidos con ciertas configuraciones de los datos. Hay que tener un especial cuidado cuando el covariograma es expresado como suma de covariogramas unidimensionales, en cuyo caso el resultado puede ser únicamente condicionalmente semidefinido positivo sobre un dominio multidimensional.

Este tipo de modelos son casos particulares del modelo lineal de regionalización descrito en la siguiente sección.

Una variante de la anisotropía zonal es el caso de covariogramas separables (también denominados factorizables) en componentes del vector de salto. Por ejemplo, un covariograma completamente separable en \mathbb{R}^2 es de la forma $C(h_1, h_2) = C_1(h_1)C_2(h_2)$, siendo $C_1(\cdot)$ y $C_2(\cdot)$ covariogramas en \mathbb{R}^1 . En este caso se puede pensar que el proceso espacial se obtiene como producto de procesos unidimensionales independientes definidos sobre cada uno de los ejes de coordenadas. Este tipo de modelos se utilizan habitualmente en geoestadística espacio-temporal (aunque no permiten modelar interacciones).

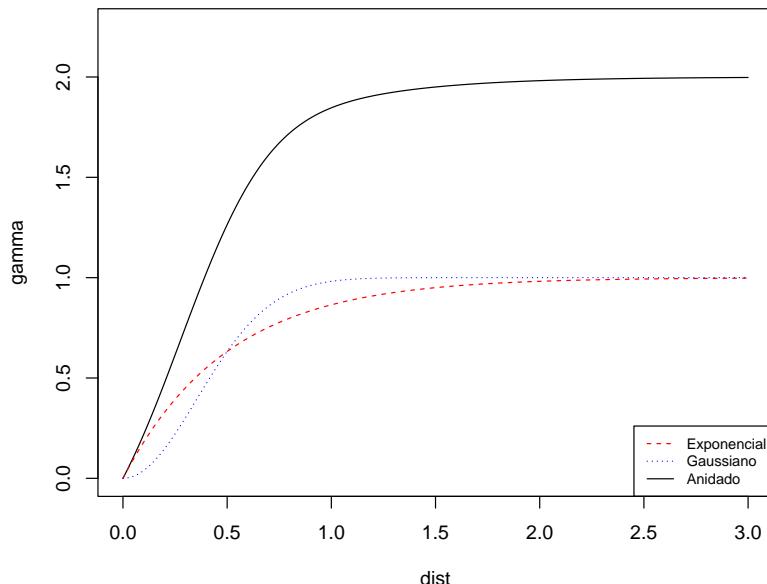
3.2.3 El modelo lineal de regionalización

En gstat se pueden definir modelos de este tipo empleando el parámetro `add.to` de la función `vgm()`.

```
v2 <- vgm(psill = 1, model = "Gau", range = 0.5)
v12 <- vgm(psill = 1, model = "Gau", range = 0.5, add.to = v1)
v12
```

```
##   model psill range
## 1   Nug     0    0.0
## 2   Exp     1    0.5
## 3   Gau     1    0.5

plot(variogramLine(v12, maxdist = 3), type = "l", ylim = c(0, 2.25))
lines(variogramLine(v1, maxdist = 3), col = "red", lty = 2)
lines(variogramLine(v2, maxdist = 3), col = "blue", lty = 3)
legend("bottomright", c("Exponencial", "Gaussiano", "Anidado"), lty = c(2, 3, 1),
       col = c("red", "blue", "black"), cex = 0.75)
```



3.3 Ajuste de un modelo válido

3.3.1 Estimación por mínimos cuadrados

3.3.2 Estimación por máxima verosimilitud

3.3.3 Comentarios sobre los distintos métodos

3.4 Modelado conjunto de la tendencia y del variograma

Si no se puede asumir que la tendencia es constante, para poder estimarla de forma eficiente sería necesario conocer la dependencia (i.e. conocer $\gamma(\cdot)$). Este problema circular se suele resolver en la práctica realizando el paso 2 de forma iterativa, como se describe en la Sección 3.4. Otra alternativa sería asumir normalidad y estimar ambos componentes de forma conjunta empleando alguno de los métodos basados en máxima verosimilitud descritos en la Sección 3.3.2.

Capítulo 4

Predicción Kriging

En preparación...

En este capítulo se comentan brevemente los métodos más conocidos de predicción espacial denominados métodos kriging¹ (ver Sección 1.2.1 para un resumen del origen de esta terminología), centrándonos únicamente en el caso de predicción lineal puntual univariante (el caso multivariante se trata en el Capítulo 5). Una revisión más completa de estos métodos se tiene por ejemplo en Cressie (1993, Capítulo 3 y secciones 5.1, 5.4 y 5.9.1) o Chilès y Delfiner (1999, capítulos 3, 4 y 6).

4.1 Introducción

Si denotamos por $\mathbf{Z} = (Z(\mathbf{s}_1), \dots, z(\mathbf{s}_n))^\top$ valores observados del proceso, los distintos métodos kriging proporcionan un predictor $p(\mathbf{Z}, \mathbf{s}_0)$ de $Z(\mathbf{s}_0)$ verificando que:

- es lineal:

$$p(\mathbf{Z}, \mathbf{s}_0) = \sum_{i=1}^n \lambda_i Z(\mathbf{s}_i) + \lambda_0,$$

- es uniformemente insesgado, para cualquier $\mu(\cdot)$:

$$E(p(\mathbf{Z}, \mathbf{s}_0)) = \mu(\mathbf{s}_0),$$

- y minimiza el error en media cuadrática (e.m.c.) de predicción²:

$$E((p(\mathbf{Z}, \mathbf{s}_0) - Z(\mathbf{s}_0))^2),$$

(al hablar de predicción óptima nos referiremos a que se verifican estas dos últimas condiciones).

Dependiendo de las suposiciones acerca de la función de tendencia $\mu(\cdot)$, se distingue principalmente entre tres métodos kriging:

1. *Kriging simple* (KS): se supone que la media es conocida (algunos autores suponen también que es constante o incluso cero). Además se asume que el covariograma existe y es conocido.
2. *Kriging ordinario* (KO): se supone que la media es constante (i.e. $E(Z(\mathbf{s})) = \mu, \forall \mathbf{s} \in D$) y desconocida. Además se asume que por lo menos existe el variograma y es conocido.

¹Podríamos definir los métodos kriging como algoritmos de predicción de mínimo error en media cuadrática que tienen en cuenta la estructura de segundo orden del proceso.

²Como es bien sabido, en el caso de normalidad el predictor óptimo (tomando como función de pérdidas el error cuadrático) es lineal y va a coincidir con los predictores kriging. Pero si el proceso no es normal no tiene porque serlo, lo que ha motivado el desarrollo de métodos kriging no lineales (ver p.e. Rivoirard, 1994) y del kriging trans-normal (ver sección 4.X).

3. *Kriging universal* (KU; también denominado kriging con modelo de tendencia): se supone que la media es desconocida y no constante, pero que es una combinación lineal (desconocida) de $p + 1$ funciones (o variables explicativas) conocidas $\{x_j(\cdot) : j = 0, \dots, p\}$:

$$\mu(\mathbf{s}) = \sum_{j=0}^p x_j(\mathbf{s})\beta_j$$

donde $\beta = (\beta_0, \dots, \beta_p)^\top \in \mathbb{R}^{p+1}$ es un vector desconocido. Se asume también que por lo menos existe el variograma y es conocido³.

Por simplicidad el kriging ordinario se tratará en este capítulo como un caso particular del kriging universal (aunque en la práctica se suele pensar en el KO como un método distinto al KU, principalmente por ciertos inconvenientes que presenta este último; ver Sección 4.X.X).

4.2 Kriging con media conocida: kriging simple

4.3 Kriging con media desconocida: kriging universal y kriging residual

4.4 Consideraciones acerca de los métodos kriging

³Siempre que una de las funciones explicativas sea idénticamente 1, p.e. $x_0(\cdot) \equiv 1$, en caso contrario las ecuaciones kriging sólo pueden expresarse en función del covariograma.

Capítulo 5

Procesos espaciales multivariantes

En preparación...

Si $\{Z_j(\mathbf{s}) : j = 1, \dots, k; \mathbf{s} \in D\}$ son k procesos espaciales univariantes (y que se suponen en principio interdependientes), el vector

$$\mathbf{Z}(\mathbf{s}) = (Z_1(\mathbf{s}), \dots, Z_k(\mathbf{s}))^\top$$

lo denominaremos proceso espacial multivariante (también se denomina proceso espacial vectorial, campo vectorial espacial o vector regionalizado).

Capítulo 6

Procesos espaciales espacio-temporales

En preparación...

Como ya se comentó en el Capítulo 1 se puede pensar en un procesos espacio-temporal como un caso particular de un proceso espacial en el que una de las componentes es el tiempo. Sin embargo, para enfatizar el carácter temporal, se utilizará una notación de la forma:

$$\{Z(\mathbf{s}, t) : (\mathbf{s}, t) \in D \times T\}$$

donde $D \times T \subset \mathbb{R}^d \times \mathbb{R}^{+,0}$, para referirse a un proceso espacio-temporal.

En algunos casos los procesos espacio-temporales son modelados también como procesos espaciales multivariantes (e.g. Egbert y Lettenmaier, 1986; Kyriakidis y Journel, 1999).

Por ejemplo, se puede considerar una representación de la forma:

$$Z(\mathbf{s}, t) = \mathbf{Z}(\mathbf{s}) = (Z_1(\mathbf{s}), \dots, Z_k(\mathbf{s}))^\top,$$

donde

$$Z_i(\mathbf{s}) = Z(\mathbf{s}, t_i), \quad i = 1, \dots, k.$$

O también:

$$Z(\mathbf{s}, t) = \mathbf{Z}(t) = (Z_1(t), \dots, Z_n(t))^\top,$$

siendo

$$Z_j(t) = Z(\mathbf{s}_j, t), \quad j = 1, \dots, n.$$

Uno de los principales problemas al utilizar estas aproximaciones es que, utilizando los modelos geostadísticos tradicionales, no es posible la predicción en todas las posiciones espacio-temporales sin algún tipo de modelado adicional. Por ejemplo, utilizando la representación y los métodos geoestadísticos de predicción espacial multivariante, se pueden obtener en principio superficies de predicción solamente en los k instantes temporales t_i , $i = 1, \dots, k$, y no es posible la interpolación temporal sin modelado adicional (ver Sección 5.3.5).

Apéndice A

Introducción al paquete sp

El paquete **sp** [Classes and methods for spatial data; E. J. Pebesma y Bivand (2005)] implementa objetos y métodos para datos espaciales...

Introducción, para más detalles ver Bivand et al. (2013)... <https://www.maths.lancs.ac.uk/~rowlings/Teaching/UseR2012/cheatsheet.html>

A.1 Tipos de objetos

```
# Librería sp:classes and methods for spatial data
library(sp) # install.packages('sp')

# Tipos de objetos
getClass("Spatial")

## Class "Spatial" [package "sp"]
##
## Slots:
##
##   ## Name:           bbox proj4string
##   ## Class:          matrix      CRS
##   ##
##   ## Known Subclasses:
##   ## Class "SpatialPoints", directly
##   ## Class "SpatialMultiPoints", directly
##   ## Class "SpatialGrid", directly
##   ## Class "SpatialLines", directly
##   ## Class "SpatialPolygons", directly
##   ## Class "SpatialPointsDataFrame", by class "SpatialPoints", distance 2
##   ## Class "SpatialPixels", by class "SpatialPoints", distance 2
##   ## Class "SpatialMultiPointsDataFrame", by class "SpatialMultiPoints", distance 2
##   ## Class "SpatialGridDataFrame", by class "SpatialGrid", distance 2
##   ## Class "SpatialLinesDataFrame", by class "SpatialLines", distance 2
##   ## Class "SpatialPixelsDataFrame", by class "SpatialPoints", distance 3
##   ## Class "SpatialPolygonsDataFrame", by class "SpatialPolygons", distance 2
```

- Clases del tipo S4 (definición formal con componentes denominadas slots)
- Tipo base: **Spatial**
 - **bbox** (bounding box): matriz con los límites mínimo y máximo de las coordenadas (principalmente para representación gráfica; normalmente se genera automáticamente).

- `proj4string`: cadena de texto que define el sistema de coordenadas de referencia (realmente objeto tipo *CRS*, coordinate reference system) en formato PROJ.4.

```
* CRS(as.character(NA)) para indicar no disponible/faltante
* CRS(+proj=latlong) para coordenadas geográficas
* CRS(+proj=latlong +ellps=WGS84) estándar GPS (World Geodetic System of 1984)
```

```
xbbox <- matrix( c(0,0,1,1), ncol=2)
colnames(xbbox) <- c("min", "max") # Normalmente la bbox se genera automáticamente al crear el objeto
x <- Spatial(xbbox, proj4string = CRS(as.character(NA)))
x
```

```
## An object of class "Spatial"
## Slot "bbox":
##   min max
## [1,] 0 1
## [2,] 0 1
##
## Slot "proj4string":
## CRS arguments: NA
```

Los objetos son del tipo S4. Los componentes se denominan slots. Se acceden con la función `slot()` o el operador `@`.

```
slot(x, 'bbox')

##   min max
## [1,] 0 1
## [2,] 0 1

x@bbox ### en s4 se pone @ en vez de $.
```

```
##   min max
## [1,] 0 1
## [2,] 0 1
```

El paquete sp dispone también de funciones para acceder/establecer los componentes:

```
bbox(x)

##   min max
## [1,] 0 1
## [2,] 0 1

proj4string(x) <- CRS("+proj=latlong +ellps=WGS84") # Importante
```

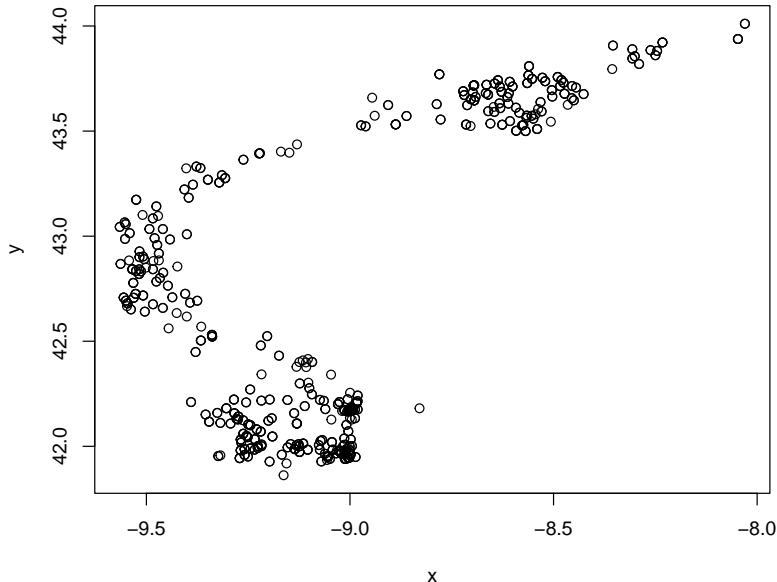
A.1.1 SpatialPoints y SpatialPointsDataFrame

- Tipo `SpatialPoints`
 - Slots: `coords`, `bbox`, `proj4string`
 - Objeto de datos básico para procesos puntuales.
- Tipo `SpatialPointsDataFrame`
 - Slots: `data`, `coords.nrs`, `coords`, `bbox`, `proj4string`
 - Objeto de datos básico para procesos geoestadísticos (y procesos puntuales marcados).

A.1.1.1 Ejemplo SpatialPoints

```
load("datos/caballa.galicia.RData")
str(caballa.galicia) # data.frame(attr(caballa.galicia, "variable.labels"))

## 'data.frame': 676 obs. of 12 variables:
## $ id      : Factor w/ 31 levels "A1","A2","B1",...: 17 17 19 19 19 21 21 23 23 ...
## $ x       : num -9.4 -9.44 -9.44 -9.4 -9.47 ...
## $ y       : num 43 43 43 43 42.8 ...
## $ fecha   : num 1.32e+10 1.32e+10 1.32e+10 1.32e+10 1.32e+10 ...
## $ semana  : num 7 7 7 7 7 8 8 8 8 ...
## $ mes     : num 2 2 2 2 2 2 2 2 2 ...
## $ ano     : num 2001 2001 2001 2001 2001 ...
## $ cpue    : num 18 240 240 18 118 ...
## $ chl_a   : num NA NA 7.08 7.08 7.08 ...
## $ sust_amar: num NA NA 0.356 0.356 0.356 ...
## $ sst     : num 14.2 14.2 16 16 16 16.1 16 15.9 15.9 15.9 ...
## $ lcpue   : num 2.89 5.48 5.48 2.89 4.77 ...
## - attr(*, "variable.labels")= Named chr [1:12] "Cuadricula" "" "" "" ...
## ..- attr(*, "names")= chr [1:12] "id" "x" "y" "fecha" ...
## - attr(*, "codepage")= int 1252
plot(y~x, data = caballa.galicia)
```



```
spt <- SpatialPoints(caballa.galicia[,c("x","y")], proj4string = CRS("+proj=longlat +ellps=WGS84"))
summary(spt)

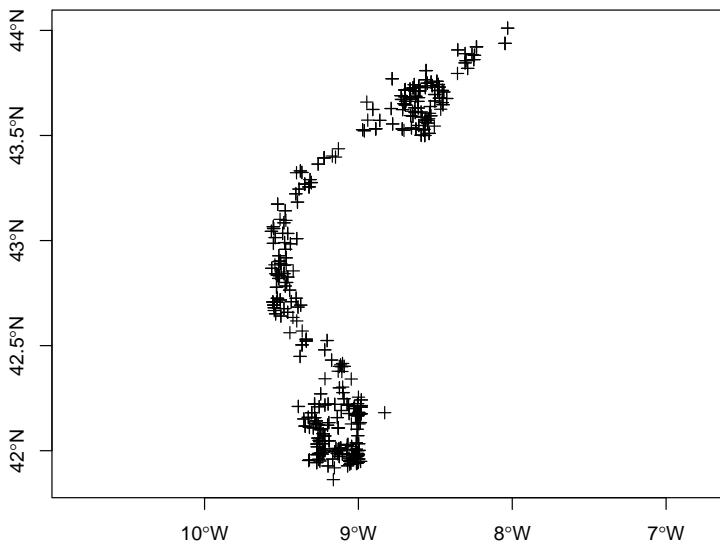
## Object of class SpatialPoints
## Coordinates:
##      min      max
## x -9.56538 -8.030065
## y 41.86240 44.010800
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84 +no_defs]
## Number of points: 676
```

```
str(spt)
```

```
## Formal class 'SpatialPoints' [package "sp"] with 3 slots
##   ..@ coords      : num [1:676, 1:2] -9.4 -9.44 -9.44 -9.4 -9.47 ...
##   ... .- attr(*, "dimnames")=List of 2
##     ... .$. : chr [1:676] "1" "2" "3" "4" ...
##     ... .$. : chr [1:2] "x" "y"
##   ..@ bbox        : num [1:2, 1:2] -9.57 41.86 -8.03 44.01
##   ... .- attr(*, "dimnames")=List of 2
##     ... .$. : chr [1:2] "x" "y"
##     ... .$. : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##     ... .@ projargs: chr "+proj=longlat +ellps=WGS84 +no_defs"
##     ... .$. comment: chr "GEOGCRS[\"unknown\", \n          DATUM[\"Unknown based on WGS84 ellipsoid\", \n          
```

Hay muchos métodos (funciones genéricas) implementados para objetos sp:

```
# plot(spt)
plot(spt, axes=TRUE)
```



A.1.1.2 Ejemplo SpatialPointsDataFrame

Importante (para preparar datos):

```
sdf1 <- SpatialPointsDataFrame(caballa.galicia[,c(2,3)], caballa.galicia[,-c(2,3)], proj4string = CRS(...))
str(sdf1)
```

```
## Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
##   ..@ data      :'data.frame': 676 obs. of 10 variables:
##   ... $. id      : Factor w/ 31 levels "A1","A2","B1",...: 17 17 19 19 19 21 21 23 23 23 ...
##   ... $. fecha   : num [1:676] 1.32e+10 1.32e+10 1.32e+10 1.32e+10 1.32e+10 ...
##   ... $. semana  : num [1:676] 7 7 7 7 7 8 8 8 8 ...
##   ... $. mes     : num [1:676] 2 2 2 2 2 2 2 2 2 ...
##   ... $. ano     : num [1:676] 2001 2001 2001 2001 2001 ...
##   ... $. cpue    : num [1:676] 18 240 240 18 118 ...
##   ... $. chl_a   : num [1:676] NA NA 7.08 7.08 7.08 ...
```

```

## ...$ sust_amar: num [1:676] NA NA 0.356 0.356 0.356 ...
## ...$ sst      : num [1:676] 14.2 14.2 16 16 16 16.1 16 15.9 15.9 15.9 ...
## ...$ lcpue    : num [1:676] 2.89 5.48 5.48 2.89 4.77 ...
## ..@ coords.nrs : num(0)
## ..@ coords     : num [1:676, 1:2] -9.4 -9.44 -9.44 -9.4 -9.47 ...
## ...- attr(*, "dimnames")=List of 2
## ...$ : chr [1:676] "1" "2" "3" "4" ...
## ...$ : chr [1:2] "x" "y"
## ..@ bbox       : num [1:2, 1:2] -9.57 41.86 -8.03 44.01
## ...- attr(*, "dimnames")=List of 2
## ...$ : chr [1:2] "x" "y"
## ...$ : chr [1:2] "min" "max"
## ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
## ...@ projargs: chr "+proj=longlat +ellps=WGS84 +no_defs"
## ...$ comment: chr "GEOGCRS[\"unknown\", \n      DATUM[\"Unknown based on WGS84 ellipsoid\"]",

```

Una alternativa normalmente preferible es modificar directamente el `data.frame`:

```

sdf <- caballa.galicia
coordinates(sdf) <- c("x", "y") # Recomendación
proj4string(sdf) <- CRS("+proj=longlat +ellps=WGS84") # También sdf@proj4string <- CRS("+proj=lon
str(sdf)

```

```

## Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
## ..@ data      :'data.frame': 676 obs. of 10 variables:
## ...$ id       : Factor w/ 31 levels "A1","A2","B1",...: 17 17 19 19 19 21 21 23 23 23 ...
## ...$ fecha    : num [1:676] 1.32e+10 1.32e+10 1.32e+10 1.32e+10 1.32e+10 ...
## ...$ semana   : num [1:676] 7 7 7 7 7 8 8 8 8 ...
## ...$ mes      : num [1:676] 2 2 2 2 2 2 2 2 2 ...
## ...$ ano      : num [1:676] 2001 2001 2001 2001 ...
## ...$ cpue     : num [1:676] 18 240 240 18 118 ...
## ...$ chl_a    : num [1:676] NA NA 7.08 7.08 7.08 ...
## ...$ sust_amar: num [1:676] NA NA 0.356 0.356 0.356 ...
## ...$ sst      : num [1:676] 14.2 14.2 16 16 16.1 16 15.9 15.9 15.9 ...
## ...$ lcpue    : num [1:676] 2.89 5.48 5.48 2.89 4.77 ...
## ..@ coords.nrs : int [1:2] 2 3
## ..@ coords     : num [1:676, 1:2] -9.4 -9.44 -9.44 -9.4 -9.47 ...
## ...- attr(*, "dimnames")=List of 2
## ...$ : chr [1:676] "1" "2" "3" "4" ...
## ...$ : chr [1:2] "x" "y"
## ..@ bbox       : num [1:2, 1:2] -9.57 41.86 -8.03 44.01
## ...- attr(*, "dimnames")=List of 2
## ...$ : chr [1:2] "x" "y"
## ...$ : chr [1:2] "min" "max"
## ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
## ...@ projargs: chr "+proj=longlat +ellps=WGS84 +no_defs"
## ...$ comment: chr "GEOGCRS[\"unknown\", \n      DATUM[\"Unknown based on WGS84 ellipsoid\"]",

```

Operaciones como en un `data.frame`.

```

names(sdf)

## [1] "id"        "fecha"      "semana"     "mes"        "ano"        "cpue"
## [7] "chl_a"     "sust_amar"   "sst"        "lcpue"

sdf$id # Equivalente a sdf$data$id

## [1] E2 E2 F2 F2 F2 G2 G2 H1 H1 I1 I1 I2 I2 J1 J1 J1 J1 J2 J2 J2 D1 E1 F1 F1
## [26] G1 G2 H2 I2 I2 J1 J2 D1 E2 E2 F2 F1 F1 F2 G1 G1 G2 G2 H2 I1 I1 I2 I2 I2
## [51] J1 J1 J2 E2 F1 F1 G1 G2 H1 H2 H2 I1 I1 I1 I2 I2 F1 F2 F2 G2 I1 I1 I1

```

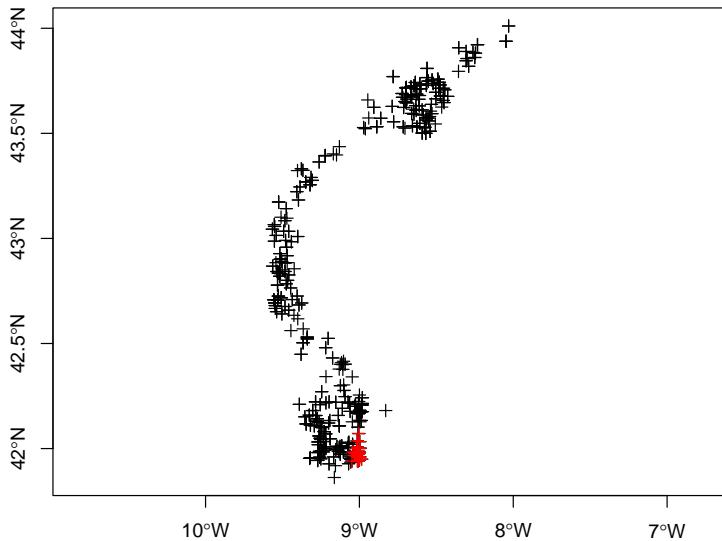
```

## [76] I1 I2 I2 I2 J1 J1 E2 F2 F2 F2 G2 H3 I3 I3 J3 J3 F2 G2 H2 F2
## [101] G2 G2 H2 I1 I1 I2 I2 J2 B2 B3 C2 C3 C3 E2 F2 F2 G2 H3 I3 B3 B3 C3 C3 C3 C4
## [126] C4 C4 I2 I2 I3 J2 J2 H2 H2 I2 I2 E2 F2 F2 G2 B2 B3 B3 C3 C3 C3 C4 C4
## [151] E1 F1 F1 F1 F2 G1 G1 G2 E1 F1 F2 G2 G2 H2 I2 I2 I2 J2 J2 A1 B2 B3 B3 B3 B4
## [176] B4 C3 I2 I2 J2 J2 I2 J2 I2 J3 J3 A1 A2 A2 B3 B4 B4 B5 C2 C3 C3 C4 D2
## [201] D3 F1 F2 C2 C2 C3 C3 C3 C4 D4 H2 H2 H3 I3 B2 B3 C2 C2 C3 C3 C3 C4 B2
## [226] B3 C2 C3 C3 C3 C4 C4 C4 B2 C2 C2 C3 C3 C3 C4 C4 H2 H2 I1 I2 I2 I1
## [251] I2 I2 I2 I2 I3 J1 J2 J2 J2 J3 B2 B3 B3 B3 C3 C3 C3 C4 C4 C4 I2 I2 I3
## [276] I3 J2 J2 J3 J3 B2 C3 C3 C4 C4 B2 B2 B3 B3 C3 C3 C4 C4 B2 B2 B3 C3 C3
## [301] C3 C3 C4 D4 B2 B2 B3 C2 C2 C3 C3 C3 C4 I2 J2 J2 B2 B3 C2 C2 C3 C3 C4
## [326] I3 J2 J2 J3 B3 B3 B4 B4 C3 C4 C4 B2 B3 C3 C3 C4 C4 B3 C3 C3 C4 C4 C4
## [351] B3 B3 B3 B4 B4 C4 G2 H1 H2 H2 B3 B3 B4 C4 C4 H2 H2 I2 I2 H2 I2 J2 B3 B3 B4
## [376] C3 C4 C4 D4 B3 C3 C3 C4 C4 C4 D4 D4 H2 I2 I2 J1 J2 D1 D2 D2 D2 H2 I2 E2
## [401] F2 F2 F2 E2 E2 F2 F2 F2 F2 G2 B2 B2 B3 B3 C3 C3 C3 C4 C1 C2 C3 C3
## [426] C4 D1 D2 D4 C1 C2 C2 C3 C3 C4 I2 I2 J1 J1 J2 J2 I3 H2 H3 I2 I2 I2 I2 I3
## [451] J1 J1 J1 J2 J2 J2 H3 H3 I2 I3 I3 I3 I3 J2 J2 J3 H3 H3 I2 I2 I2 I3
## [476] I3 I3 J2 J2 G3 G3 H2 H2 H2 H2 H3 G3 H2 H2 E1 F1 E1 F1 E1 G2 H1 H2
## [501] H2 H3 I3 C2 D1 D1 D1 D2 D2 E2 E2 F2 B1 B2 C2 C2 C3 D1 D1 D1 D2 E2 E2
## [526] H2 I2 I2 I2 J1 J1 J1 J2 J2 H3 H3 I3 I3 I3 I3 J3 J3 H1 I1 H3 H3 I2 I2 I3 I3
## [551] I3 I3 J2 J2 I2 I3 J2 H3 H3 I3 I3 I3 I3 J3 H3 I3 I3 J3 H3 I2 I2 I3
## [576] I3 I3 I3 J3 J3 H3 I2 I3 I3 J2 J2 J3 F1 F2 G2 G2 H1 H2 H2 F1 G1 G2 I1
## [601] I2 I2 I2 J1 J1 J2 J2 I2 I2 I2 J2 J2 D1 D2 E2 E2 F2 F2 G2 D1
## [626] D1 D1 D1 D2 D2 E1 E2 E2 F1 G1 G2 G2 F1 G1 G1 G2 G2 G1 G2 H3 I2 I3 J2
## [651] J2 J3 J3 H3 I3 I3 I3 J2 J3 J3 J3 H3 I2 I3 I3 I3 I3 J3 J3 J3
## [676] J3

## 31 Levels: A1 A2 B1 B2 B3 B4 B5 C1 C2 C3 C4 D1 D2 D3 D4 E1 E2 F1 F2 G1 ... J3

plot(sdf, axes = TRUE)
plot(sdf[sdf$id == "J3", ], col = "red", add = TRUE)

```



Importante (para análisis descriptivo):

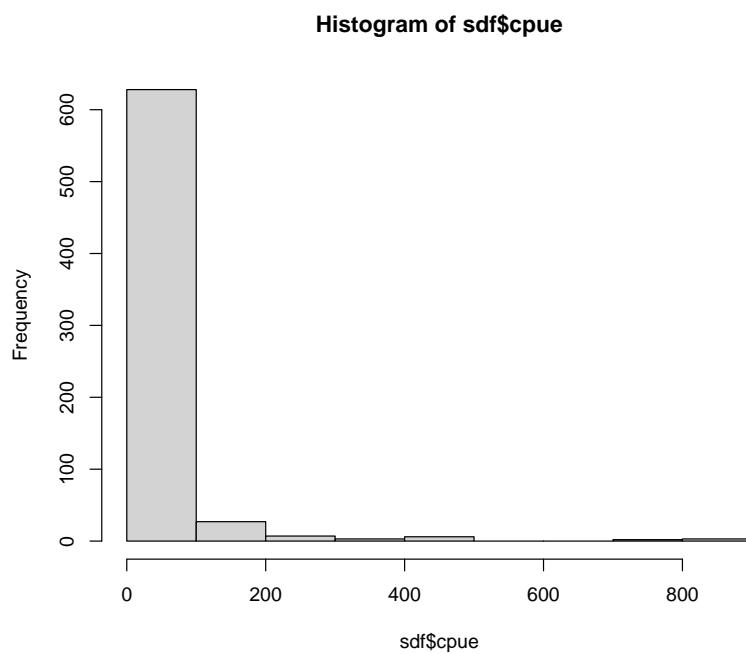
```

summary(sdf[,c("cpue", "lcpue")])

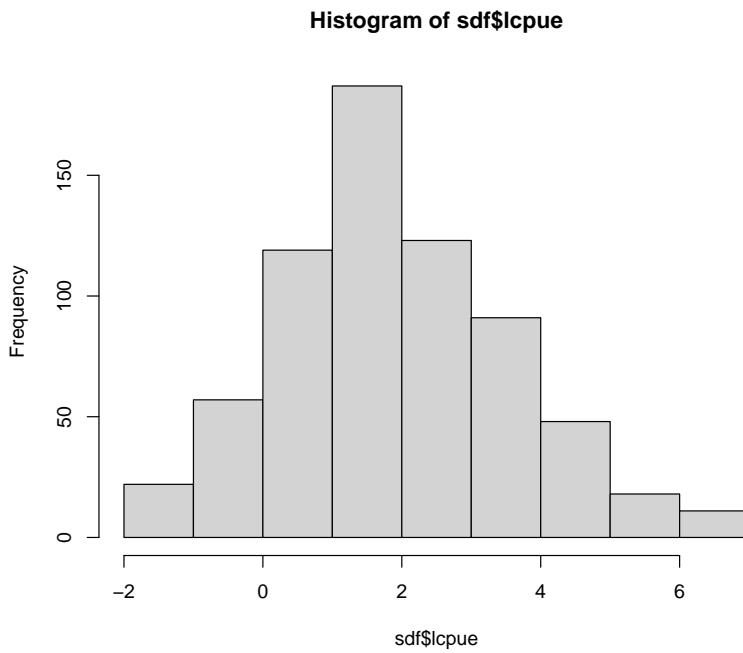
## Object of class SpatialPointsDataFrame

```

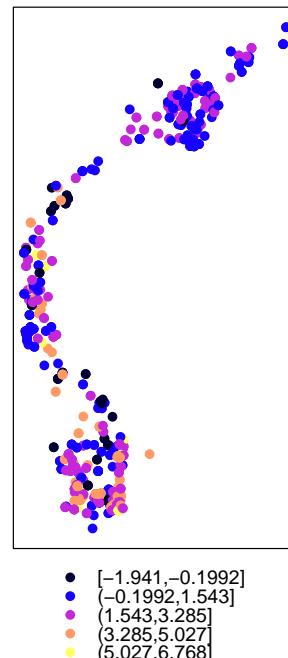
```
## Coordinates:  
##      min      max  
## x -9.56538 -8.030065  
## y 41.86240 44.010800  
## Is projected: FALSE  
## proj4string : [+proj=longlat +ellps=WGS84 +no_defs]  
## Number of points: 676  
## Data attributes:  
##      cpue      lcpue  
## Min.   : 0.1435   Min.   :-1.9411  
## 1st Qu.: 1.9559   1st Qu.: 0.6708  
## Median : 5.8537   Median : 1.7671  
## Mean   : 30.9208  Mean   : 1.9087  
## 3rd Qu.: 19.5349  3rd Qu.: 2.9722  
## Max.   :870.0000  Max.   : 6.7685  
  
hist(sdf$cpue)
```



```
hist(sdf$lcpue)
```



```
spplot(sdf, "lcpue")
```



A.1.2 SpatialLines y SpatialPolygons

- Tipo *SpatialLines*
 - Basados en *Line*: *coords*
 - Se combinan objetos *Line* en listas: *lines*, *bbox*, *proj4string*
 - De utilidad principalmente para representaciones gráficas (y para generar polígonos).
- Tipo *SpatialPolygons*

- Basados en Polygon: `labpt`, `area`, `hole`, `ringDir`, `coords`(extiende Line de forma que la primera y la última línea es la misma).
- Se combinan objetos Polygon en listas: `polygons`, `plotOrder`, `bbox`, `proj4string`.
- De utilidad principalmente para representaciones gráficas (y `overlay`).
- Se extienden también a `Spatial*DataFrame` (slot `data`)
 - `SpatialPolygonsDataFrame`: útil para procesos reticulares.

A.1.2.1 Ejemplo SpatialLines

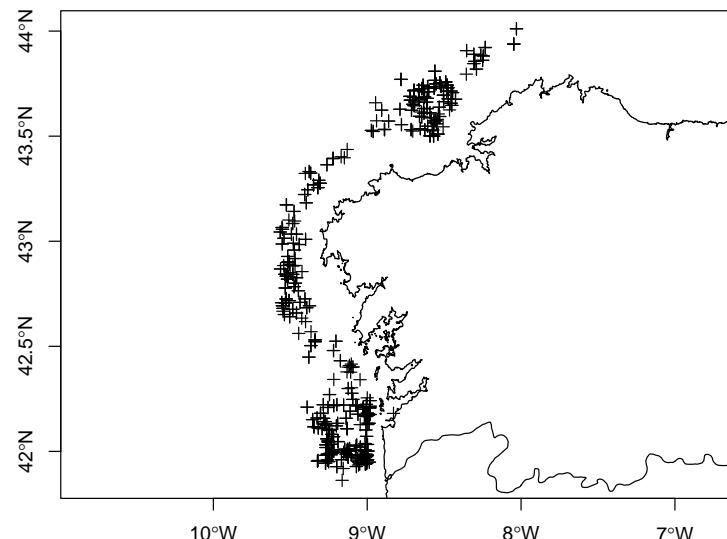
El fichero `costa.galicia.txt` contiene la costa de Galicia en formato Mapgen. Descargada del (difunto) Coastline Extractor

```
library(maptools) # Utilidades para convertir datos entre diferentes formatos espaciales

## Checking rgeos availability: TRUE
costa.galicia <- MapGen2SL("datos/costa.galicia.txt", CRS("+proj=longlat +ellps=WGS84"))

summary(costa.galicia)

## Object of class SpatialLines
## Coordinates:
##       min     max
## x -9.305495 -6.500147
## y 41.500846 43.791944
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84 +no_defs]
plot(sdf, axes=TRUE)
plot(costa.galicia, add=TRUE)
```



A.1.2.2 Ejemplo SpatialPolygonsDataFrame

Los objetos de este tipo se suelen crear a partir de objetos *SpatialLines*, pero hay que asegurarse de que definen adecuadamente un polígono.

Objetos de este tipo se pueden descargar de GADM database of Global Administrative Areas. Contienen límites administrativos a distintos niveles, e.g.:

- *ESP_adm0.rds* límites España e islas
- *ESP_adm1.rds* límites Autonomías
- *ESP_adm2.rds* límites Provincias
- *ESP_adm3.rds* límites Comarcas
- *ESP_adm4.rds* límites Ayuntamientos

NOTA: Se podrían descargar directamente desde R, e.g.:

```
con <- url('http://gadm.org/data/rda/ESP_adm1.rds')
gadm <- readRDS(con)
close(con)
```

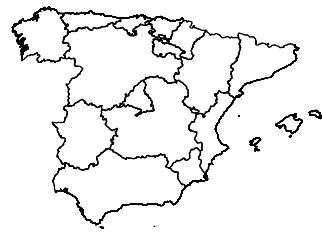
Carga de un objeto *gadm*:

```
gadm <- readRDS("datos/ESP_adm1.rds")
summary(gadm)
```

```
## Object of class SpatialPolygonsDataFrame
## Coordinates:
##      min     max
## x -18.16153 4.328195
## y 27.63736 43.791527
## Is projected: FALSE
## proj4string :
## [+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0]
## Data attributes:
##      OBJECTID      ID_0      ISO      NAME_0
## Min.   : 1.00  Min.   :215  Length:18      Length:18
## 1st Qu.: 5.25  1st Qu.:215  Class :character  Class :character
## Median : 9.50  Median :215  Mode  :character  Mode  :character
## Mean   : 9.50  Mean   :215
## 3rd Qu.:13.75 3rd Qu.:215
## Max.   :18.00  Max.   :215
##
##      ID_1      NAME_1      HASC_1      CCN_1
## Min.   : 1.00  Length:18  Length:18  Min.   : NA
## 1st Qu.: 5.25  Class :character  Class :character  1st Qu.: NA
## Median : 9.50  Mode  :character  Mode  :character  Median : NA
## Mean   : 9.50
## 3rd Qu.:13.75
## Max.   :18.00
## NA's   :18
##
##      CCA_1      TYPE_1      ENGTTYPE_1      NL_NAME_1
## Length:18  Length:18  Length:18  Length:18
## Class :character  Class :character  Class :character  Class :character
## Mode  :character  Mode  :character  Mode  :character  Mode  :character
##
##
```

```
##  VARNAME_1
##  Length:18
##  Class :character
##  Mode   :character
##
## 
## 
## 
## 

plot(gadm)
```



• ↗

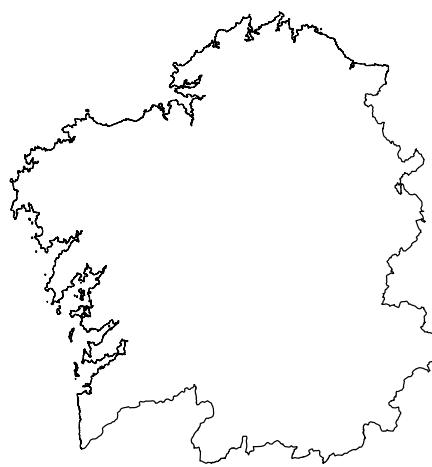
```
# Cuidado objeto muy grande: str(gadm)
# Mejor emplear str(gadm, 3)
```

Extraer autonomía de Galicia:

```
names(gadm)
```

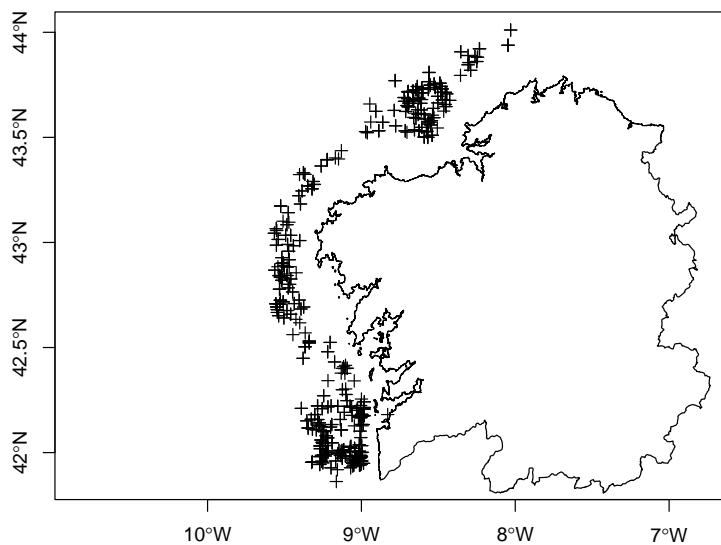
```
## [1] "OBJECTID"    "ID_0"        "ISO"         "NAME_0"       "ID_1"        "NAME_1"
## [7] "HASC_1"       "CCN_1"       "CCA_1"       "TYPE_1"      "ENGTYPE_1"   "NL_NAME_1"
## [13] "VARNAME_1"

galicia <- gadm[gadm$NAME_1 == "Galicia", ]
plot(galicia)
```



Es preferible emplear este tipo de objetos a `SpatialLines`:

```
plot(sdf, axes=TRUE)
plot(galicia, add=TRUE)
```



A.1.3 SpatialGrid y SpatialPixels

Es habitual trabajar con datos espaciales en formato rejilla (grid) (e.g. predicciones en geoestadística):

- Rejilla (posiciones) definida por un objeto `GridTopology`: `cellcentre.offset`, `cellsize`, `cells.dim`
- Tipos `SpatialGrid` y `SpatialPixels`: `grid`, `grid.index`, `coords`, `bbox`, `proj4string`

- Se extienden también a `Spatial*DataFrame` (slot `data`)
- Los objetos `SpatialGrid` se corresponden con la rejilla completa:
- Los objetos `SpatialPixels` se corresponden con una rejilla incompleta
 - `coords` contiene todas las coordenadas (objetos equivalentes a `SpatialPoints`)
 - `grid.index` indices de la rejilla

A.1.3.1 Ejemplo SpatialGrid

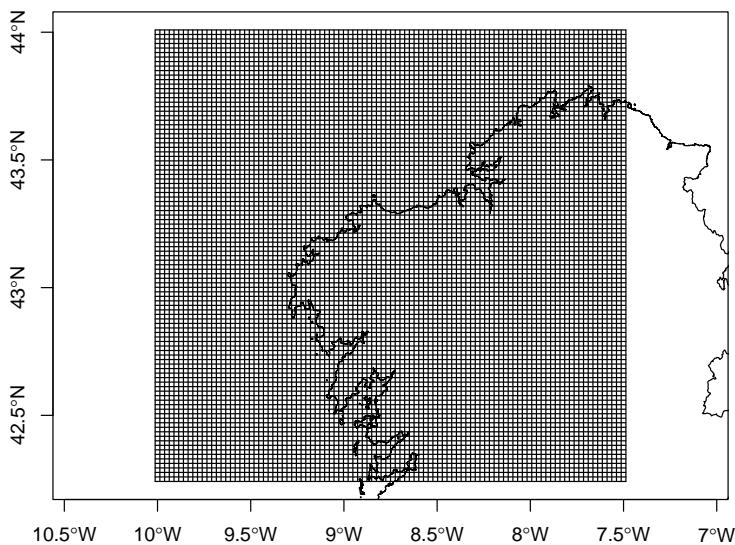
Importante si se utiliza el paquete `gstat...`

```
xrange <- c(-10, -7.5)
yrange <- c(42.25, 44)
nx <- 100
ny <- 100
hx <- diff(xrange)/(nx-1)
hy <- diff(yrange)/(ny-1)

gridtop <- GridTopology(cellcentre.offset = c(min(xrange), min(yrange)),
                         cellsize = c(hx, hy), cells.dim = c(nx, ny))
spgrid <- SpatialGrid(gridtop, proj4string = proj4string(gadm))

str(spgrid)
```

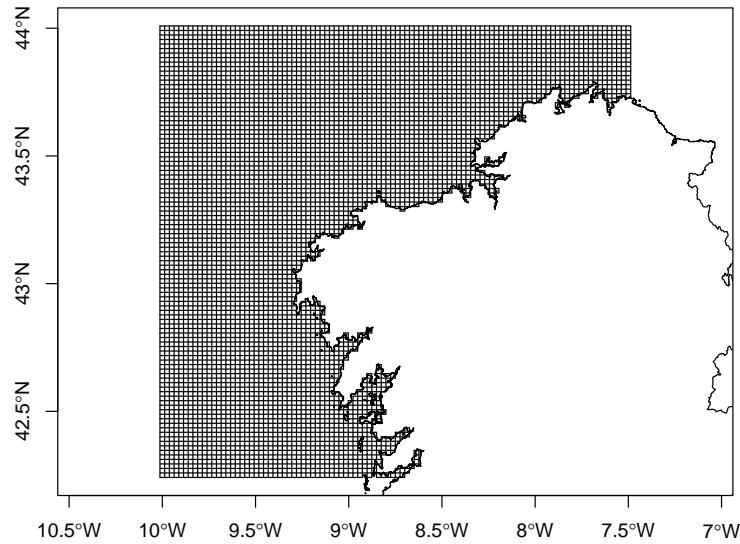
```
## Formal class 'SpatialGrid' [package "sp"] with 3 slots
##   ..@ grid      :Formal class 'GridTopology' [package "sp"] with 3 slots
##     .. . . .@ cellcentre.offset: num [1:2] -10 42.2
##     .. . . .@ cellsize       : num [1:2] 0.0253 0.0177
##     .. . . .@ cells.dim     : int [1:2] 100 100
##   ..@ bbox        : num [1:2, 1:2] -10.01 42.24 -7.49 44.01
##   .. . .- attr(*, "dimnames")=List of 2
##     .. . . .$ : NULL
##     .. . . .$ : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##     .. . . .@ projargs: chr "+proj=longlat +datum=WGS84 +no_defs"
##     .. . . .$ comment: chr "GEOGCRS[\"unknown\" ,\n          DATUM[\"World Geodetic System 1984\" ,\nplot(spgrid, axes = TRUE)
plot(galicia, add = TRUE)
```



```
# over: combinación de objetos espaciales
index <- over(sppgrid, as(galicia, "SpatialPolygons"))
sppix <- as(sppgrid, "SpatialPixels")[is.na(index), ]

str(sppix)

## Formal class 'SpatialPixels' [package "sp"] with 5 slots
##   ..@ grid      :Formal class 'GridTopology' [package "sp"] with 3 slots
##     ... .@ celcentre.offset: num [1:2] -10 42.2
##     ... .@ cellsize       : num [1:2] 0.0253 0.0177
##     ... .@ cells.dim     : int [1:2] 100 100
##     ..@ grid.index : int [1:5631] 1 2 3 4 5 6 7 8 9 10 ...
##     ..@ coords      : num [1:5631, 1:2] -10 -9.97 -9.95 -9.92 -9.9 ...
##     ... .- attr(*, "dimnames")=List of 2
##       ... .$. : NULL
##       ... .$. : chr [1:2] "s1" "s2"
##     ..@ bbox        : num [1:2, 1:2] -10.01 42.24 -7.49 44.01
##     ... .- attr(*, "dimnames")=List of 2
##       ... .$. : chr [1:2] "s1" "s2"
##       ... .$. : chr [1:2] "min" "max"
##     ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##       ... .@ projargs: chr "+proj=longlat +datum=WGS84 +no_defs"
##       ... .$. comment: chr "GEOGCRS[\"unknown\", \n           DATUM[\"World Geodetic System 1984\", \nplot(sppix, axes = TRUE)
plot(galicia, add = TRUE)
```



```
# NOTA: puede ser preferible asignar NA's a variables en SpatialGridDataFrame...
object.size(spgrid)

## 4040 bytes
object.size(sppix)

## 117680 bytes
# Otras funciones:
# as(sppix, "SpatialGrid") reconstruye la rejilla completa
# gridded(ObjetoSpatialPoints) <- TRUE convierte el objeto SpatialPoints en SpatialPixels
```

A.2 Métodos y procedimientos clases sp

Método	Descripción
[selecciona elementos espaciales (puntos, líneas, polígonos, filas/columnas de una rejilla) y/o variables.
\$, [[obtiene, establece o agrega variables (columnas).
coordinates	obtiene o establece (creando objetos sp) coordenadas.
as(obj, clase)	convierte un objeto a una clase.
over,	combina objetos espaciales.
overlay	
spsample	muestrea puntos dentro de una región (rectangular, polígono, línea o rejilla).

A.2.1 Importar/exportar/transformar

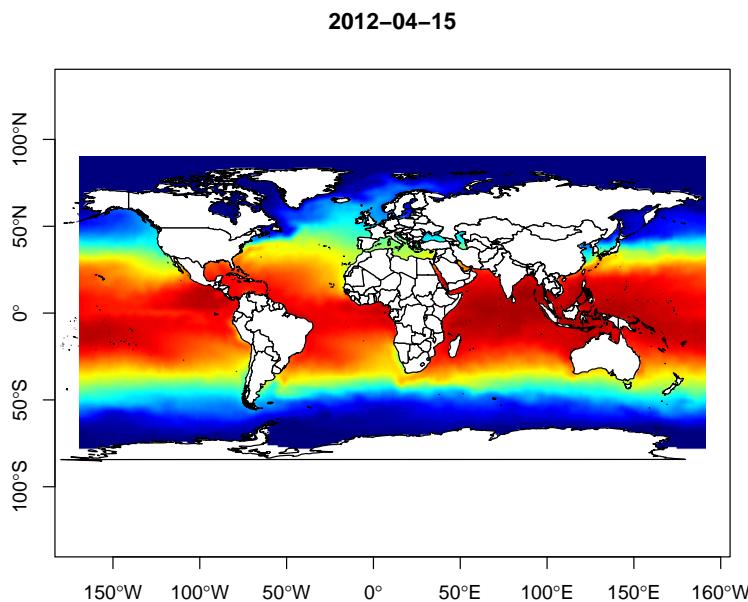
A través de R (paquetes que emplean sp) es fácil exportar datos a otras aplicaciones (e.g. Google Earth).

Ejemplo importación:

Datos descargados en formato NetCDF (Network Common Data Form) de NOAA Optimum Interpolated Sea Surface Temperature V2 (OISST) y procesados con *NOAA_OISST_extraction_v2.R*:

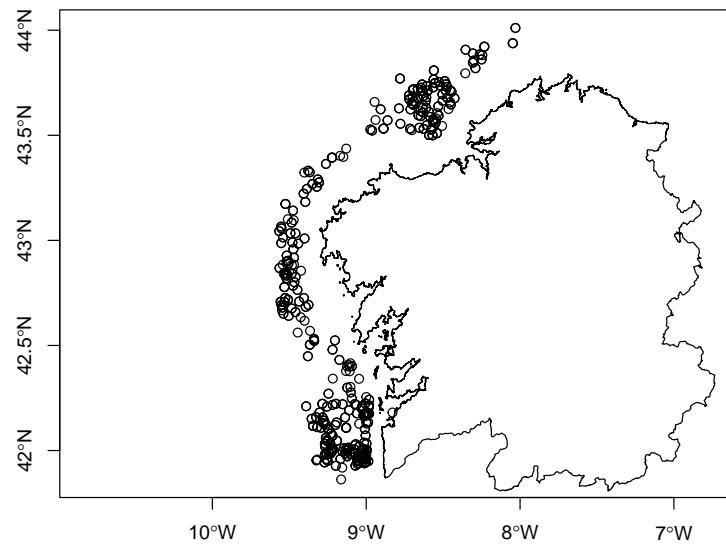
```
load("datos/sstsp.RData") # SST 15-04-2012
jet.colors <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow", "#FF7F00")
image(sstsp, col = jet.colors(128), axes = TRUE)
title(attr(sstsp@data, "label"))

# Ejemplo importar datos de otros paquetes:
library(maps)
library(maptools)
world <- map("world", fill = TRUE, plot = FALSE) # Hay un mapa con mayor resolución en mapdata::wo
world_pol <- map2SpatialPolygons(world, world$names, CRS("+proj=longlat +ellps=WGS84"))
plot(world_pol, col = 'white', add = TRUE)
```



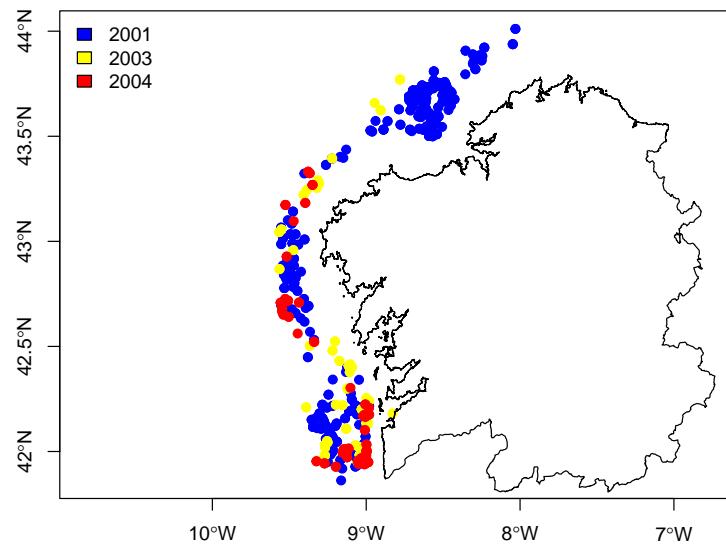
A.3 Representaciones gráficas

```
plot(sdf, axes = TRUE, pch = 1)
plot(galicia, add = TRUE)
```



Color en función de una variable categórica:

```
sdf$ano <- factor(sdf$ano) # convertir año a factor
colores <- c("blue", "yellow", "red")
color <- colores[as.numeric(sdf$ano)]
plot(sdf, axes = TRUE, col = color, pch = 19)
legend("topleft", fill = colores, legend = levels(sdf$ano), bty = "n")
plot(galicia, add = TRUE)
```



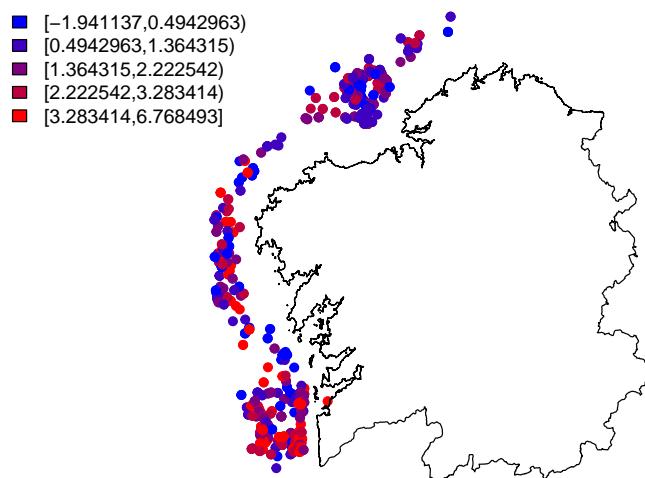
Usando p.e. la función classIntervals del paquete classInt se puede establecer los colores en función de una variable continua:

```
library(classInt) # install.packages('classInt')

class.int <- classIntervals(sdf$lcpue, n = 5, style = "quantile")
pal <- c("blue", "red")
# plot(class.int, pal = pal)

class.col <- findColours(class.int, pal = pal)

plot(sdf, col = class.col, pch = 19)
legend("topleft", fill = attr(class.col, "palette"),
       legend = names(attr(class.col, "table")), bty = "n")
plot(galicia, add = TRUE)
```



```
# methods(image) para rejillas
# ver tambien splot, simage,... en library(npsp)
```

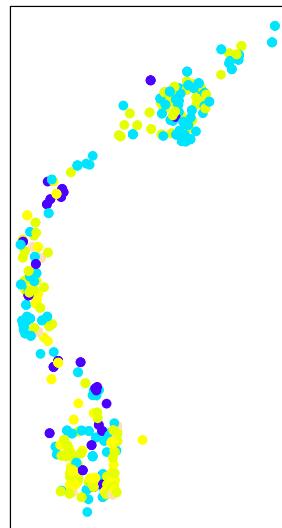
A.3.2 Gráficos lattice: spplot

- Ventajas: “Ideales” para las clases sp (para gráfico automáticos...)
- Inconveniente: los gráficos lattice requieren mayor tiempo de aprendizaje (dificultades para personalizarlos...)

```
library(lattice)

spplot(sdf, "lcpue", main = "CPUE (escala logarítmica)",
       col.regions = topo.colors(6), cuts=5)
```

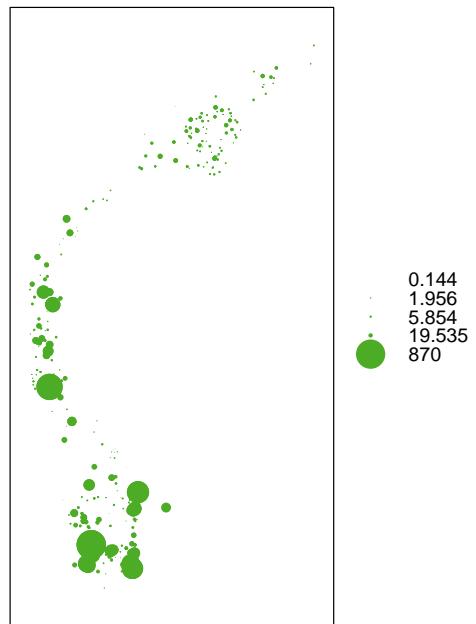
CPUE (escala logarítmica)



- [-1.941,-0.1992]
- (-0.1992,1.543]
- (1.543,3.285]
- (3.285,5.027]
- (5.027,6.768]

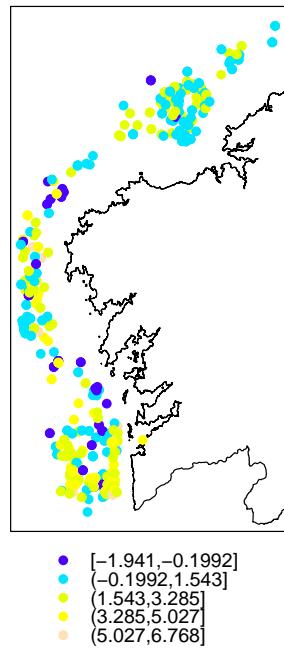
```
bubble(sdf, "cpue", main = "CPUE")
```

CPUE



Añadir perfil de Galicia:

```
sp.layout <- list("sp.polygons", galicia) # Para añadir elementos se utiliza el parámetro sp.layout
spplot(sdf, "lcpue", main = "CPUE (escala logarítmica)",
       col.regions = topo.colors(6), cuts = 5, sp.layout = sp.layout )
```

CPUE (escala logarítmica)

Alternativamente gráficos ggplot (ggplot2) con el paquete ggspatial...

Apéndice B

Introducción al paquete geoR

El paquete **geoR** proporciona herramientas para el análisis de datos geoestadísticos en R (otra alternativa es el paquete **gstat**, por ejemplo...). A continuación se ilustran algunas de las capacidades de este paquete.

B.1 Inicio de una sesión y de carga de datos

Después de iniciar la sesión R, cargar **geoR** con el comando **library** (o **require**). Si el paquete se carga correctamente aparece un mensaje.

```
library(geoR)

## -----
## Analysis of Geostatistical Data
## For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
## geoR version 1.8-1 (built on 2020-02-08) is now loaded
## -----
```

B.1.1 Archivos de datos

Normalmente, los datos se almacenan como un objeto (una lista) de la clase **geodata**. Un objeto de esta clase contiene obligatoriamente dos elementos:

- **\$coords**: las coordenadas de las posiciones de los datos.
- **\$data**: los valores observados de la variables.

Opcionalmente pueden tener otros elementos, como covariables y coordenadas de las fronteras de la zona de estudio.

Hay algunos conjuntos de datos incluidos en el paquete de distribución.

```
# data()                      # lista todos los conjuntos de datos disponibles
# data(package = "geoR")       # lista los conjuntos de datos en el paquete geoR

data(wolfcamp)                 # carga el archivo de datos wolfcamp
summary(wolfcamp)

## Number of data points: 85
##
## Coordinates summary
##      Coord.X   Coord.Y
## min -233.7217 -145.7884
## max  181.5314  136.4061
```

```
##
## Distance summary
##      min          max
## 0.3669819 436.2067085
##
## Data summary
##   Min. 1st Qu. Median     Mean 3rd Qu. Max.
## 312.1095 471.8218 547.7156 610.2845 774.1778 1088.4209
```

Se pueden importar directamente un archivo de datos en formato texto:

```
ncep <- read.geodata('ncep.txt', header = FALSE, coords.col = 1:2, data.col = 4)
# plot(ncep)
# summary(ncep)
```

También se puede convertir un `data.frame` a un objeto `geodata`:

```
ncep.df <- read.table('ncep.txt', header = FALSE)
names(ncep.df) <- c('x', 'y', 't', 'z')
# str(ncep.df)
# Nota: los datos son espacio-temporales, pero geor sólo admite datos 2D

datgeo <- as.geodata(ncep.df, coords.col = 1:2, data.col = 4)
# plot(datgeo)
# summary(datgeo)
```

O objetos de datos espaciales (entre ellos los compatibles del paquete `sp`), por ejemplo el siguiente código crea un objeto `SpatialPointsDataFrame` y lo convierte a `geodata`:

```
library(sp)
load("caballa.galicia.RData")
coordinates(caballa.galicia) <- c("x", "y")
proj4string(caballa.galicia) <- CRS("+proj=longlat +ellps=WGS84")

datgeo <- as.geodata(caballa.galicia["lcpue"])
# Problemas con coordenadas duplicadas (ver ?duplicated)
# plot(datgeo)
# summary(datgeo)
```

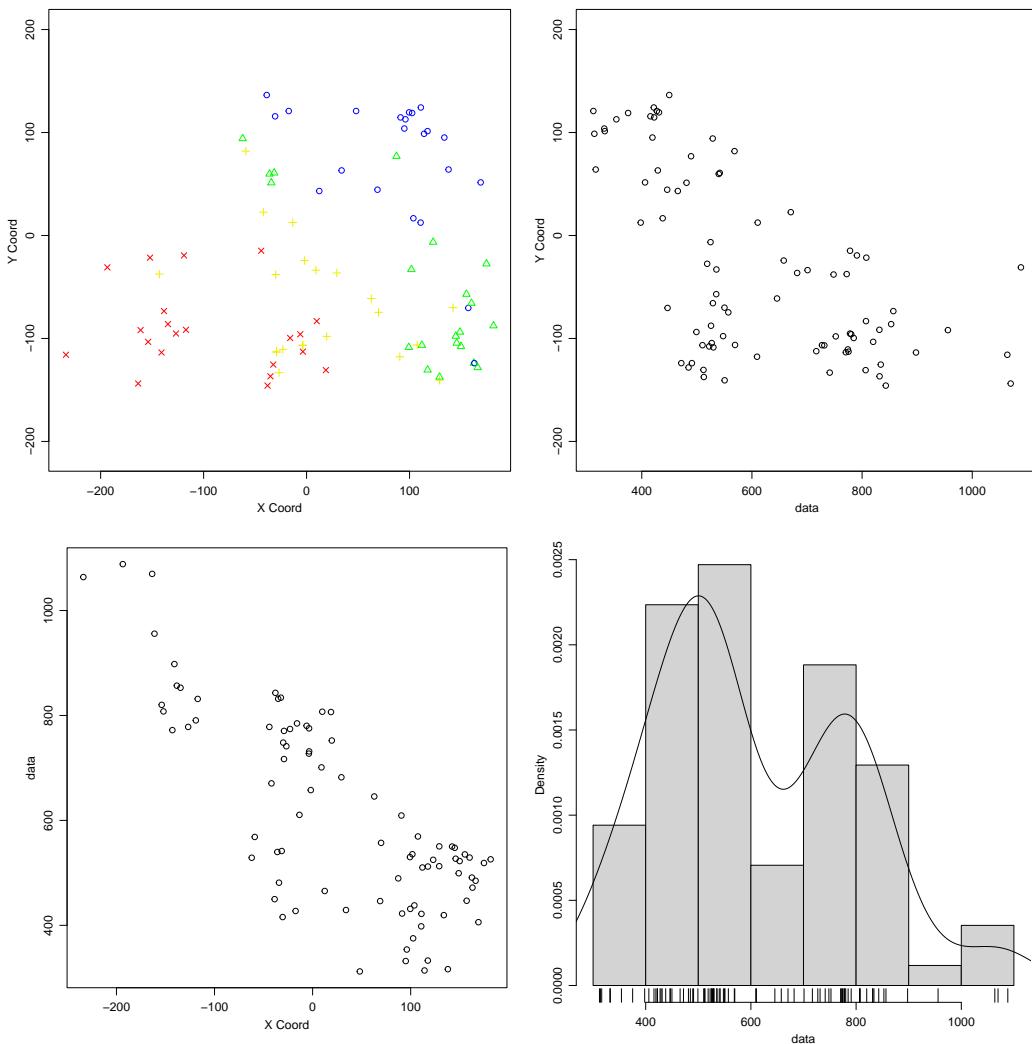
En la documentación de las funciones `as.geodata` y `read.geodata` hay más información sobre cómo importar/convertir datos.

B.2 Análisis descriptivo de datos geoestadísticos

Como se mostró anteriormente, el método `summary` proporciona un breve resumen descriptivo de los datos (ver `?summary.geodata`).

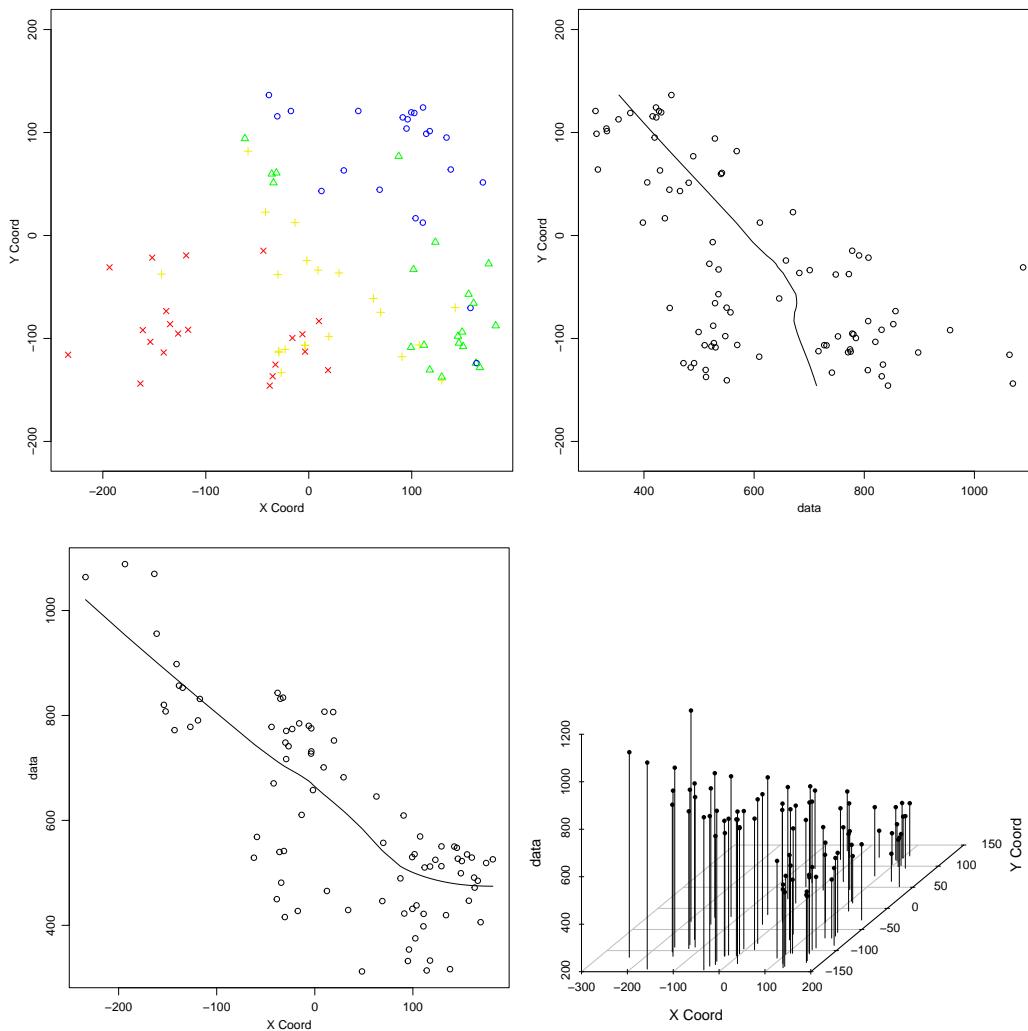
La función `plot()` genera por defecto gráficos de los valores en las posiciones espaciales (distinguiendo según cuartiles), los datos frente a las coordenadas y un histograma de los datos:

```
plot(wolfcamp)
```



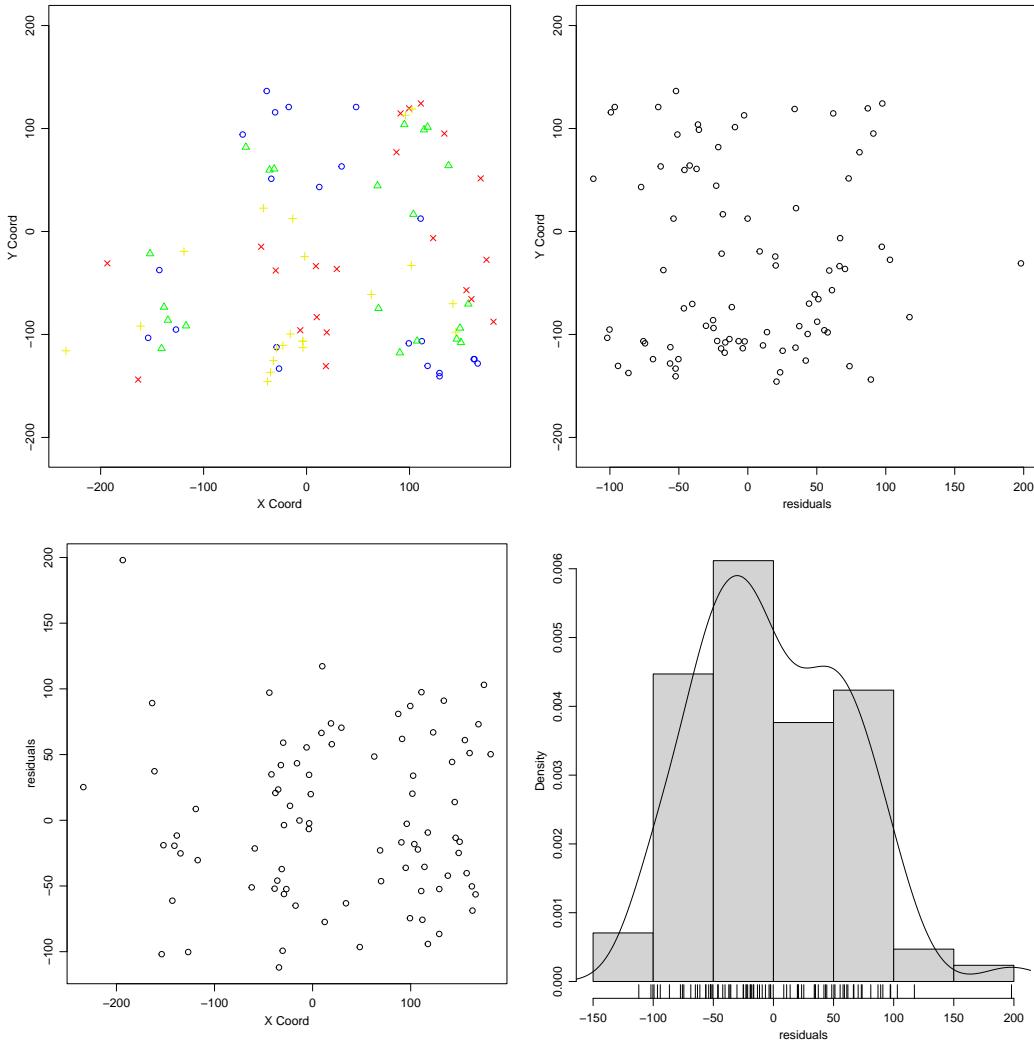
Los gráficos de dispersión de los datos frente a las coordenadas nos pueden ayudar a determinar si hay una tendencia. También, en lugar del histograma, nos puede interesar un gráfico de dispersión 3D

```
plot(wolfcamp, lowess = TRUE, scatter3d = TRUE)
```



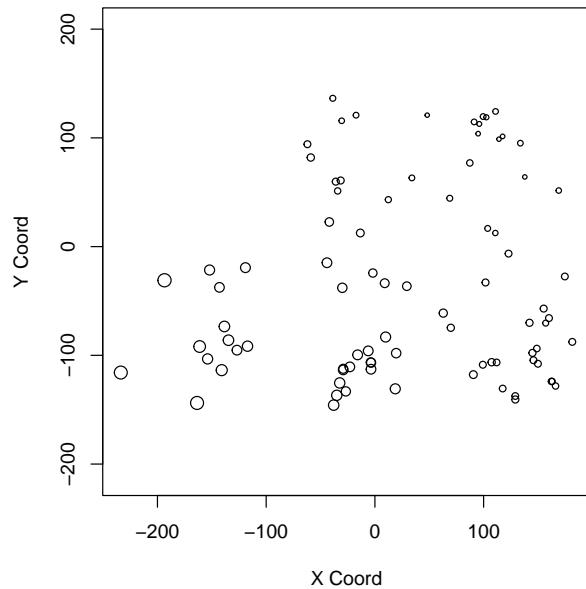
Si se asume que hay una tendencia puede interesar eliminarla:

```
plot(wolfcamp, trend=~coords)
```



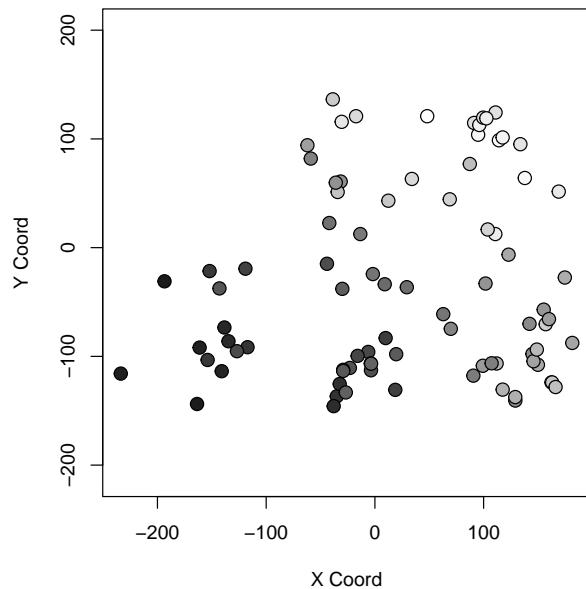
El comando `points(geodata)` (función `points.geodata`) genera un gráfico con las posiciones de los datos (y por defecto con el tamaño de los puntos proporcional al valor):

```
points(wolfcamp)
```



Se pueden establecer los tamaños de los puntos, simbolos y colores a partir de los valores de los datos. Por ejemplo, para los puntos, empleando el argumento: `pt.divide = c("data.proportional", "rank.proportional", "quintiles", "quartiles", "deciles", "equal")`.

```
points(wolfcamp, col = "gray", pt.divide = "equal")
```



B.3 Modelado de la dependencia

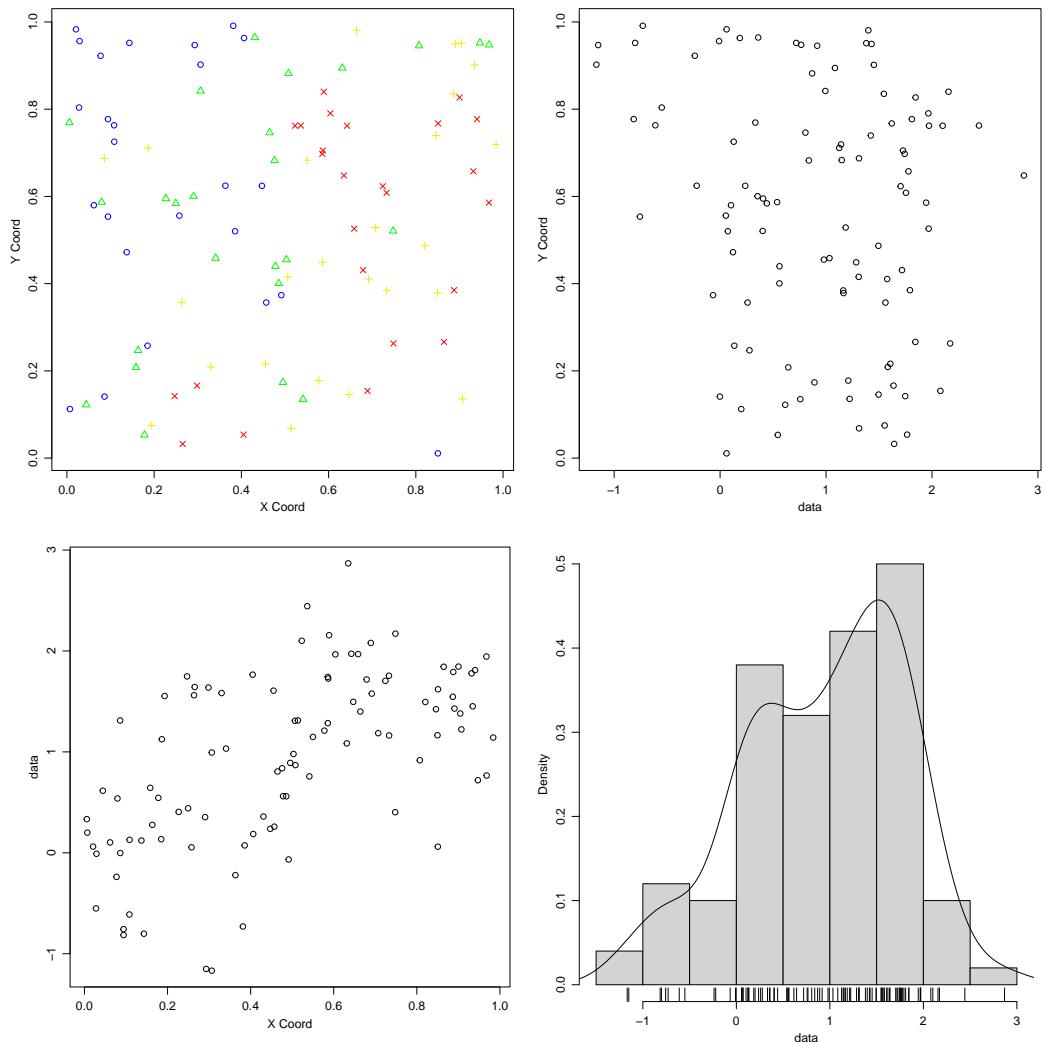
En la primera parte de esta sección consideraremos un proceso espacial sin tendencia:

```

data(s100) # Cargar datos estacionarios
summary(s100)

## Number of data points: 100
##
## Coordinates summary
##      Coord.X     Coord.Y
## min 0.005638006 0.01091027
## max 0.983920544 0.99124979
##
## Distance summary
##      min         max
## 0.007640962 1.278175109
##
## Data summary
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.1676955  0.2729882  1.1045936  0.9307179  1.6101707  2.8678969
##
## Other elements in the geodata object
## [1] "cov.model" "nugget"    "cov.pars"   "kappa"     "lambda"
plot(s100)

```



En el último apartado se tratará el caso general.

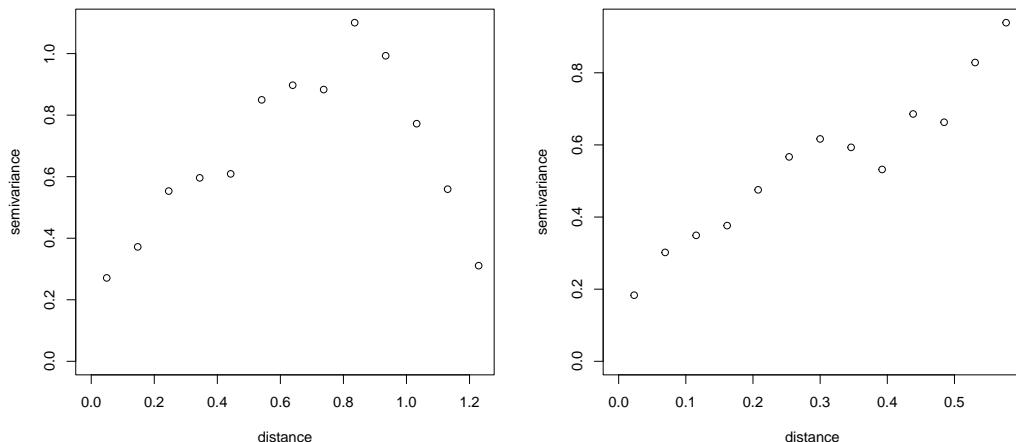
B.3.1 Variogramas empíricos

Los variogramas empíricos se calculan utilizando la función `variog`:

```
oldpar <- par(mfrow=c(1,2))
plot(variog(s100))
```

```
## variog: computing omnidirectional variogram
plot(variog(s100, max.dist = 0.6))
```

```
## variog: computing omnidirectional variogram
```



```
par(oldpar)
```

La recomendación es considerar solo saltos hasta la mitad de la máxima distancia (ver ‘Distance summary’ en resultados del sumario).

```
vario <- variog(s100, max.dist = 0.6)
```

```
## variog: computing omnidirectional variogram
```

```
names(vario)
```

```
## [1] "u"                  "v"                  "n"                  "sd"
## [5] "bins.lim"           "ind.bin"            "var.mark"           "beta.ols"
## [9] "output.type"        "max.dist"           "estimator.type"    "n.data"
## [13] "lambda"              "trend"               "pairs.min"          "nugget.tolerance"
## [17] "direction"           "tolerance"           "uvec"                "call"
# str(vario)
```

NOTA: La componente `u` contiene los saltos, `v` las estimaciones del semivariograma (semivarianzas) y `n` el número de aportaciones.

Los resultados pueden ser nubes de puntos (semivarianzas), valores discretizados (binned) o suavizados, dependiendo del parámetro: `option = c("bin", "cloud", "smooth")`

```
# Calculo de los variogramas empíricos
vario.b <- variog(s100, max.dist = 0.6) #discretizado
```

```
## variog: computing omnidirectional variogram
```

```

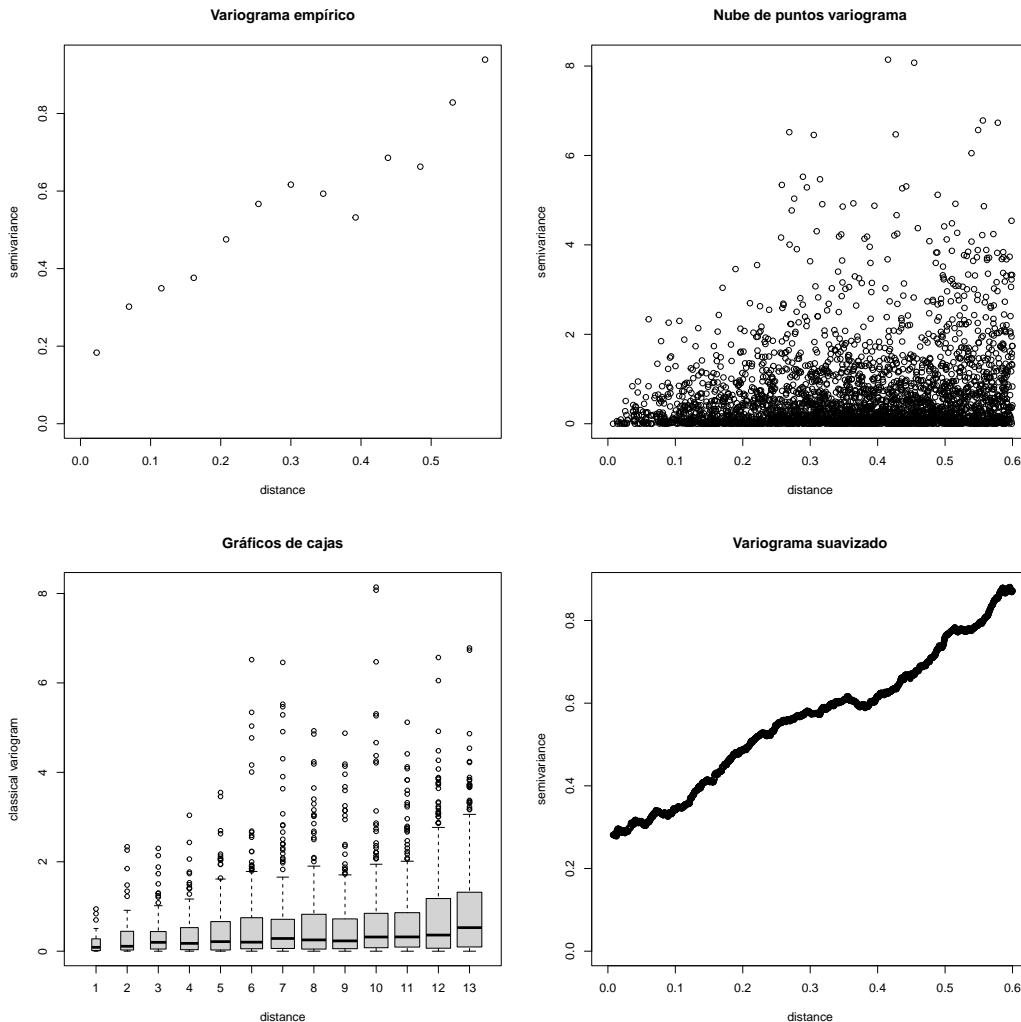
vario.c <- variog(s100, max.dist=0.6, op="cloud") #nube

## variog: computing omnidirectional variogram
vario.bc <- variog(s100, max.dist=0.6, bin.cloud=TRUE) #discretizado+nube

## variog: computing omnidirectional variogram
vario.s <- variog(s100, max.dist=0.6, op="sm", band=0.2) #suavizado

## variog: computing omnidirectional variogram
# Representación gráfica
oldpar<-par(mfrow=c(2,2)) # Preparar para 4 gráficos por ventana
plot(vario.b, main="Variograma empírico")
plot(vario.c, main="Nube de puntos variograma")
plot(vario.bc, bin.cloud=TRUE, main="Graficos de cajas")
title("Gráficos de cajas") # Corregir fallo del comando anterior
plot(vario.s, main="Variograma suavizado")

```



```
par(oldpar) # Restaurar opciones de gráficos
```

Si hay valores atípicos (o la distribución de los datos es asimétrica) puede ser preferible utilizar el estimador robusto. Se puede calcular este estimador estableciendo `estimator.type = "modulus"`:

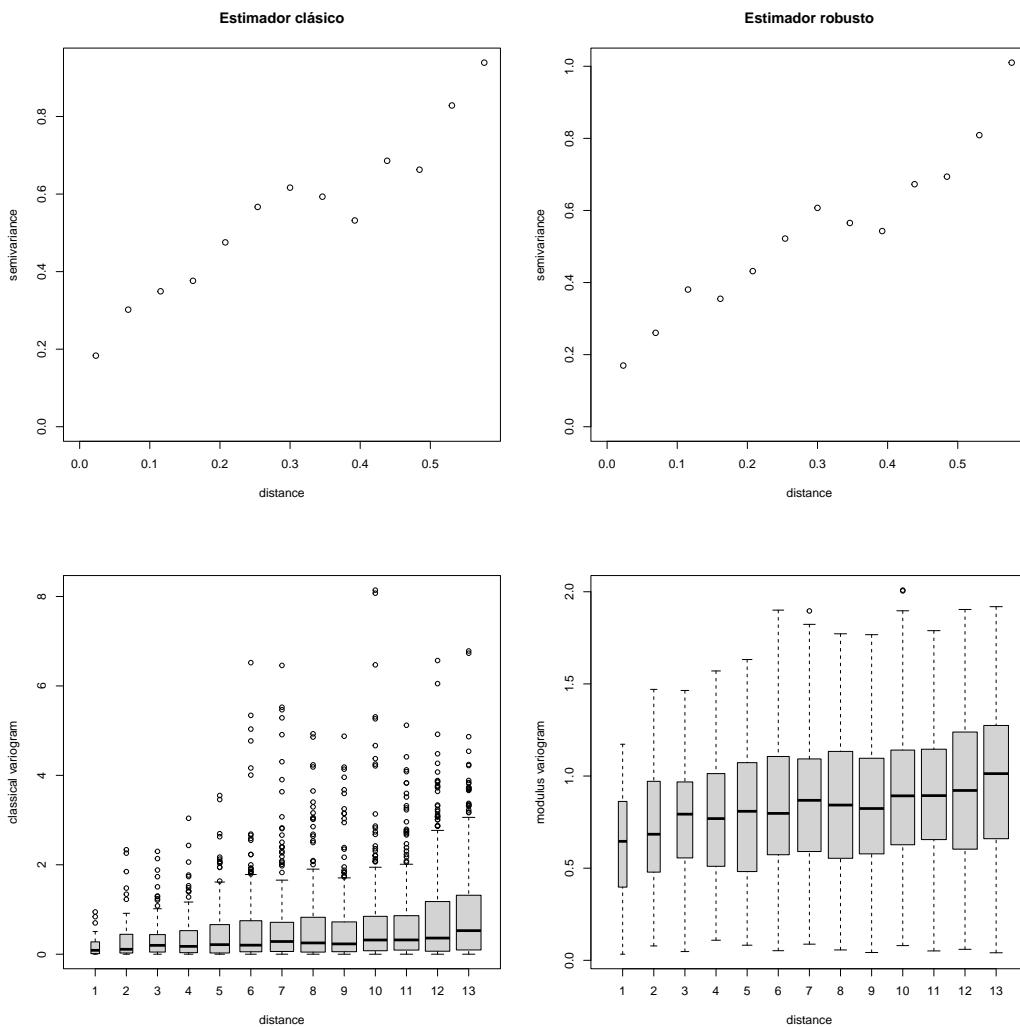
```

varior.b <- variog(s100, estimator.type = "modulus", max.dist=0.6)

## variog: computing omnidirectional variogram
varior.bc <- variog(s100, estimator.type = "modulus", max.dist=0.6, bin.cloud=TRUE)

## variog: computing omnidirectional variogram
oldpar<-par(mfrow=c(2,2)) #Preparar para 4 gráficos por ventana
plot(vario.b, main="Estimador clásico")
plot(varior.b, main="Estimador robusto")
plot(vario.bc, bin.cloud=TRUE)
plot(varior.bc, bin.cloud=TRUE)

```



```
par(oldpar) #Restaurar opciones de gráficos
```

En el caso de anisotropía, también se pueden obtener variogramas direccionales con la función `variog` mediante los argumentos `direction` y `tolerance`. Por ejemplo, para calcular un variograma en la dirección de 60 grados (con la tolerancia angular por defecto de 22.5 grados):

```

vario.60 <- variog(s100, max.dist = 0.6, direction = pi/3) #variograma en la dirección de 60 grados

## variog: computing variogram for direction = 60 degrees (1.047 radians)
##           tolerance angle = 22.5 degrees (0.393 radians)

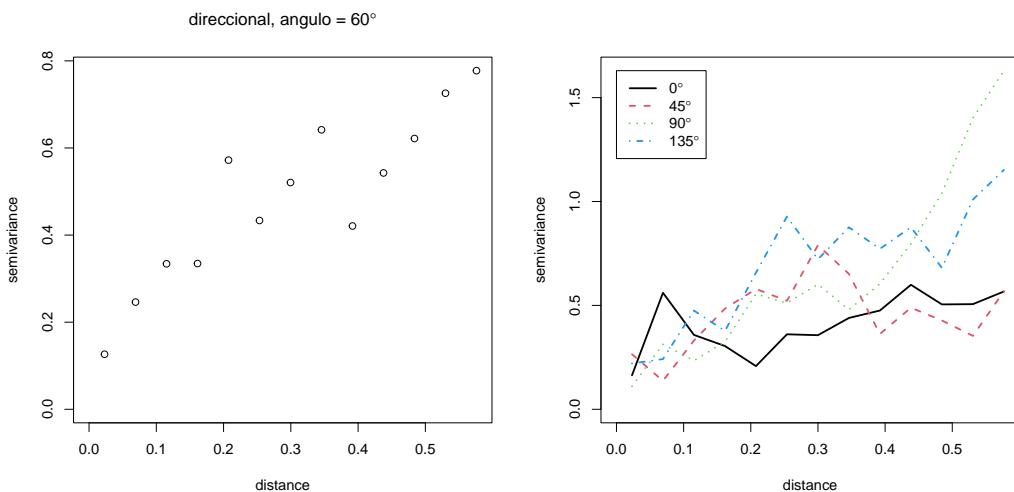
```

Para estudiar si hay anisotropía, se pueden calcular de forma rápida variogramas direccionales con la función `variog4`. Por defecto calcula cuatro variogramas direccionales, correspondientes a los ángulos 0, 45, 90 y 135 grados:

```
vario.4 <- variog4(s100, max.dist = 0.6)

## variog: computing variogram for direction = 0 degrees (0 radians)
## tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 45 degrees (0.785 radians)
## tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 90 degrees (1.571 radians)
## tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing variogram for direction = 135 degrees (2.356 radians)
## tolerance angle = 22.5 degrees (0.393 radians)
## variog: computing omnidirectional variogram

oldpar <- par(mfrow=c(1,2))
plot(vario.60)
title(main = expression(paste("direccional, angulo = ", 60 * degree)))
plot(vario.4, lwd = 2)
```



```
par(oldpar)
```

B.3.2 Ajuste de un modelo de variograma

Los estimadores empíricos no pueden ser empleados en la práctica (no verifican necesariamente las propiedades de un variograma válido), por lo que se suele recurrir en la práctica al ajuste de un modelo válido. Con el paquete `geoR` podemos realizar el ajuste:

1. “A ojo”: representando diferentes modelos sobre un variograma empírico (usando la función `lines.variomodel` o la función `eyefit`).
2. Por mínimos cuadrados: ajustando por mínimos cuadrados ordinarios (OSL) o ponderados (WLS) al variograma empírico (usando la función `variofit`),
3. Por máxima verosimilitud: estimando por máxima verosimilitud (ML) o máxima verosimilitud restringida (REML) los parámetros a partir de los datos (utilizando la función `likfit`),
4. Métodos bayesianos (utilizando la función `krige.bayes`).

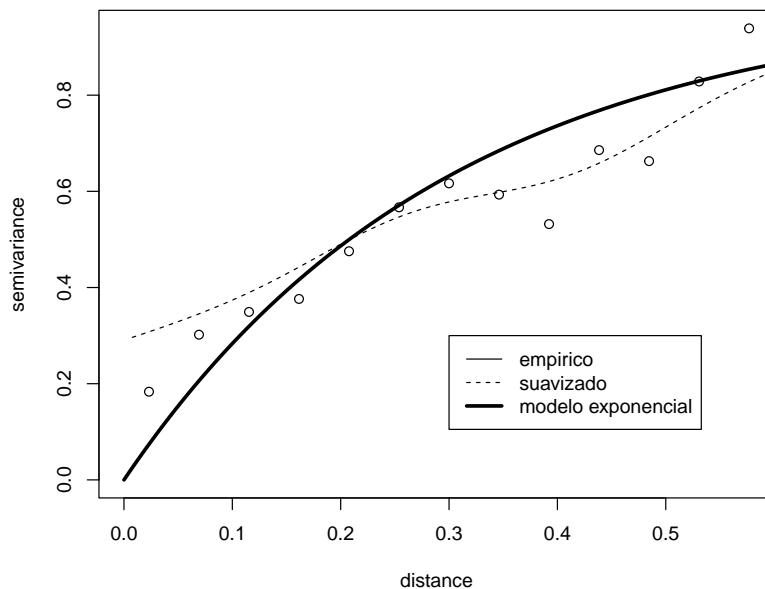
Ejemplo de ajuste “a ojo”:

```
vario.b <- variog(s100, max.dist=0.6) #discretizado
```

```
## variog: computing omnidirectional variogram
vario.s <- variog(s100, max.dist=0.6, option = "smooth", kernel = "normal", band = 0.2) #suavizado

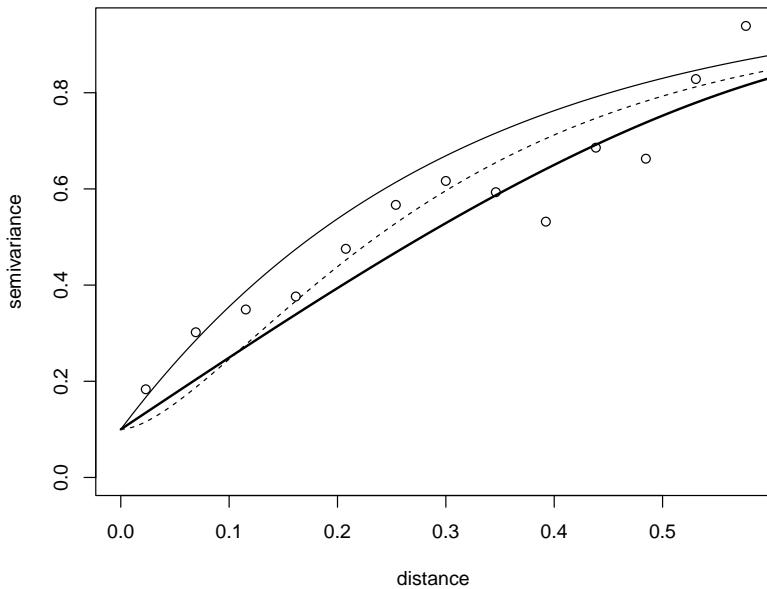
## variog: computing omnidirectional variogram
plot(vario.b)
lines(vario.s, type = "l", lty = 2)

lines.variomodel(cov.model = "exp", cov.pars = c(1,0.3), nugget = 0, max.dist = 0.6, lwd = 3)
legend(0.3, 0.3, c("empirico", "suavizado", "modelo exponencial"), lty = c(1, 2, 1), lwd = c(1, 1, 3))
```



Otros ajustes:

```
plot(vario.b)
lines.variomodel(cov.model = "exp", cov.pars = c(0.9,0.3), nug = 0.1, max.dist = 0.6)
lines.variomodel(cov.model = "mat", cov.pars = c(0.85,0.2), nug = 0.1, kappa = 1, max.dist = 0.6, lty = 2)
lines.variomodel(cov.model = "sph", cov.pars = c(0.8,0.8), nug = 0.1, max.dist = 0.6, lwd = 2)
```



Nota: no hace falta escribir el nombre completo de los parámetros (basta con que no dé lugar a confusión).

En las versiones recientes de geoR está disponible una función para realizar el ajuste gráficamente de forma interactiva (cuadro de diálogo en tcl/tk):

```
eyefit(vario.b)
```

Cuando se utilizan las funciones `variofit` y `likfit` para la estimación de parámetros, el efecto pepita (nugget) puede ser estimado o establecido a un valor fijo. Lo mismo ocurre con los parámetros de suavidad, anisotropía y transformación de los datos. También se dispone de opciones para incluir una tendencia. Las tendencias pueden ser polinomios en función de las coordenadas y/o funciones lineales de otras covariables.

Ejemplos de estimación por mínimos cuadrados (llamadas a `variofit`):

```
#  Modelo exponencial con par ini umbral 1 y escala 0.5 (1/3 rango =1.5)

vario.ols <- variofit(vario.b, ini = c(1, 0.5), weights = "equal") #ordinarios

## variofit: covariance model used is matern
## variofit: weights used: equal
## variofit: minimisation function used: optim
vario.wls <- variofit(vario.b, ini = c(1, 0.5), weights = "cressie") #ponderados

## variofit: covariance model used is matern
## variofit: weights used: cressie
## variofit: minimisation function used: optim
vario.wls

## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: matern with fixed kappa = 0.5 (exponential)
## parameter estimates:
##   tausq sigmasq     phi
## 0.1955 2.0110 1.4811
## Practical Range with cor=0.05 for asymptotic range: 4.437092
```

```

## 
## variofit: minimised weighted sum of squares = 31.5115
summary(vario.wls)

## $pmethod
## [1] "WLS (weighted least squares)"
##
## $cov.model
## [1] "matern"
##
## $spatial.component
##   sigmasq      phi
## 2.010972 1.481138
##
## $spatial.component.extra
##   kappa
## 0.5
##
## $nugget.component
##   tausq
## 0.1955322
##
## $fix.nugget
## [1] FALSE
##
## $fix.kappa
## [1] TRUE
##
## $practicalRange
## [1] 4.437092
##
## $sum.of.squares
##   value
## 31.5115
##
## $estimated.pars
##   tausq   sigmasq      phi
## 0.1955322 2.0109718 1.4811376
##
## $weights
## [1] "cressie"
##
## $call
## variofit(vario = vario.b, ini.cov.pars = c(1, 0.5), weights = "cressie")
##
## attr(),"class"
## [1] "summary.variomodel"

```

Ejemplo de estimación por máxima verosimilitud (llamada a likfit):

```

vario.ml <- likfit(s100, ini = c(1, 0.5)) #Modelo exponencial con par ini umbral y escala (1/3 rango)

## -----
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##         arguments for the maximisation function.
##         For further details see documentation for optim.

```

```

## likfit: It is highly advisable to run this function several
##          times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## -----
## likfit: end of numerical maximisation.

vario.ml

## likfit: estimated model parameters:
##      beta    tausq   sigmasq     phi
## "0.7766" "0.0000" "0.7517" "0.1827"
## Practical Range with cor=0.05 for asymptotic range: 0.547383
##
## likfit: maximised log-likelihood = -83.57
summary(vario.ml)

## Summary of the parameter estimation
## -----
## Estimation method: maximum likelihood
##
## Parameters of the mean component (trend):
##      beta
## 0.7766
##
## Parameters of the spatial component:
##      correlation function: exponential
##      (estimated) variance parameter sigmasq (partial sill) =  0.7517
##      (estimated) cor. fct. parameter phi (range parameter) =  0.1827
##      anisotropy parameters:
##          (fixed) anisotropy angle = 0  ( 0 degrees )
##          (fixed) anisotropy ratio = 1
##
## Parameter of the error component:
##      (estimated) nugget =  0
##
## Transformation parameter:
##      (fixed) Box-Cox parameter = 1 (no transformation)
##
## Practical Range with cor=0.05 for asymptotic range: 0.547383
##
## Maximised Likelihood:
##      log.L n.params      AIC      BIC
## "-83.57"        "4"    "175.1"  "185.6"
##
## non spatial model:
##      log.L n.params      AIC      BIC
## "-125.8"        "2"    "255.6"  "260.8"
##
## Call:
## likfit(geodata = s100, ini.cov.pars = c(1, 0.5))

```

Ejemplo de estimación por máxima verosimilitud restringida (opción de `likfit`):

```

vario.reml <- likfit(s100, ini = c(1, 0.5), lik.method = "RML")

## -----
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional

```

```

##           arguments for the maximisation function.
##           For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##           times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## -----
## likfit: end of numerical maximisation.

summary(vario.reml)

## Summary of the parameter estimation
## -----
## Estimation method: restricted maximum likelihood
##
## Parameters of the mean component (trend):
##   beta
## 0.7478
##
## Parameters of the spatial component:
##   correlation function: exponential
##   (estimated) variance parameter sigmasq (partial sill) = 0.8473
##   (estimated) cor. fct. parameter phi (range parameter) = 0.2102
##   anisotropy parameters:
##     (fixed) anisotropy angle = 0 (0 degrees)
##     (fixed) anisotropy ratio = 1
##
## Parameter of the error component:
##   (estimated) nugget = 0
##
## Transformation parameter:
##   (fixed) Box-Cox parameter = 1 (no transformation)
##
## Practical Range with cor=0.05 for asymptotic range: 0.6296295
##
## Maximised Likelihood:
##   log.L n.params      AIC      BIC
## "-81.53"      "4"  "171.1"  "181.5"
##
## non spatial model:
##   log.L n.params      AIC      BIC
## "-125.1"      "2"  "254.1"  "259.3"
##
## Call:
## likfit(geodata = s100, ini.cov.pars = c(1, 0.5), lik.method = "RML")

```

NOTAS:

- Para fijar el nugget a un valor p.e. 0.15 añadir las opciones: `fix.nugget = TRUE`, `nugget = 0.15`.
- Se puede tener en cuenta anisotropía geométrica en los modelos de variograma a partir de los parámetros `psiA` (ángulo, en radianes, de la dirección de mayor dependencia espacial i.e. con el máximo rango) y `psiR` (relación, mayor o igual que 1, entre los rangos máximo y mínimo). Se pueden fijar a distintos valores o estimarlos incluyendo las opciones `fix.psiA = FALSE` y `fix.psiR = FALSE` en las llamadas a las rutinas de ajuste.)

Representación gráfica junto al estimador empírico:

```

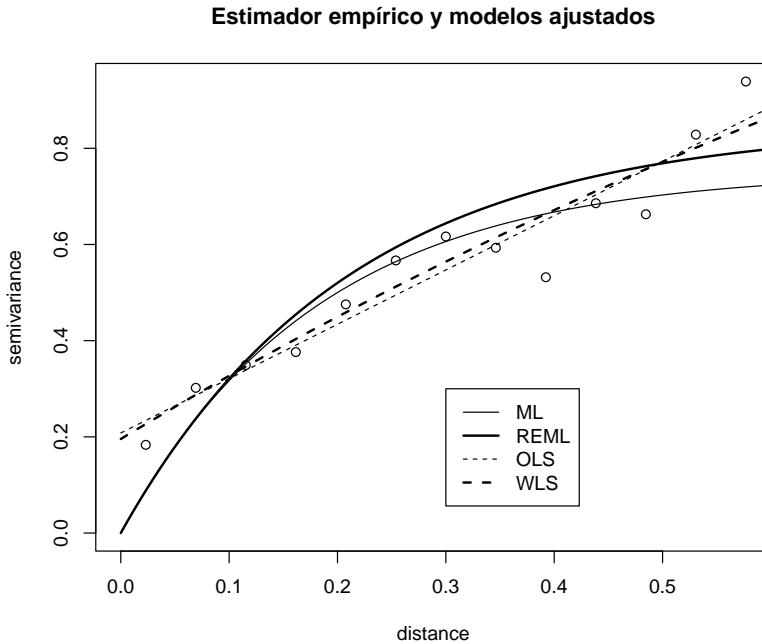
plot(vario.b, main = "Estimador empírico y modelos ajustados")
lines(vario.ml, max.dist = 0.6)

```

```

lines(vario.reml, lwd = 2, max.dist = 0.6)
lines(vario.ols, lty = 2, max.dist = 0.6)
lines(vario.wls, lty = 2, lwd = 2, max.dist = 0.6)
legend(0.3, 0.3, legend = c("ML", "REML", "OLS", "WLS"), lty = c(1, 1, 2, 2), lwd = c(1, 2, 1, 2))

```



B.3.3 Inferencia sobre el variograma

Se pueden obtener dos tipos de envolventes (envelopes, i.e. valores máximos y mínimos aproximados) del variograma empírico mediante simulación:

- Bajo la hipótesis de que no hay correlación espacial (obtenidos por permutaciones aleatorias de los datos sobre las posiciones espaciales), para estudiar si hay una dependencia espacial “significativa”.
- Bajo un modelo de variograma, para ilustrar la variabilidad del variograma empírico.

```

env.indep <- variog.mc.env(s100, obj.var = vario.b)

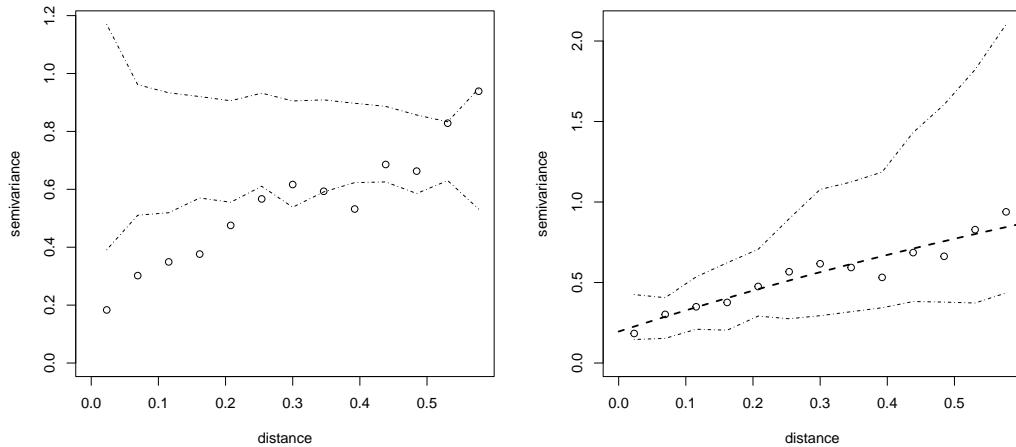
## variog.env: generating 99 simulations by permutating data values
## variog.env: computing the empirical variogram for the 99 simulations
## variog.env: computing the envelops

env.model <- variog.model.env(s100, obj.var = vario.b, model = vario.wls)

## variog.env: generating 99 simulations (with 100 points each) using the function grf
## variog.env: adding the mean or trend
## variog.env: computing the empirical variogram for the 99 simulations
## variog.env: computing the envelops

oldpar <- par(mfrow = c(1, 2))
plot(vario.b, envelope = env.indep)
plot(vario.b, envelope = env.model)
lines(vario.wls, lty = 2, lwd = 2, max.dist = 0.6)

```

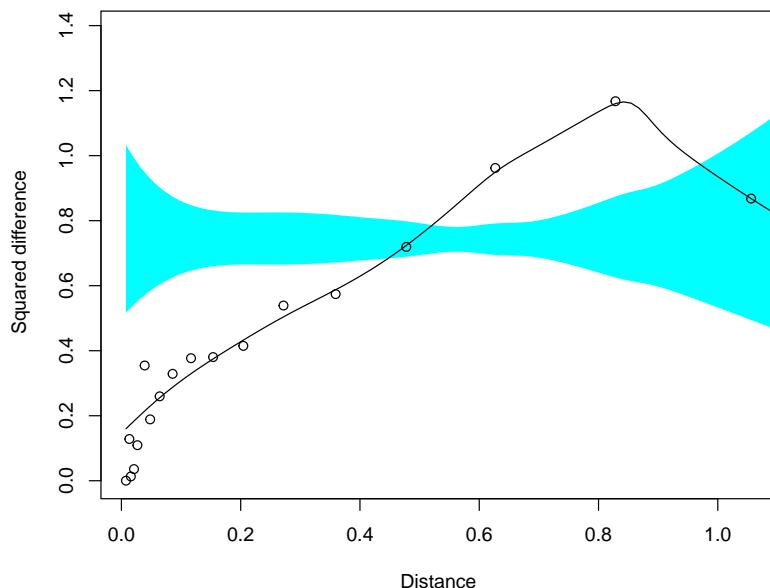


```
par(oldpar)
```

Para estudiar si hay una dependencia espacial “significativa” se puede emplear también la rutina `sm.variogram` del paquete `sm`. Estableciendo `model = "independent"` devuelve un p-valor para contrastar la hipótesis nula de independencia (i.e. se acepta que hay una dependencia espacial si $p \leq \alpha = 0.05$) y un gráfico en el que se muestra el estimador empírico robusto, un estimador suavizado y una región de confianza para el variograma suponiendo que el proceso es independiente (i.e. consideraríamos que hay dependencia espacial si el variograma suavizado no está contenido en esa región).

```
library(sm)
```

```
## Package 'sm', version 2.2-5.6: type help(sm) for summary information
sm.variogram(s100$coords, s100$data, model = "independent")
## Test of spatial independence: p = 0.024
```



Nota: Se puede realizar contrastes adicionales estableciendo el parámetro `model` a "isotropic" o

"stationary".

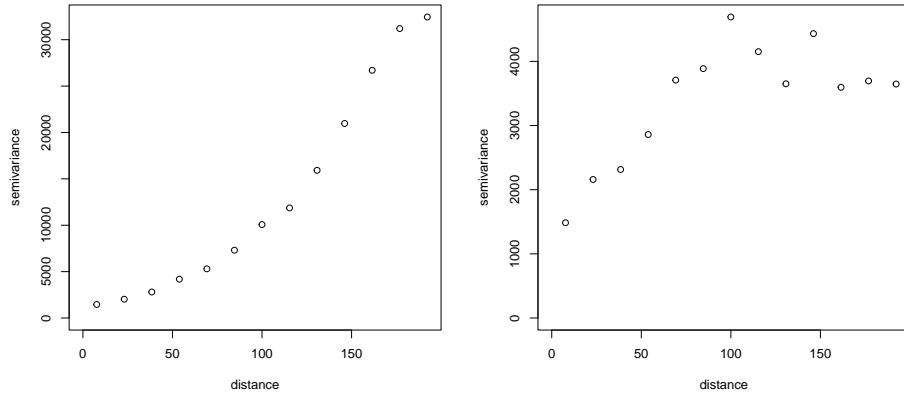
B.3.4 Estimación del variograma en procesos no estacionarios

Cuando el proceso no es estacionario (no se puede emplear directamente los estimadores empíricos) hay que eliminar la tendencia para estimar el variograma:

```
oldpar <- par(mfrow=c(1,2))
plot(variog(wolfcamp, max.dist = 200)) # Supone que el proceso es estacionario

## variog: computing omnidirectional variogram
plot(variog(wolfcamp, trend = ~coords, max.dist = 200)) # Asume una tendencia lineal en las coordenadas

## variog: computing omnidirectional variogram
```



```
par(oldpar)
```

B.4 Predicción espacial (kriging)

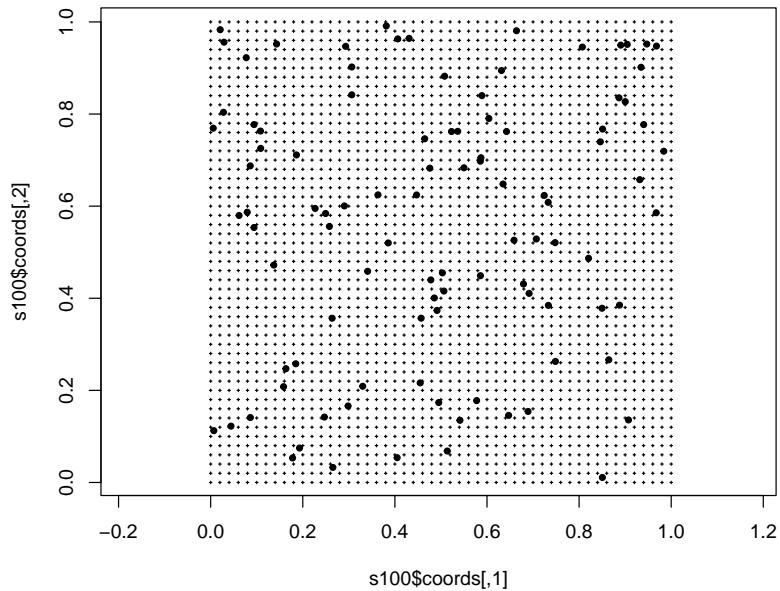
El paquete `geoR` dispone de opciones para los métodos kriging tradicionales, que dependiendo de las suposiciones acerca de la función de tendencia se clasifican en:

- *Kriging simple (KS)*: media conocida
- *Kriging ordinario (KO)*: se supone que la media es constante y desconocida.
- *Kriging universal (KU)*: también denominado kriging con modelo de tendencia, se supone que la media es una combinación lineal (desconocida) de las coordenadas o de otras variables explicativas.

Existen también opciones adicionales para kriging trans-normal (con transformaciones Box-Cox para aproximarse a la normalidad y transformación de nuevo de resultados a la escala original manteniendo insesgadez). También admite modelos de variograma geométricamente anisotrópicos.

Para obtener una rejilla discreta de predicción puede ser de utilidad la función `expand.grid`:

```
# Rejilla regular 51x51 en cuadrado unidad
xx <- seq(0, 1, 1 = 51)
yy <- seq(0, 1, 1 = 51)
pred.grid <- expand.grid(x = xx, y = yy)
plot(s100$coords, pch = 20, asp = 1)
points(pred.grid, pch = 3, cex = 0.2)
```



El comando para realizar kriging ordinario con variograma vario.wls sería:

```
ko.wls <- krige.conv(s100, loc = pred.grid, krige = krige.control(obj.m = vario.wls))
```

```
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

El resultado es una lista incluyendo predicciones (ko.wls\$predict) y varianzas kriging (ko.wls\$krige.var):

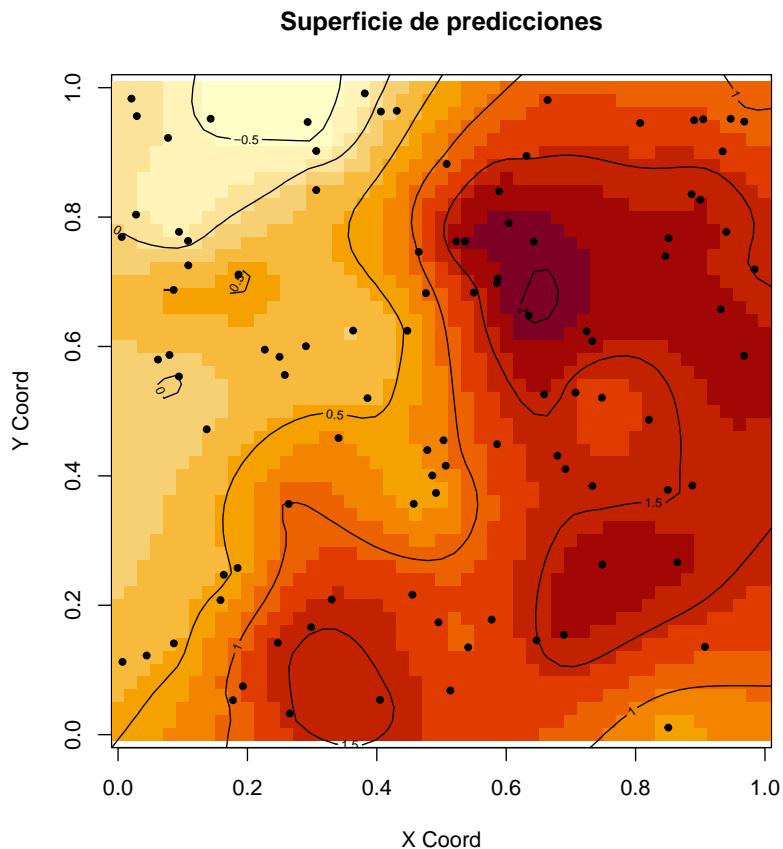
```
names(ko.wls)
```

```
## [1] "predict"      "krige.var"     "beta.est"      "distribution" "message"
## [6] "call"
```

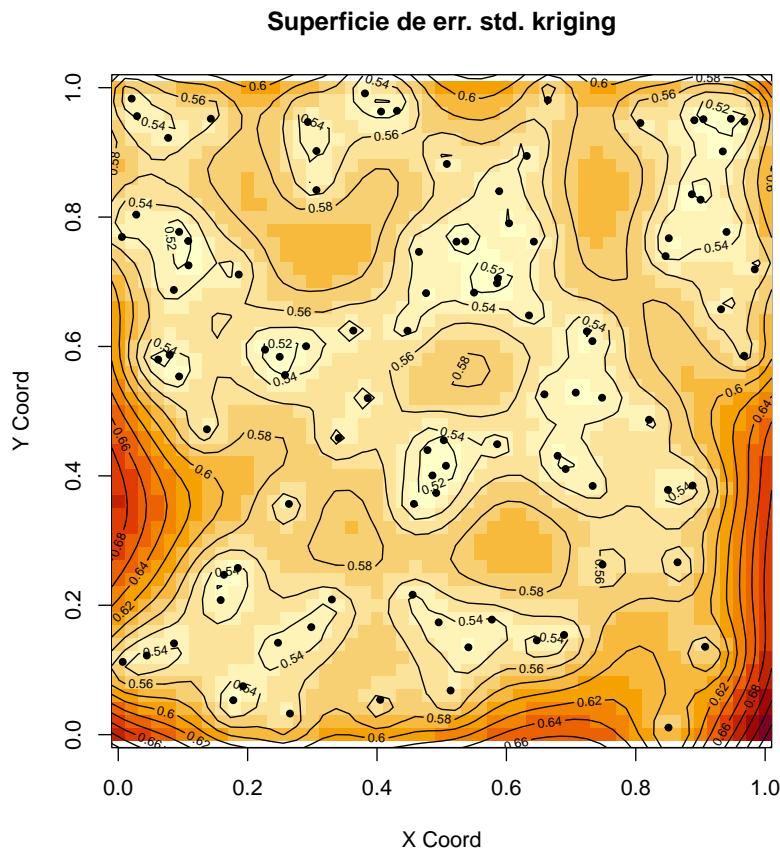
Para ver todas las opciones de kriging disponibles ejecutar ?krige.control. Para kriging con vecindario local (archivos de datos grandes) se puede utilizar la función ksline.

Para representar las superficies se podría utilizar la función `image()`, aunque la última versión del método `image.krige()` puede fallar al añadir elementos (por lo menos en RMarkdown; tampoco es compatible con `par(mfrow)`):

```
# oldpar <- par(mfrow = c(1, 2))
# image.krige no es compatible con mfrow en últimas versiones
image(ko.wls, coords.data=s100$coords, main = "Superficie de predicciones")
contour(ko.wls, add = TRUE) #añadir gráfico de contorno
```



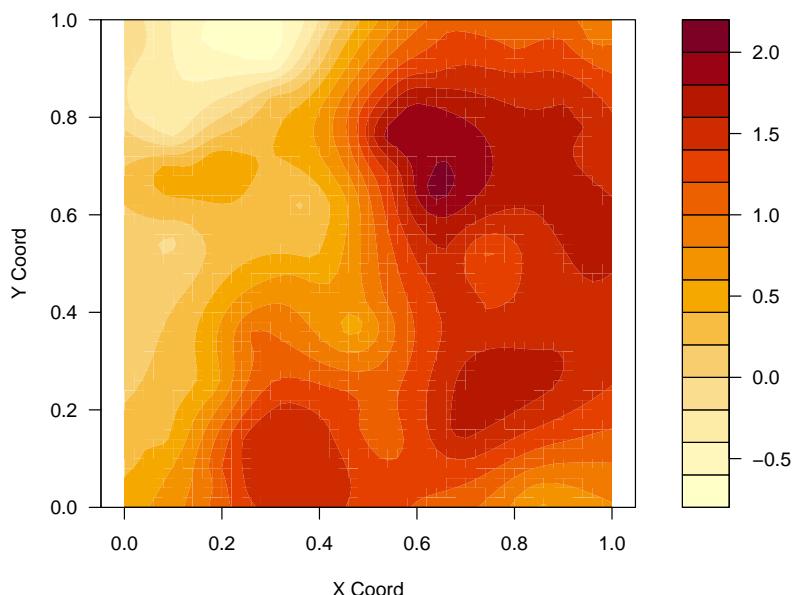
```
image(ko.wls, coords.data=s100$coords, values = sqrt(ko.wls$krige.var), main = "Superficie de errores")
contour(ko.wls, values = sqrt(ko.wls$krige.var), add = TRUE)
```



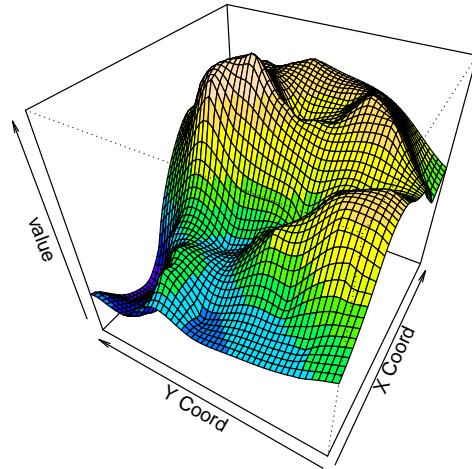
```
# par(oldpar)
```

Otras opciones:

```
contour(ko.wls, filled = TRUE)
```

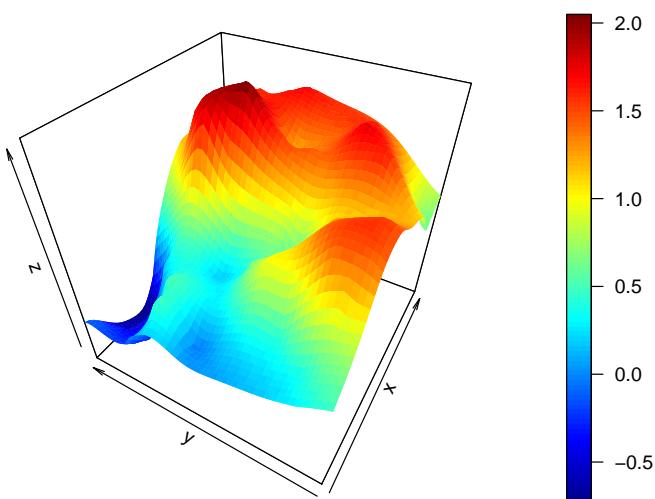


```
fcol <- topo.colors(10)[cut(matrix(ko.wls$pred, nrow=51, ncol=51)[-1,-1], 10, include.lowest=TRUE)]
persp(ko.wls, theta=-60, phi=40, col=fcol)
```



```
if(!require(plot3D))
  stop('Required package `plot3D` not installed.') # install.packages('plot3D')

## Loading required package: plot3D
persp3D(xx, yy, matrix(ko.wls$predict, nrow = length(xx)), theta=-60, phi=40)
```



```
if(!require(npss)) {
  cat("Required package `npss` not installed!\n")
```

```

cat("On windows, run `install.packages('https://github.com/rubenfcasal/npsp/releases/download/v0.7-8/spersp.R')`")
} else
  spersp(xx, yy, ko.wls$predict, theta=-60, phi=40)

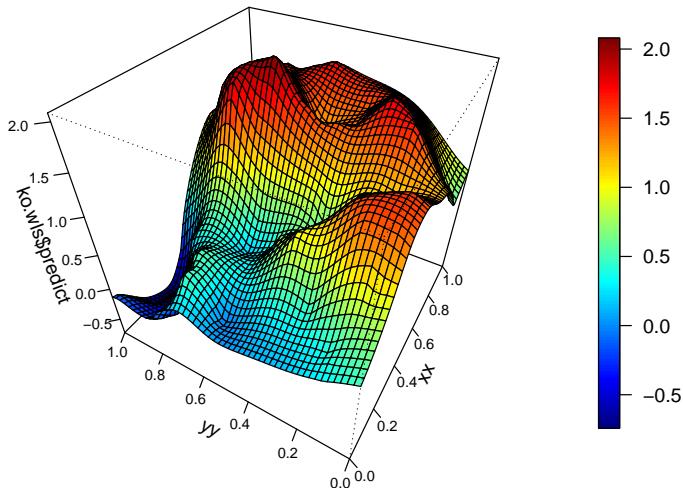
## Loading required package: npsp

## Package npsp: Nonparametric Spatial Statistics,
## version 0.7-8 (built on 2021-05-10).
## Copyright (C) R. Fernandez-Casal 2012-2021.
## Type `help(npsp)` for an overview of the package or
## visit https://rubenfcasal.github.io/npsp.

##
## Attaching package: 'npsp'

## The following object is masked from 'package:sm':
##
##     binning

```



B.4.1 Validación cruzada

Para verificar si un modelo (de tendencia y variograma) describe adecuadamente la variabilidad espacial de los datos (p.e. para comparar modelos), se emplea normalmente la técnica de validación cruzada, función `xvalid` en `geor`. Por defecto la validación se realiza sobre los datos eliminando cada observación (y utilizando las restantes para predecir), aunque se puede utilizar un conjunto diferente de posiciones (o de datos) mediante el argumento `location.xvalid` (y `data.xvalid`).

En el caso de procesos estacionarios permitiría diagnosticar si el modelo de variograma describe adecuadamente la dependencia espacial de los datos:

```
xv.wls <- xvalid(s100, model = vario.wls)
```

```

## xvalid: number of data locations      = 100
## xvalid: number of validation locations = 100
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
## xvalid: end of cross-validation

```

```
summary(xv.wls)

##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## errors     -1.429944 -0.4017821 0.04881742 0.0008450629 0.3359677 1.319640
## std.errors -2.110654 -0.7048560 0.07804159 0.0011568059 0.5922810 2.228054
##           sd
## errors     0.5299818
## std.errors 0.9190753

xv.reml <- xvalid(s100, model = vario.reml)

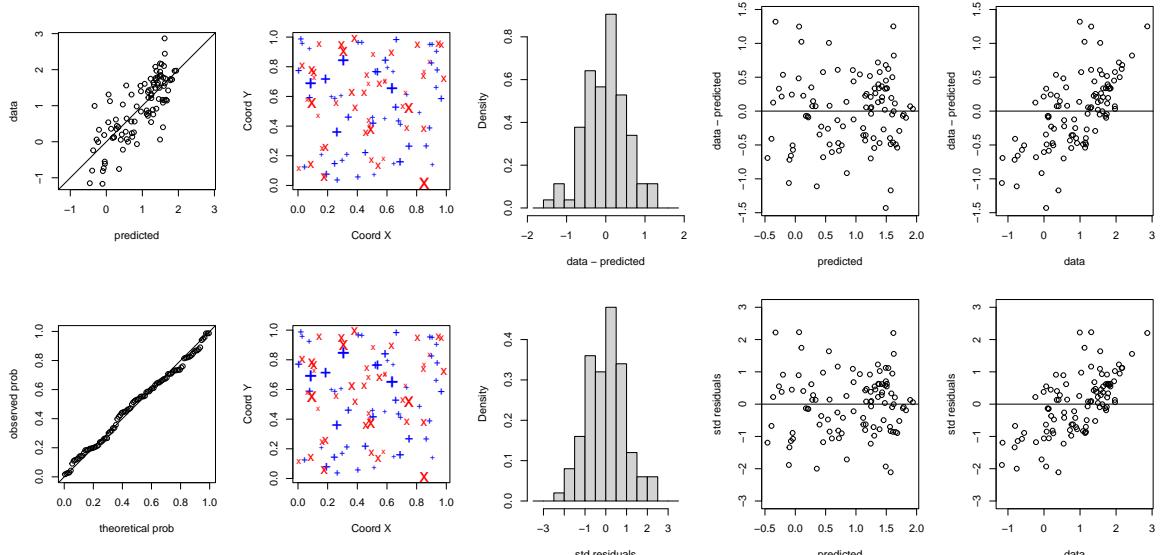
## xvalid: number of data locations      = 100
## xvalid: number of validation locations = 100
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
## xvalid: end of cross-validation

summary(xv.reml)

##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## errors     -1.178020 -0.3109277 0.02326020 0.011894019 0.2631596 1.521489
## std.errors -2.419106 -0.7304294 0.07954355 0.009241635 0.5802049 2.690047
##           sd
## errors     0.4813133
## std.errors 0.9906166
```

Por defecto la función `plot (plot.xvalid)` muestra 10 gráficos diferentes (para más información ejecutar `?plot.xvalid`), a grosso modo los cinco primeros se corresponden con residuos simples (valores observados menos predicciones) y los siguientes con residuos estandarizados (dividiendo por la raíz cuadrada de la varianza de predicción).

```
oldpar <- par(mfrow = c(2, 5), mar = c(bottom = 4.5, left = 4, top = 2, right = 2))
plot(xv.wls, ask = FALSE)
```



```
par(oldpar)

# plot(xv.reml)
```

NOTA: Para re-estimar los parámetros del modelo cada vez que se elimina una observación (i.e. validar el procedimiento de estimación) añadir la opción `reest = TRUE` (puede requerir mucho tiempo de computación).

Referencias

Bibliografía básica

- Bivand, R.S., Pebesma, E.J. y Gómez-Rubio, V. (2008). *Applied Spatial Data Analysis with R*. Springer.
- Diggle, P. y Ribeiro, P.J. (2007). *Model-based Geostatistics*. Springer.
- Schabenberger, O. y Gotway, C.A. (2005). *Statistical Methods for Spatial Data Analysis*. Chapman and Hall.

Bibliografía complementaria

- Chilès, J.P. y P. Delfiner (2012). *Geostatistics: modeling spatial uncertainty*. Wiley.
- Cressie, N. (1993). *Statistics for Spatial Data*. John Wiley.
- Wikle, C.K., Zammit-Mangion, A. y Cressie, N. (2019). *Spatio-temporal Statistics with R*. Chapman and Hall/CRC (accesible online).

Bibliografía completa

- Bivand, R. S., Pebesma, E., y Gómez-Rubio, V. (2013). *Applied Spatial Data Analysis with R* (Second). Springer. <http://www.asdar-book.org/>
- Pebesma, E. (2018). Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10(1), 439-446. <https://doi.org/10.32614/RJ-2018-009>
- Pebesma, E., y Bivand, R. (2021). *Spatial Data Science*. <https://keen-swartz-3146c4.netlify.app>
- Pebesma, E. J., y Bivand, R. S. (2005). Classes and methods for spatial data in R. *R News*, 5(2), 9-13. <https://CRAN.R-project.org/doc/Rnews/>