

Estadística Espacial con R

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

2022-11-22

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	13
1.4 Objetivos y procedimiento	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	29
2.4 Operaciones con datos espaciales	30
2.5 Análisis exploratorio de datos espaciales	35
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	52
3.4 Comentarios sobre los distintos métodos	62
4 Predicción Kriging	65
4.1 Introducción	65
4.2 Kriging con media conocida: kriging simple	66
4.3 Kriging con media desconocida: kriging universal y kriging residual	67
4.4 Kriging con el paquete gstat	70
4.5 Consideraciones acerca de los métodos kriging	73
4.6 Validación cruzada del modelo ajustado	76
4.7 Otros métodos kriging	82
5 Procesos espaciales multivariantes	85
6 Procesos espacio-temporales	87
A Introducción al paquete sp	89
A.1 Tipos de objetos	89
A.2 Métodos y procedimientos clases sp	103
A.3 Representaciones gráficas	104
B Introducción al paquete geoR	109
B.1 Inicio de una sesión y de carga de datos	109
B.2 Análisis descriptivo de datos geoestadísticos	110
B.3 Modelado de la dependencia	114
B.4 Predicción espacial (kriging)	127

Referencias	135
Bibliografía completa	135

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 (e.g. ver enlace), siendo 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.

“... the first law of geography: everything is related to everything else, but near things are more related than distant things”.

— Tobler, 1970.

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, economía, 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]

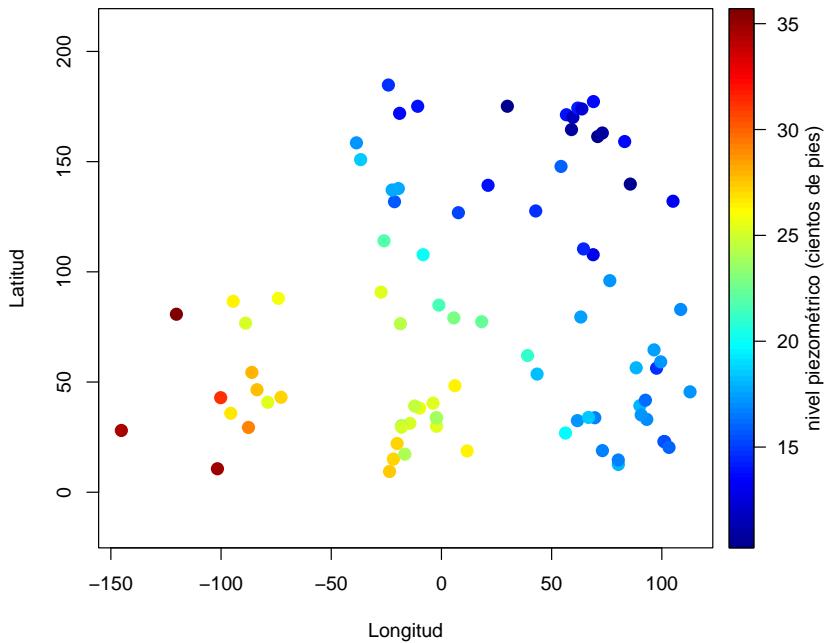


Figura 1.1: Nivel del agua subterránea en 85 localizaciones del acuífero Wolfcamp (obtenidas durante un estudio sobre el posible emplazamiento de un depósito de residuos nucleares).

- **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]

```
## Linking to GEOS 3.9.1, GDAL 3.2.1, PROJ 7.2.1
```

- **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 $\mathbf{s} = (s_1, \dots, s_{d-1}, t) \in \mathbb{R}^{d-1} \times \mathbb{R}^{+,0}$, donde $\mathbb{R}^{+,0} = \{t \in \mathbb{R} : t \geq 0\}$. Por

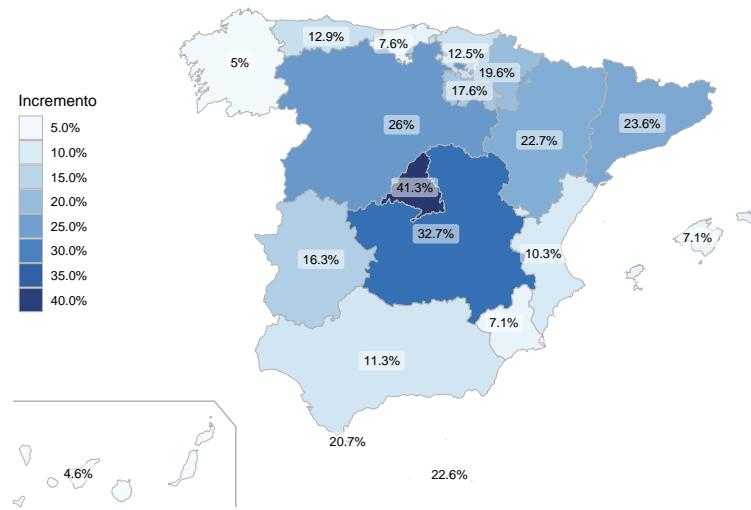


Figura 1.2: Porcentaje de incremento de las defunciones en el año 2020 respecto al 2019 por CCAA (datos [INE][\(<https://www.ine.es/jaxiT3/Tabla.htm?t=6546>\)](https://www.ine.es/jaxiT3/Tabla.htm?t=6546)).



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

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...

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`, `stars`, `rgdal` y `rgeos`, entre otros. En la Sección 2.1 se incluye información adicional sobre estos paquetes.

En este libro emplearemos principalmente el paquete `gstat` para el análisis de datos geoestadísticos (siguiente sección; 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).

1.1.2 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` (Apéndice A) y los objetos de datos de los paquetes `sf` y `stars` (Capítulo 2).

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.

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 1.3.

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:

$$Z(\mathbf{s}) = \sum_{j=0}^p X_j(\mathbf{s})\beta_j + \varepsilon(\mathbf{s}), \quad \mathbf{s} \in D, \quad (1.2)$$

(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 $\mathbf{X}_{ij} = X_{j-1}(\mathbf{s}_i)$ 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*):

$$\hat{\beta}_{ols} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Z}, \quad (1.3)$$

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:

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

¹D. 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).

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*):

$$\hat{\beta}_{gls} = (\mathbf{X}^\top \Sigma^{-1} \mathbf{X})^{-1} \mathbf{X}^\top \Sigma^{-1} \mathbf{Z}. \quad (1.4)$$

Si $\Sigma = \sigma^2 \mathbf{I}_n$, siendo \mathbf{I}_n la matriz identidad $n \times n$, los estimadores (1.3) y (1.4) 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 :

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

donde $\mathbf{x}_0 = (X_0(\mathbf{s}_0), \dots, X_p(\mathbf{s}_0))^\top$. Bajo las hipótesis del modelo clásico el predictor óptimo sería la estimación de la tendencia $\hat{\mu}(\mathbf{s}_0) = \mathbf{x}_0^\top \hat{\beta}_{ols}$ (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:

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

sin embargo no es el predictor lineal óptimo. Puede verse (e.g. Goldberger, 1962; Sección 4.3.2) que en este caso el mejor predictor lineal insesgado es:

$$\tilde{Z}(\mathbf{s}_0) = \mathbf{x}_0^\top \hat{\beta}_{gls} + \mathbf{c}^\top \Sigma^{-1} (\mathbf{Z} - \mathbf{X} \hat{\beta}_{gls}), \quad (1.5)$$

(el denominado predictor del *kriging universal*, Sección 4.3), 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 $Var(\hat{Z}(\mathbf{s}_0)) - Var(\tilde{Z}(\mathbf{s}_0)) \geq 0$ 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 (ver Sección 3.2.1):

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

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 (ver Sección 3.3), podemos obtener por ejemplo una aproximación del predictor óptimo de $Z(\mathbf{s}_0)$ sustituyendo en (1.5) $\Sigma(\theta)$ por $\Sigma(\hat{\theta})$ y $\mathbf{c}(\theta)$ por $\mathbf{c}(\hat{\theta})$.

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.

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:

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

(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 varía:

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

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*; Sección 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 $C(\mathbf{0}) = \text{Var}(Z(\mathbf{s})) > 0$. 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:

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

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:

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

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; ver Sección 3.3.2).

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.

1.3.1 Características del variograma

Además del umbral (si existe, ya que el variograma podría no estar acotado; ver sección anterior), 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.

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

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

⁴Además de poder interpretar su influencia en la predicción espacial (Sección 4.5.2), 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).

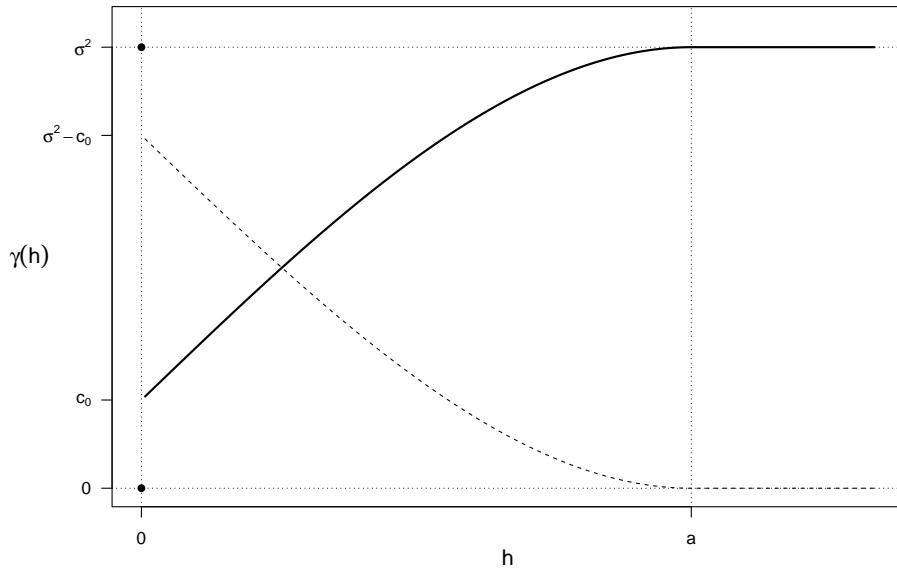


Figura 1.4: Relación entre el covariograma (línea discontinua) y el variograma (línea continua) en el caso unidimensional (o isotrópico), y principales características: nugget (c_0), umbral (σ^2 ; umbral parcial $\sigma^2 - c_0$) y rango (a).

$Z(\cdot)$ (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.

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 parcial.

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.

1.3.2 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:

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

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 e.g. 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 semidefinido negativo, es decir:

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

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).

1.3.3 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:

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

Aunque nos centraremos principalmente en el caso de soporte puntual, los métodos descritos en este libro pueden ser extendidos para el caso de distintos soportes (por ejemplo el *block kriging* descrito en la Sección 4.7.1). 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 procedimiento

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 (puede ser un 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 4.6) o reformulación del mismo.
 4. Empleo del modelo aceptado (Capítulo 4).

Como ya se comentó, emplearemos el paquete **gstat** en este proceso (Sección 1.1.2).

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, 2.3 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 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 1.3.3), aunque en algunos casos es el valor puntual correspondiente al centro de la celda (nodo de una rejilla vectorial).

En este libro 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 1.3.3, muchos métodos geoestadísticos admiten datos en distintos soportes (por ejemplo el *block kriging* descrito en la Sección 4.7.1), aunque combinar datos en diferentes soportes puede presentar en la práctica serias dificultades (para más detalles ver referencias al final de la Sección 1.3.3).

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 GEOS) 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:

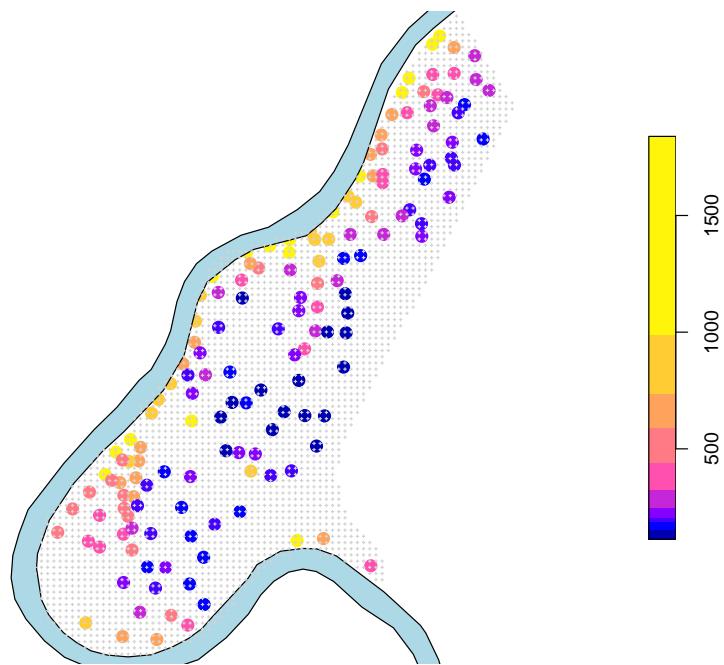


Figura 2.1: Concentración de zinc (ppm) en el entorno del río Meuse (datos ‘sp::meuse’).

- 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

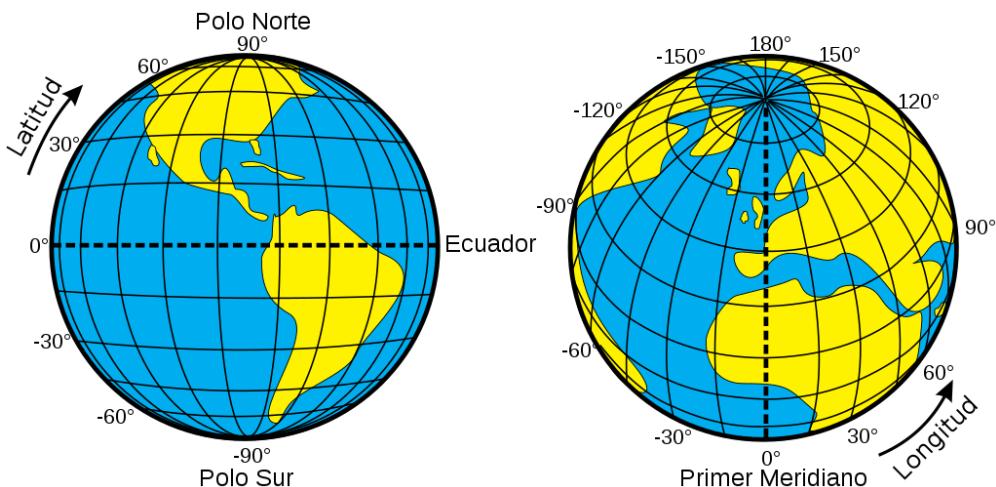


Figura 2.2: Coordenadas geográficas en la superficie terrestre (Fuente Wikimedia Commons).

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 (también puede ser una cadena de texto *PROJ.4*). El estándar más empleado es el EPSG (*European Petroleum Survey Group*), y es 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 de texto *PROJ.4* 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 INE), con el siguiente código:

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

```

library(dplyr)
library(tidyr)
library(sf)
library(mapSpain) # install.packages("mapSpain")
load("datos/mortalidad.RData")

mort_sf <- mortalidad %>%
  filter(!ccaa.code %in% c("00", "99"), periodo %in% c("2019", "2020"), sexo == "Total") %>%
  select(-sexo, -ccaa.name) %>%
  pivot_wider(names_from = periodo, values_from = value, names_prefix = "mort.") %>%
  mutate(incremento = 100*(mort.2020 - mort.2019)/mort.2019)

# CUIDADO: el primer elemento de xxx_join debe ser un objeto sf
# para que lo procese el paquete sf
mort_sf <- esp_get_ccaa() %>%
  left_join(mort_sf, by = c("codauto" = "ccaa.code"))
library(ggplot2)
ggplot(mort_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()

```

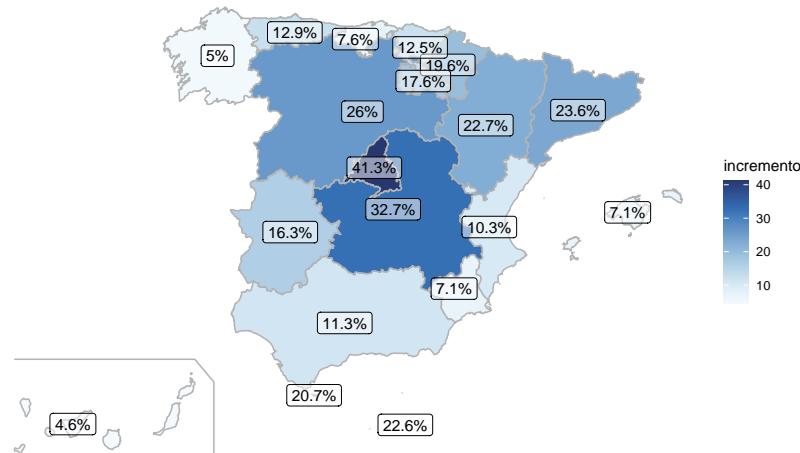


Figura 2.3: Ejemplo de gráfico generado empleando los paquetes ‘dplyr’ y ‘ggplot2’.

[Figura 2.3]

Sin embargo, en este libro 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`, `mapsf`, `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, nbreacks, 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).
- `nbreacks`: 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, 2.7).
-: 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.

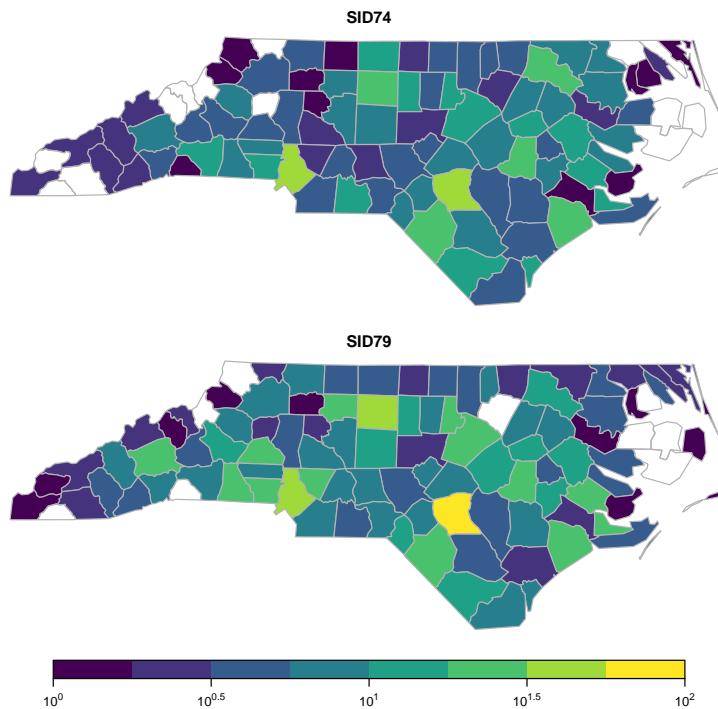


Figura 2.4: Ejemplo de gráfico con múltiples atributos (con colores personalizados y leyenda común, en escala logarítmica personalizada).

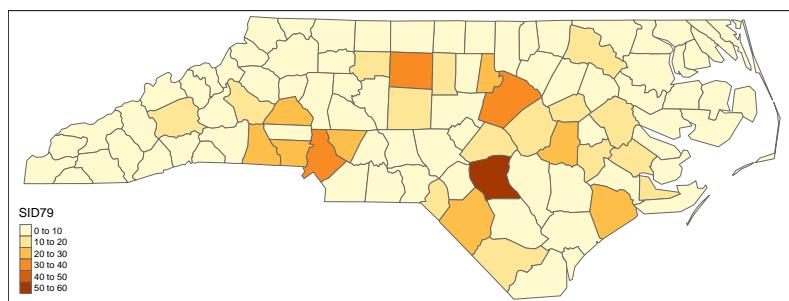


Figura 2.5: Ejemplo de mapa estático creado con ‘tmap’.

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.

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 (por ejemplo `.shp` para *ESRI Shapefile*) o a partir del prefijo (por ejemplo `PG`: para *PostGIS/PostgreSQL*):

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

## [1] "nc.dbf" "nc.prj" "nc.shp" "nc.shx"
# ESRI Shapefile, consta de 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 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:

- `rnatuarlearth`: 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 2.6]



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*.
- GSHHG A *Global Self-consistent, Hierarchical, High-resolution Geography Database*.

Muchos de los archivos de datos están en formato NetCDF (*Network Common Data Form*) y se pueden importar a R con el paquete `ncdf4`.

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)

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.

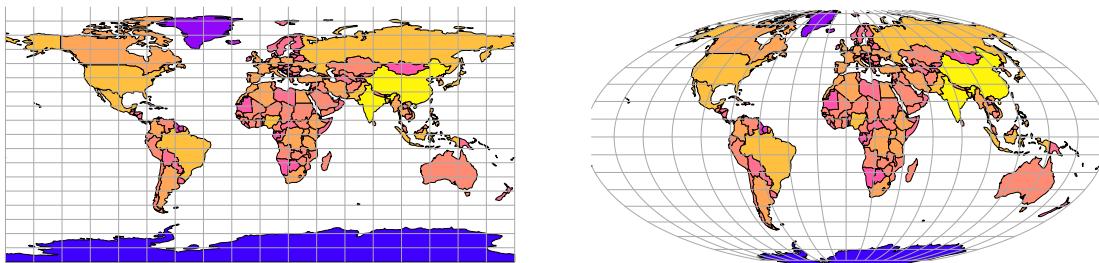


Figura 2.7: Mapa de la población estimada por países (en escala logarítmica), datos sin proyectar (izquierda) y con proyección de Mollweide (derecha).

- `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()`.

Ejemplo 2.1 (Creación de una rejilla de predicción). Continuando con los datos del Ejercicio 2.2, para crear un objeto con las posiciones de predicción, podríamos generar un buffer (`st_buffer()`) de radio 40 en torno a las posiciones de observación y a partir de él crear una rejilla vectorial (`st_make_grid(..., what = "centers")`) de dimensiones 50 por 50 e intersecarla con el buffer.

```
load("datos/aquifer.RData")
aquifer$head <- aquifer$head/100 # en cientos de pies
aquifer_sf <- st_as_sf(aquifer, coords = c("lon", "lat"), agr = "constant")
buffer <- aquifer_sf %>% st_geometry() %>% st_buffer(40)
grid <- buffer %>% st_make_grid(n = c(50, 50), what = "centers") %>%
  st_intersection(buffer)
plot(buffer)
plot(grid, pch = 3, cex = 0.5, add = TRUE)
```

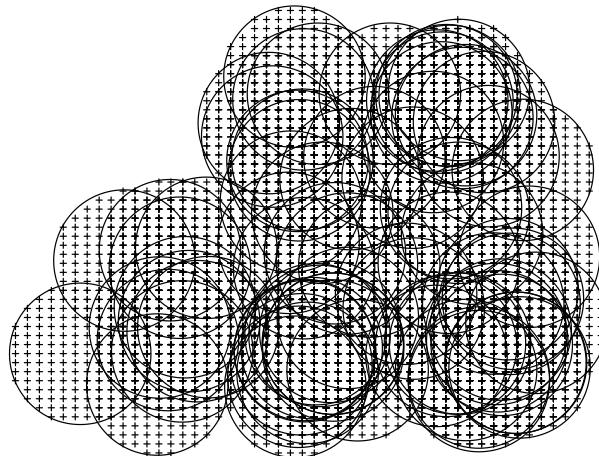


Figura 2.8: Rejilla en torno a las posiciones de los datos de ‘aquifer’.

Sin embargo, en lugar de emplear una rejilla `sf`, puede resultar preferible (por ejemplo para la representación gráfica) emplear una rejilla `stars`

```
library(stars)

## Loading required package: abind
grid <- buffer %>% st_as_stars(nx = 50, ny = 50) %>% st_crop(buffer)
idw <- gstat::idw(formula = head ~ 1, locations = aquifer_sf, newdata = grid)

## [inverse distance weighted interpolation]
plot(idw["var1.pred"], col = sf.colors(64), main = "")

# Error gstat::idw, cambia las coordenadas del objeto stars
# summary(st_coordinates(grid))
# summary(st_coordinates(idw))
# Posible solución: añadir el resultado a `grid` y emplearlo en lugar de `idw`
# grid$var1.pred <- idw$var1.pred
# plot(grid["var1.pred"], col = sf.colors(64), axes = TRUE, main = "")
```

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 3.3.3) y los predictores kriging serían los más eficientes (Sección 4.5). 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).

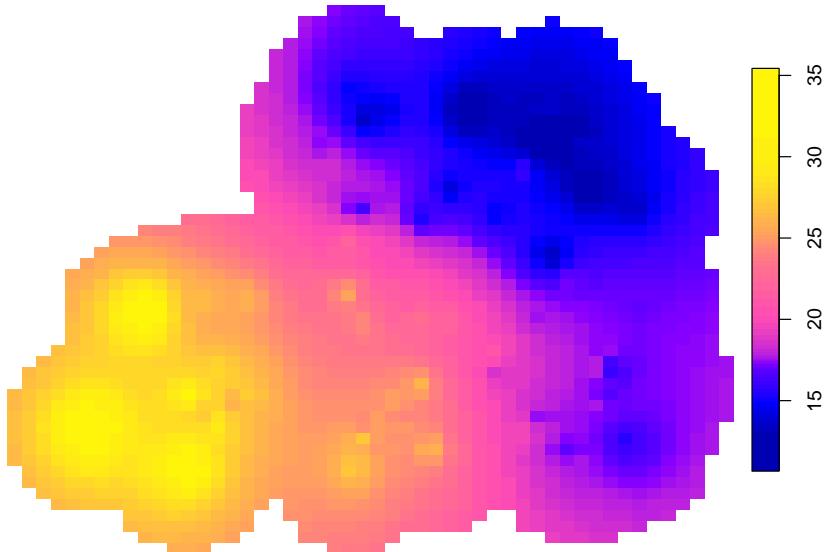


Figura 2.9: Interpolación por IDW (Inverse Distance Weighting) de los datos del acuífero Wolfcamp.

```

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')

```

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")

```

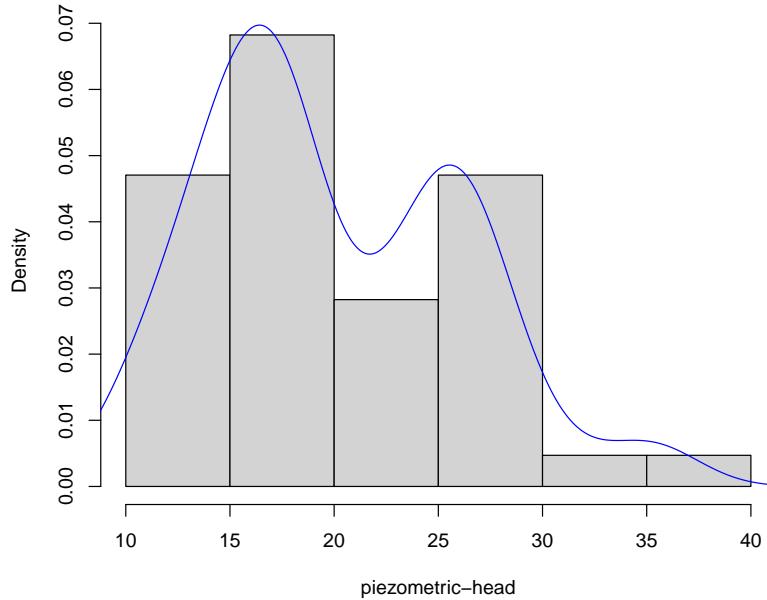


Figura 2.10: Distribución del nivel del agua subterránea en el acuífero Wolfcamp.

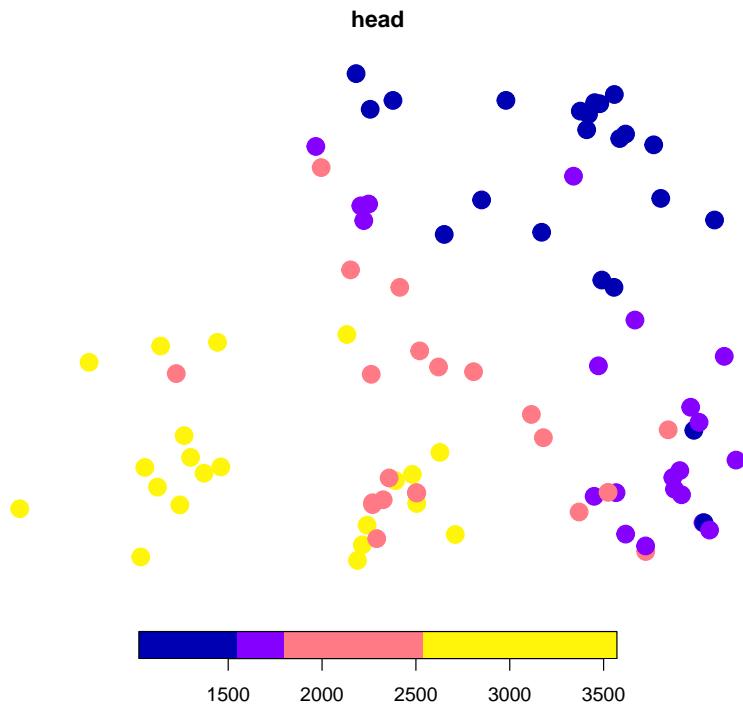


Figura 2.11: Distribución espacial de las observaciones del nivel del agua subterránea en el acuífero Wolfcamp.

```
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.

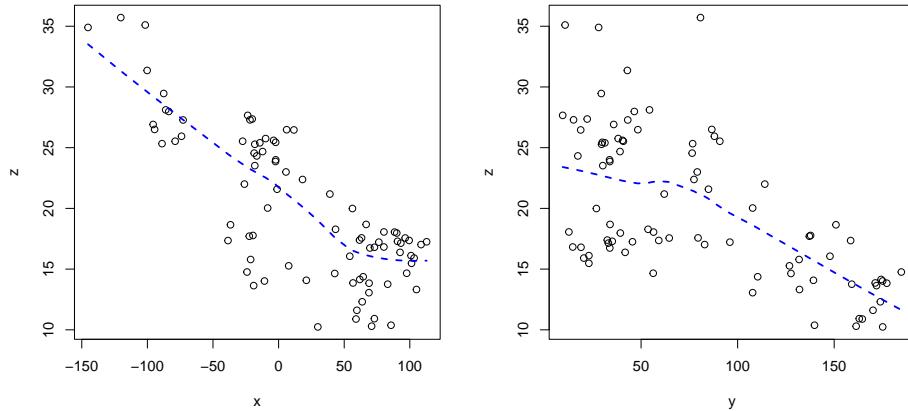


Figura 2.12: Gráficos de dispersión del nivel del agua subterránea frente a coordenadas (acuífero Wolfcamp).

Ejercicio 2.3 (Análisis exploratorio de la tendencia). Realizar un análisis exploratorio del conjunto de datos `s100` del paquete `geoR` (que contiene una simulación de un proceso espacial estacionario, sin tendencia; ver Sección 3.1).

Ejercicio 2.4 (Análisis exploratorio con variables explicativas). Realizar un análisis exploratorio del conjunto de datos `meuse_sf` (almacenado en el archivo `st_meuse.RData`; ver Figura 2.1) considerando como respuesta la concentración de zinc y como variables explicativas, además de las coordenadas espaciales, las variables que comparte con la rejilla `meuse_grid` (y que se podrían emplear en la predicción kriging; Capítulo 4).

Realizar también un análisis exploratorio multivariante, considerando la respuesta y el resto de variables explicativas (que podrían considerarse realizaciones de otros procesos espaciales y emplearlas para predicción multivariante, cokriging; Capítulo 5).

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

Capítulo 3

Modelado de procesos geoestadísticos

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 3.3.2. 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 3.3.3.

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 4.6 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 e.g. 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 ( posible ) 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"
```

En el resultado, la componente `dist` contiene los saltos, `gamma` las estimaciones del semivariograma (o las 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)
# Representar
oldpar <- par(mfrow = c(1, 2))
# Nube de semivarianzas clásicas
with(vario.cloud, plot(dist, gamma, xlab = "distance", ylab = "semivariances"))
# Nube de semivarianzas robustas
with(rvario.cloud, plot(dist, gamma, xlab = "distance", ylab = "semivariances"))

par(oldpar)

with(vario, plot(dist, gamma, pch = 19, ylim = c(0, 1),
                 xlab = "distance", ylab = "semivariance"))
with(rvario, points(dist, gamma))
legend("bottomright", c("clásico", "robusto"), pch = c(19, 1))
```

Para detectar observaciones atípicas podríamos emplear la nube de semivarianzas (preferiblemente las robustas, ya que tienen una distribución más próxima a la normalidad)²:

²Estableciendo `identify = TRUE` (o `digitize = TRUE`) en `plot.variogramCloud()` podríamos identificar semivarian-

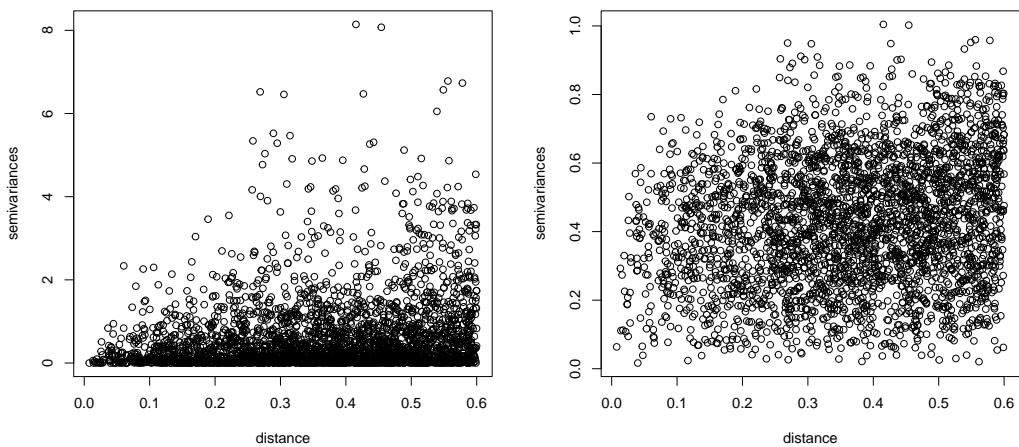


Figura 3.1: Nubes de semivarianzas clásicas (izquierda) y robustas (derecha) del conjunto de datos simulado.

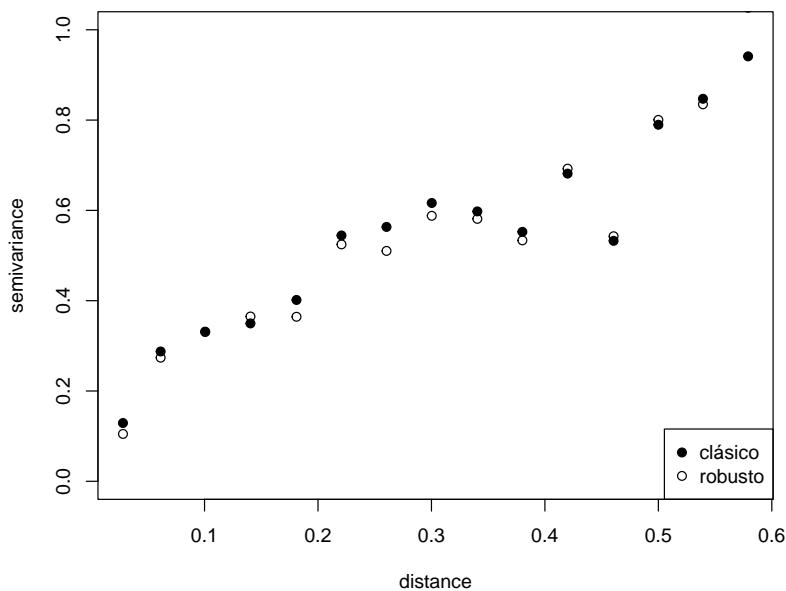
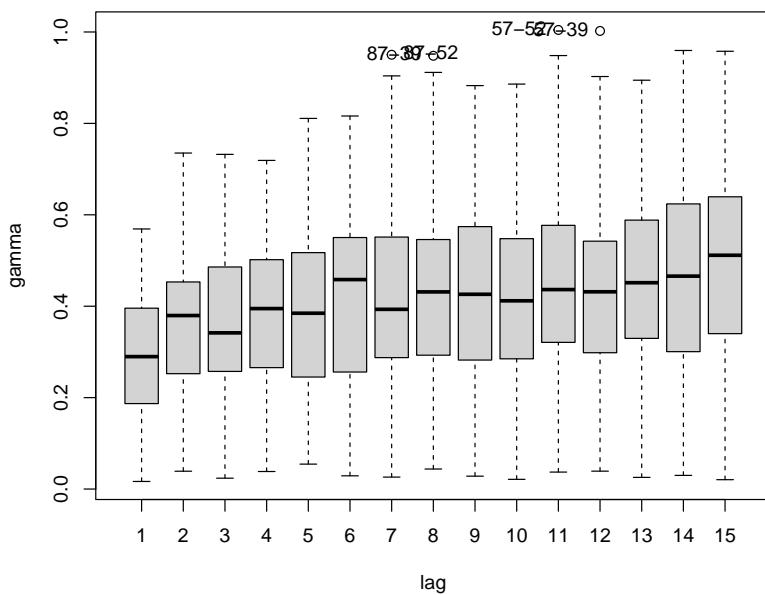


Figura 3.2: Estimaciones clásicas y robustas del semivariograma (datos simulados).

```
res <- as.data.frame(rvario.cloud)
boundaries <- attr(rvario, "boundaries")
res$lag <- cut(res$dist, breaks = boundaries, labels = seq_len(length(boundaries)-1))
res$labels <- with(res, paste(left, right, sep="-"))
with(res, car::Boxplot(gamma ~ lag, id = list(labels = labels)))
```

zas atípicas (o pares de datos atípicos) de forma interactiva.

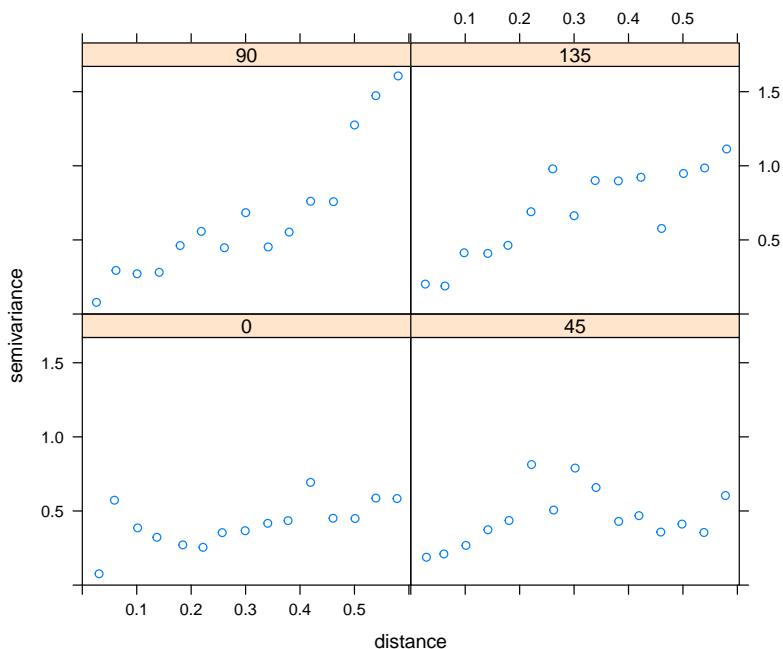


```
## [1] "87-52" "87-39" "57-52" "57-39"
```

Nos preocuparía especialmente la presencia de datos atípicos en saltos pequeños (indicaría que observaciones cercanas tienen valores muy distintos).

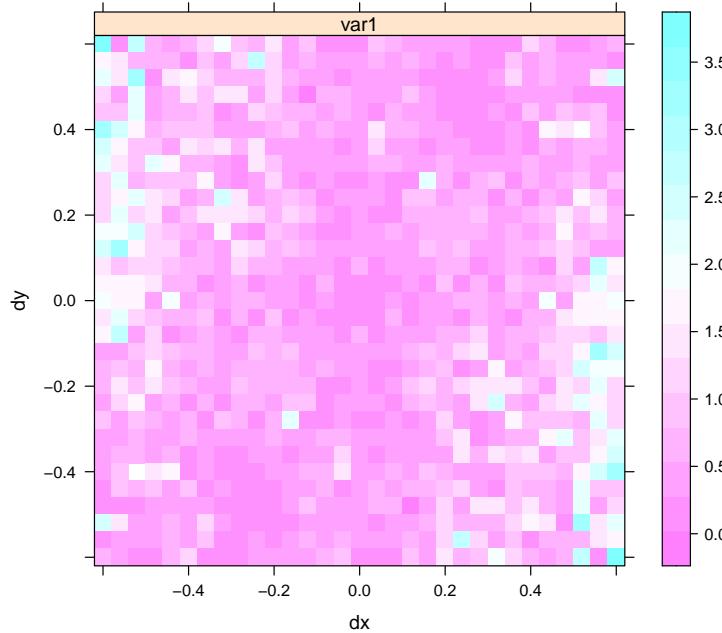
Para un análisis exploratorio de la anisotropía, podemos obtener variogramas direccionales indicando el ángulo y los grados de tolerancia en cada eje:

```
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).

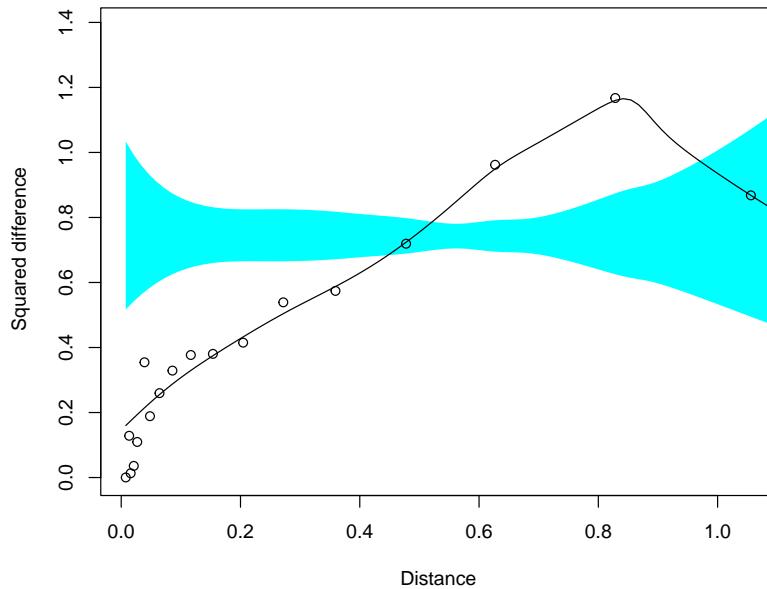


Figura 3.3: Estimaciones robustas y suavizadas del semivariograma, junto con una región de confianza para el semivariograma suponiendo que el proceso es independiente.

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 3.4 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 e.g. Wackernagel, 1998, pp. 120-123). El modelo exponencial se obtiene también como caso particular cuando $\lambda = 1$.

- Modelo oscilatorio:

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

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 e.g. 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 (para que tome los valores por defecto en el ajuste). Por defecto considerará que el nugget es nulo (y no se estimará), únicamente se considerará un efecto nugget si se especifica, aunque sea `nugget = NA`. 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()
```

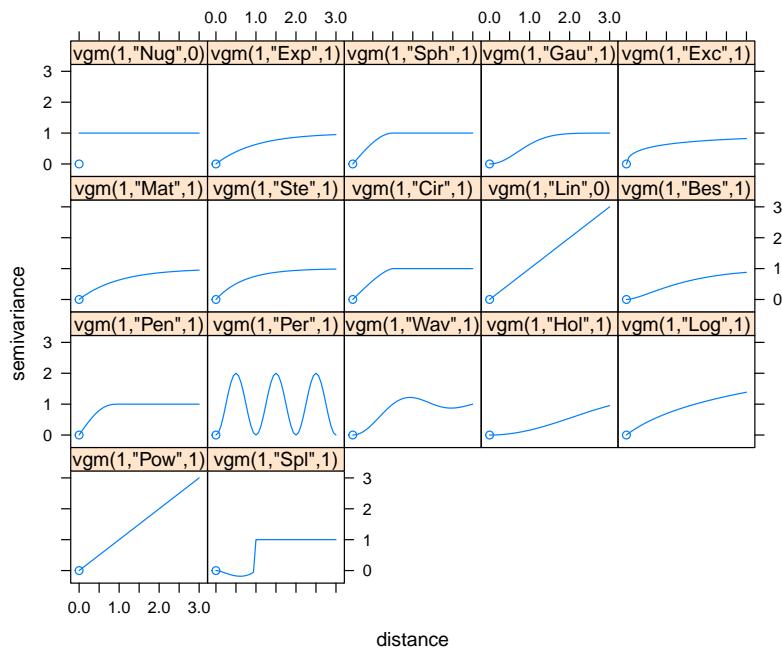


Figura 3.4: Representaciones de los modelos paramétricos isotrópicos de semivariogramas implementados en el paquete ‘gstat’.

```

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

```

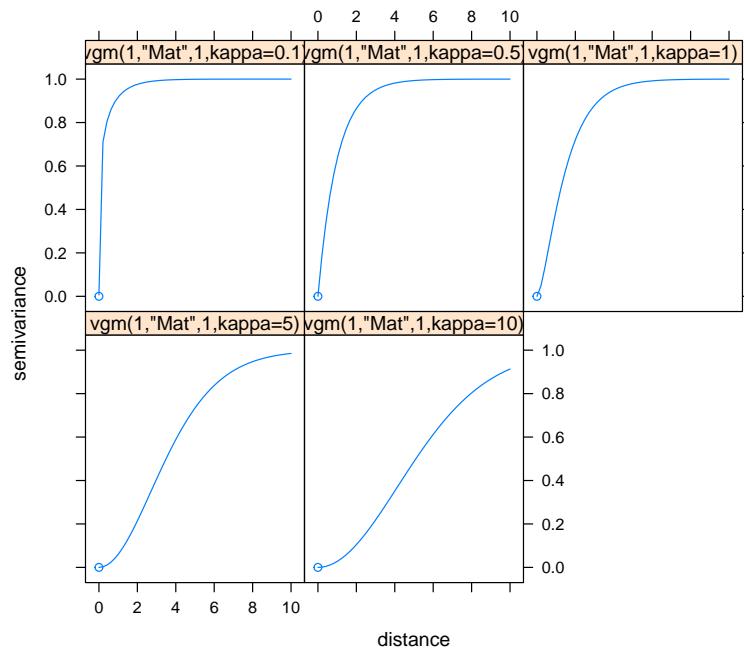


Figura 3.5: Modelo de Matérn con distintos valores del parámetro de suavizado.

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

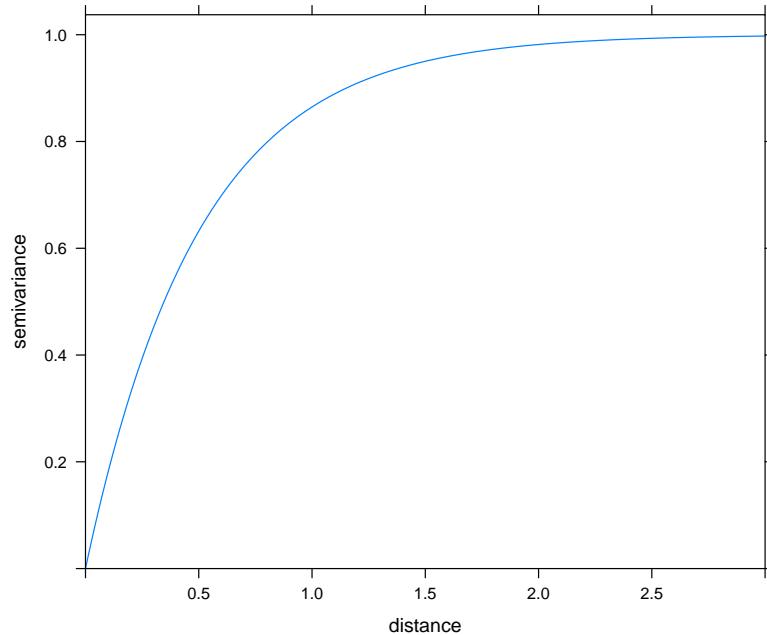


Figura 3.6: Ejemplo de modelo exponencial.

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 3.2.1). 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 e.g. 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).

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.
- *Anisotropía zonal*: cuando el umbral del semivariograma varía con la dirección (también se denomina anisotropía estratificada).
- *Anisotropía mixta*: 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 isotrópico³. En este caso se dice que el variograma es *geométricamente anisotrópico*. Por ejemplo, en el caso bidimensional, se suele considerar una matriz de la forma:

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

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

En `gstat` se puede definir⁴ anisotropía mediante el argumento `anis` de la función `vgm()`. En dos dimensiones es un vector con dos componentes `anis = c(alpha, ratio)`, `alpha` es el ángulo para la dirección principal de variabilidad (en grados, medido en el sentido del reloj partiendo de la dirección norte, i.e. `phi = (90 - alpha)*pi/180`) y `ratio` la relación entre el rango mínimo y máximo ($0 \leq ratio = a_2/a_1 \leq 1$).

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
## $ 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
```

³Esta 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 Guttorm (1992) consideraron transformaciones no lineales obtenidas mediante técnicas de escalamiento óptimo multidimensional.

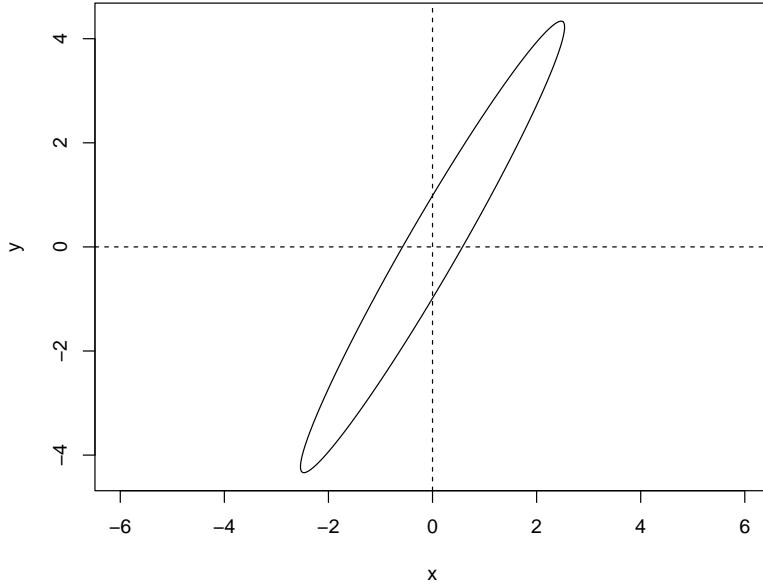
⁴Sin embargo no permite ajustar los parámetros de anisotropía, algo que se puede hacer con las herramientas implementadas en el paquete `geoR` (ver Sección B.3.2).

```

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 semiejes
  # 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 e.g. 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

A partir de las propiedades 1 y 2 del semivariograma mostradas en la Sección 1.3.2, se obtienen los denominados *modelos lineales de regionalización* (también *modelos anidados* o *nested models*):

$$\gamma(\mathbf{h}) = \sum_{k=0}^q b_k \gamma_k(\mathbf{h}),$$

siendo $b_k \geq 0$ y $\gamma_k(\mathbf{h})$ modelos básicos de semivariogramas, $k = 1, \dots, q$. La denominación de “modelos lineales” surge porque estos modelos se obtienen al suponer que el proceso espacial es una combinación lineal procesos espaciales intrínsecamente estacionarios mutuamente independientes.

Los modelos básicos suelen incluir un efecto nugget y algunos de los modelos mostrados en la sección anterior con efecto nugget nulo y umbral unidad. Además, cada modelo básico puede incorporar algún tipo de anisotropía (Sección 3.2.2), típicamente anisotropía geométrica:

$$\gamma_k(\mathbf{h}) \equiv \gamma_k(\|\mathbf{A}_k \mathbf{h}\|)$$

siendo $\mathbf{A}_k, k = 0, \dots, q$ matrices $d \times d$. De esta forma los modelos pueden ser lo suficientemente flexibles como para modelar la mayoría de situaciones que se pueden presentar en la práctica. Sin embargo, es difícil establecer un procedimiento automático (o semi-automático) para la selección y el ajuste de este tipo de modelos. Esto provoca que el proceso normalmente se realice en la práctica de forma interactiva por el usuario y utilizando principalmente herramientas gráficas⁵; siendo por tanto poco recomendables para algunos casos. En primer lugar hay que especificar el número y tipo de estructuras básicas, y en segundo lugar (aunque se suele hacer en la práctica de forma simultánea) está el problema de la estimación de los parámetros, donde puede ser especialmente complicado la determinación de los rangos y los parámetros de anisotropía de los distintos componentes (es de esperar que aparezcan problemas en la optimización).

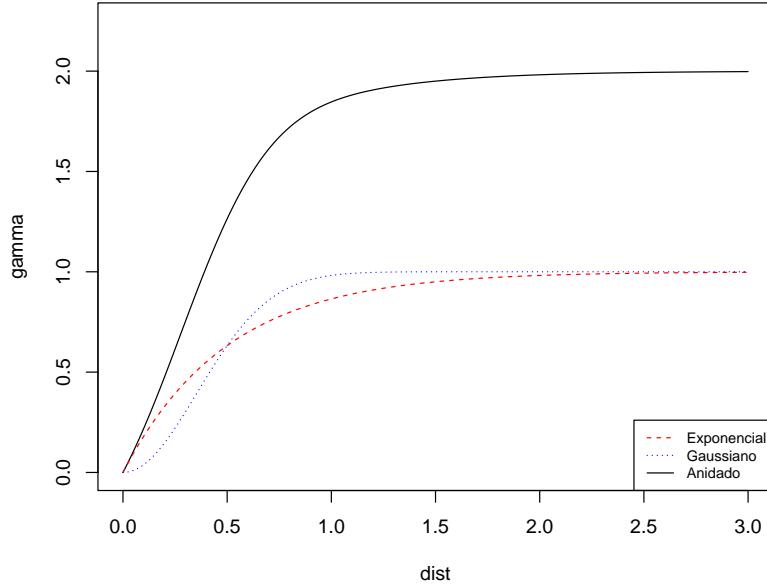
En gstat se pueden definir modelos de este tipo empleando el parámetro `add.to` de la función `vgm()`.

```
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

# Cuidado con plot.variogramModel() si se pretende añadir elementos
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)
```

⁵Ver por ejemplo Goovaerts (1997, Sección 4.2.4) para detalles sobre el uso en la práctica de éste tipo de modelos.



3.3 Ajuste de un modelo válido

Como ya se comentó anteriormente, en general los estimadores del variograma no pueden ser usados directamente en la predicción espacial; no son condicionalmente semidefinitos negativos y eso puede causar por ejemplo sistemas kriging inválidos o estimaciones negativas de la varianza kriging. Este problema normalmente se remedia buscando un modelo paramétrico válido que describa adecuadamente la dependencia espacial presente en los datos. Supongamos que $P = \{2\gamma(\mathbf{h}; \theta) : \theta \in \Theta\}$, donde $2\gamma(\mathbf{h}; \theta)$ es un variograma válido en \mathbb{R}^d (normalmente isotrópico), es la familia parametrizada de variogramas escogida. Se trata de encontrar el mejor elemento de P , para lo que se han propuesto diversos criterios de bondad de ajuste (ver e.g. Cressie, 1993, Sección 2.6). Entre ellos hay que destacar los basados en mínimos cuadrados y en máxima verosimilitud, descritos a continuación.

3.3.1 Estimación por mínimos cuadrados

Supongamos que $2\gamma(\mathbf{h}; \theta_0)$ es el variograma teórico y que $\hat{\gamma}_i = \hat{\gamma}(\mathbf{h}_i)$, $i = 1, \dots, K$, son las estimaciones del semivariograma obtenidas utilizando algún tipo de estimador piloto (e.g. alguno de los mostrados en la Sección 4.1.1). Normalmente, siguiendo las recomendaciones sugeridas por Journel y Huijbregts (1978, p. 194), solamente se consideran en el ajuste saltos menores o iguales que la mitad del máximo salto (i.e. $\|\mathbf{h}_i\| \leq \frac{1}{2} \max \{\|\mathbf{s}_k - \mathbf{s}_l\|\}$); y, si se utiliza el estimador empírico (o uno similar), de forma que el número de aportaciones a cada estimación sea por lo menos de treinta (i.e. $|N(\mathbf{h}_i)| \geq 30$). Habitualmente (e.g. Cressie, 1993, p. 96-97) la estimación por mínimos cuadrados de θ_0 se obtiene al minimizar:

$$(\hat{\gamma} - \gamma(\theta))^T \mathbf{V}(\theta) (\hat{\gamma} - \gamma(\theta)), \quad (3.3)$$

siendo $\hat{\gamma} = (\hat{\gamma}(\mathbf{h}_1), \dots, \hat{\gamma}(\mathbf{h}_K))^T$, $\gamma(\theta) = (\gamma(\mathbf{h}_1; \theta), \dots, \gamma(\mathbf{h}_K; \theta))^T$ y $\mathbf{V}(\theta)$ una matriz $K \times K$ semidefinida positiva que puede depender de θ , considerando alguno de los siguientes casos:

- Mínimos cuadrados ordinarios (OLS): $\mathbf{V}(\theta) = \mathbf{I}_K$, la matriz identidad $K \times K$.
- Mínimos cuadrados ponderados (WLS): $\mathbf{V}(\theta) = \text{diag}(w_1(\theta), \dots, w_K(\theta))$, con $w_i(\theta) \geq 0$, $i = 1, \dots, K$. Normalmente se suele tomar estos pesos inversamente proporcionales a $\text{Var}(\hat{\gamma}(\mathbf{h}_i))$.
- Mínimos cuadrados generalizados (GLS): $\mathbf{V}(\theta) = \Sigma_{\hat{\gamma}}(\theta)^{-1}$, la inversa de la matriz de covarianzas (asintótica) de $\hat{\gamma}$ obtenida suponiendo que el variograma teórico es $2\gamma(\mathbf{h}; \theta)$.

Es importante señalar que al utilizar el criterio GLS el cálculo de la matriz de covarianzas $\Sigma_{\hat{\gamma}}(\theta)$ generalmente no resulta fácil (por ejemplo en Cressie 1993, p. 96, se tienen las expresiones para el estimador empírico y el estimador robusto, suponiendo normalidad). Esto produce que la minimización de la función objetivo (3.3) sea computacionalmente prohibitiva en muchos casos. El método de mínimos cuadrados ponderados puede verse como un compromiso entre la eficiencia del método de GLS y la simplicidad del método de OLS. Además, suponiendo normalidad y que el variograma teórico es $2\gamma(\mathbf{h}; \theta)$, Cressie (1985) probó que:

$$Var(\hat{\gamma}(\mathbf{h}_i)) \simeq 2 \frac{\gamma(\mathbf{h}_i; \theta)^2}{|N(\mathbf{h}_i)|},$$

en el caso del estimador empírico; y para el estimador robusto:

$$Var(\tilde{\gamma}(\mathbf{h}_i)) \simeq 2.885 \frac{\gamma(\mathbf{h}_i; \theta)^2}{|N(\mathbf{h}_i)|},$$

siendo esta aproximación incluso mejor que en el caso anterior. Proponiendo en estos casos la minimización de:

$$\sum_{i=1}^K w_i(\theta) (\hat{\gamma}(\mathbf{h}_i) - \gamma(\mathbf{h}_i; \theta))^2,$$

siendo $w_i(\theta) = |N(\mathbf{h}_i)| / \gamma(\mathbf{h}_i; \theta)^2$, como aproximación al criterio WLS.

Estos métodos de ajuste tiene unas propiedades interesantes, cuanto mayor sea $|N(\mathbf{h}_i)|$ mayor peso recibe el residuo en el salto \mathbf{h}_i y además, cuanto más pequeño sea el valor del variograma teórico mayor peso recibe también el residuo correspondiente. Por este motivo, los saltos próximos al origen típicamente reciben mayor peso con lo que se consigue un buen ajuste del modelo de variograma cerca del origen (esto es especialmente importante; ver e.g. Stein, 1988, y comentarios en la Sección 4.5). Adicionalmente estos métodos pueden ser implementados fácilmente en la práctica (de forma similar al OLS).

Aunque para obtener las expresiones (o aproximaciones) de las varianzas y covarianzas de las estimaciones piloto se supone habitualmente que la distribución de los datos es normal, se puede probar fácilmente que los procedimientos de ajuste obtenidos son también válidos para el caso de datos normales transformados (ver e.g. Cressie, 1993, p. 98). Esta es una de las principales ventajas de los métodos WLS o GLS frente a otras alternativas (como los métodos basados en máxima verosimilitud); como utilizan solamente la estructura de segundo orden (asintótica) del estimador del variograma, no es necesario hacer suposiciones sobre la distribución completa de los datos⁶.

Como comentario final, en la función objetivo (3.3) de los criterios WLS y GLS anteriores, la matriz de pesos utilizada en el ajuste $\mathbf{V}(\theta)$ depende también del parámetro sobre el que se realiza la minimización (y al minimizar (3.3) en cierto sentido se están maximizando también las varianzas), por lo que puede ser preferible utilizar un algoritmo iterativo. Por ejemplo comenzar con pesos OLS (o WLS con $w_i = |N(\mathbf{h}_i)| / \|\mathbf{h}_i\|^2$) y posteriormente en cada etapa k obtener una nueva aproximación $\hat{\theta}_0^{(k)}$ al minimizar:

$$(\hat{\gamma} - \gamma(\theta))^T \mathbf{V}(\hat{\theta}_0^{(k-1)}) (\hat{\gamma} - \gamma(\theta)),$$

repitiendo este proceso hasta convergencia (realmente muchos de los algoritmos diseñados para el ajuste por mínimos cuadrados proceden de esta forma).

En `gstat` el ajuste OLS y WLS se realiza mediante la función:

```
fit.variogram(object, model, fit.sills = TRUE, fit.ranges = TRUE,
               fit.method = 7, fit.kappa = FALSE, ...)
```

- **object**: semivariograma empírico, obtenido con la función `variogram()`.
- **model**: modelo de semivariograma, generado con la función `vgm()`.

⁶La distribución y eficiencia asintótica de los estimadores mínimo cuadráticos ha sido estudiada por Lahiri et al. (2003), demostrando su consistencia y normalidad asintótica bajo condiciones muy generales.

- `fit.sills`, `fit.ranges`, `fit.kappa`: determinan si se ajustan los correspondientes parámetros (TRUE) o se mantienen fijos (FALSE).
- `fit.method`: selección de los pesos en el criterio WLS.
 - `fit.method = 6`: $w_i = 1$, OLS.
 - `fit.method = 1`: $w_i = |N(\mathbf{h}_i)|$.
 - `fit.method = 7`: $w_i = |N(\mathbf{h}_i)| / \|\mathbf{h}_i\|^2$.
 - `fit.method = 2`: $w_i = |N(\mathbf{h}_i)| / \gamma(\mathbf{h}_i; \theta)^2$.

Los parámetros iniciales se fijan a los establecidos en `model`. Si alguno es desconocido (NA), le asigna un valor por defecto:

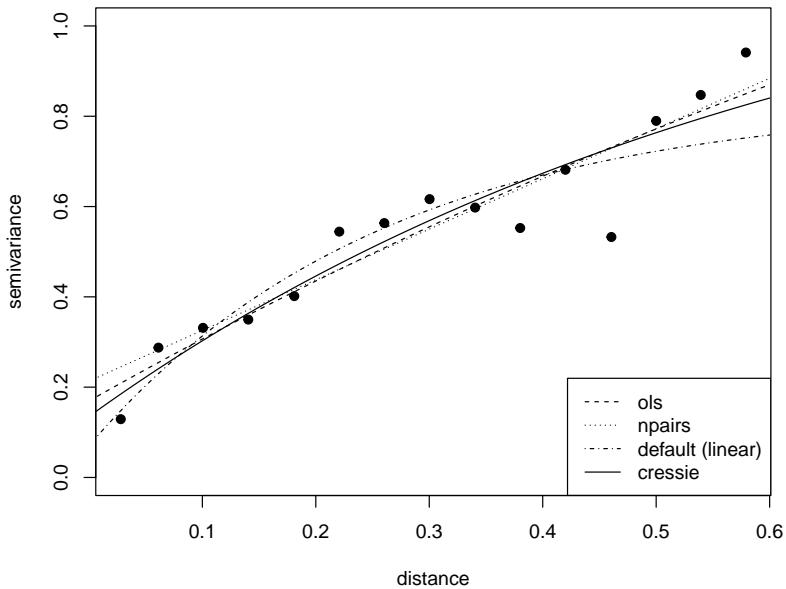
- el rango se establece a 1/3 de la distancia máxima del variograma empírico,
- al umbral parcial se le asigna el promedio de los últimos 5 valores del variograma empírico,
- y el efecto nugget (siempre que haya sido establecido explícitamente con `nugget = NA`) se toma como la media de los tres primeros valores del variograma empírico.

Como ejemplo, a continuación se ajusta un modelo exponencial al variograma empírico calculado en la Sección 3.1, mediante OLS y WLS con diferentes pesos:

```
modelo <- vgm(model = "Exp", nugget = NA) # Valores iniciales por defecto, incluyendo nugget
# modelo <- vgm(psill = 0.6, model = "Exp", range = 0.2, nugget = 0.0) # Valores iniciales
fit.ols <- fit.variogram(vario, model = modelo, fit.method = 6)
# fit.npairs <- fit.variogram(vario, model = modelo, fit.method = 1) # Warning: No convergence
fit.npairs <- fit.variogram(vario, model = fit.ols, fit.method = 1)

## Warning in fit.variogram(vario, model = fit.ols, fit.method = 1): No convergence
## after 200 iterations: try different initial values?

fit.lin <- fit.variogram(vario, model = modelo, fit.method = 7)
fit <- fit.variogram(vario, model = fit.lin, fit.method = 2)
# Representar:
# Cuidado con plot.variogramModel() si se pretende añadir elementos
plot(vario$dist, vario$gamma, xlab = "distance", ylab = "semivariance",
     pch = 19, ylim = c(0, 1))
lines(variogramLine(fit.ols, maxdist = 0.6), lty = 2)
lines(variogramLine(fit.npairs, maxdist = 0.6), lty = 3)
lines(variogramLine(fit.lin, maxdist = 0.6), lty = 4)
lines(variogramLine(fit, maxdist = 0.6))
legend("bottomright", c("ols", "npairs", "default (linear)", "cressie"), lty = c(2, 3, 4, 1))
```



En general, se recomendaría emplear pesos inversamente proporcionales a la varianza (`fit.method = 2`). Si quisiésemos comparar el ajuste de distintos modelos (con el mismo criterio de ajuste) se podría considerar el valor mínimo de la función objetivo WLS, almacenado como un atributo del resultado (aunque la recomendación sería emplear validación cruzada, Sección 4.6):

```
attr(fit, "SSErr")
```

```
## [1] 52.83535
```

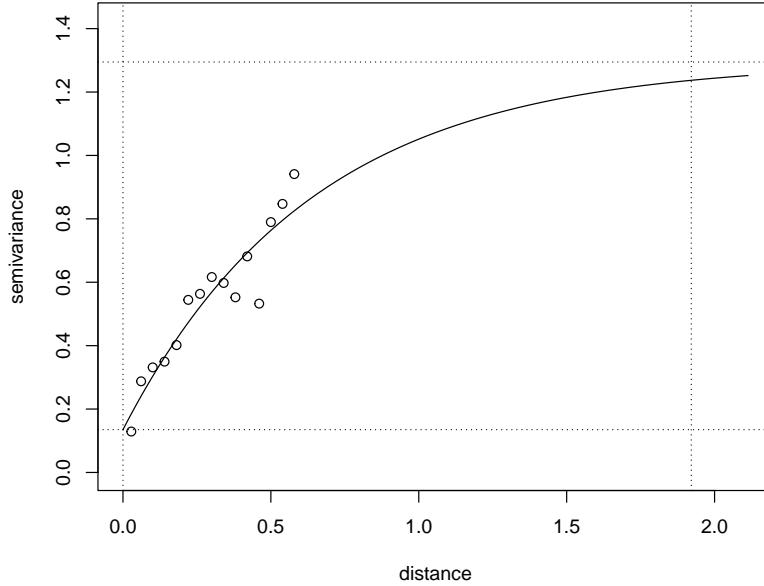
Imprimiendo el resultado del ajuste obtenemos las estimaciones de los parámetros, que podríamos interpretar (ver Sección 1.3.1 y Sección 4.5.2).

```
fit
```

```
##   model    psill      range
## 1 Nug 0.13495 0.0000000
## 2 Exp 1.15982 0.6403818
nugget <- fit$psill[1]
sill <- nugget + fit$psill[2]
range <- 3*fit$range[2] # Parámetro de escala en model = "Exp"
```

NOTA: Cuidado, en el caso del modelo exponencial, el parámetro que aparece como `range` es un parámetro de escala proporcional al verdadero rango práctico (tres veces ese valor).

```
plot(vario$dist, vario$gamma, xlab = "distance", ylab = "semivariance",
     xlim = c(0, range*1.1), ylim = c(0, sill*1.1))
lines(variogramLine(fit, maxdist = range*1.1))
abline(v = 0, lty = 3)
abline(v = range, lty = 3)
abline(h = nugget, lty = 3)
abline(h = sill, lty = 3)
```



En gstat el ajuste con el criterio GLS se podría realizar mediante la función:

```
fit.variogram.gls(formula, data, model, maxiter = 30, eps = .01,
                   trace = TRUE, ignoreInitial = TRUE, cutoff = Inf, plot = FALSE)
```

Sin embargo, actualmente solo admite datos tipo `Spatial*` del paquete `sp` y además es habitual que aparezcan problemas computacionales, por lo que no se recomendaría su uso.

```
fit.variogram.gls(z ~ 1, as(datos, "Spatial"), modelo,
                   maxiter = 2, cutoff = 0.6, plot = TRUE)
# Error in if (any(model$range < 0)) { : missing value where TRUE/FALSE needed
```

3.3.2 Modelado del variograma en procesos no estacionarios

Como ya se comentó en la introducción de este capítulo, si no se puede asumir que la tendencia es constante no es apropiado utilizar directamente los estimadores del semivariograma mostrados en la Sección 3.1. Por ejemplo, considerando el modelo lineal (1.2) de la Sección 1.2.1 (el modelo del *kriging universal*, Sección 4.3; que emplearemos en el resto de este capítulo), tendríamos que:

$$E(Z(\mathbf{s}_1) - Z(\mathbf{s}_2))^2 = 2\gamma(\mathbf{s}_1 - \mathbf{s}_2) + \left(\sum_{j=0}^p \beta_j (X_j(\mathbf{s}_1) - X_j(\mathbf{s}_2)) \right)^2.$$

Muchas veces, cuando las estimaciones experimentales del semivariograma aparecen no estar acotadas (e.g. Figura 3.7 izquierda), es debido a que la tendencia no está especificada correctamente.

El procedimiento habitual en geoestadística es eliminar la tendencia y estimar el variograma a partir de los residuos. Por ejemplo, en este caso, podríamos considerar los residuos de un ajuste OLS de la tendencia:

$$\mathbf{r}_{ols} = \mathbf{Z} - \mathbf{X}\hat{\boldsymbol{\beta}}_{ols} = (\mathbf{I}_n - (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top) \mathbf{Z}.$$

Esto se puede hacer con la función `variogram()` del paquete `gstat` especificando la fórmula del modelo como primer argumento (ver Figura 3.7 derecha).

Como ejemplo consideraremos los datos del acuífero Wolfcamp:

```

load("datos/aquifer.RData")
library(sf)
aquifer$head <- aquifer$head/100 # en cientos de pies
aquifer_sf <- st_as_sf(aquifer, coords = c("lon", "lat"), remove = FALSE, agr = "constant")
# maxlag <- 0.5*sqrt(sum(diff(matrix(st_bbox(aquifer_sf), nrow = 2, byrow = TRUE))^2))

vario.est <- variogram(head ~ 1, aquifer_sf, cutoff = 150)
vario.resid <- variogram(head ~ lon + lat, aquifer_sf, cutoff = 150)
oldpar <- par(mfrow = c(1, 2))
# plot(vario.est) # no compatible con mfrow
with(vario.est, plot(dist, gamma, xlab = "distance", ylab = "semivariance"))
# plot(vario.resid)
with(vario.resid, plot(dist, gamma, xlab = "distance", ylab = "semivariance"))

```

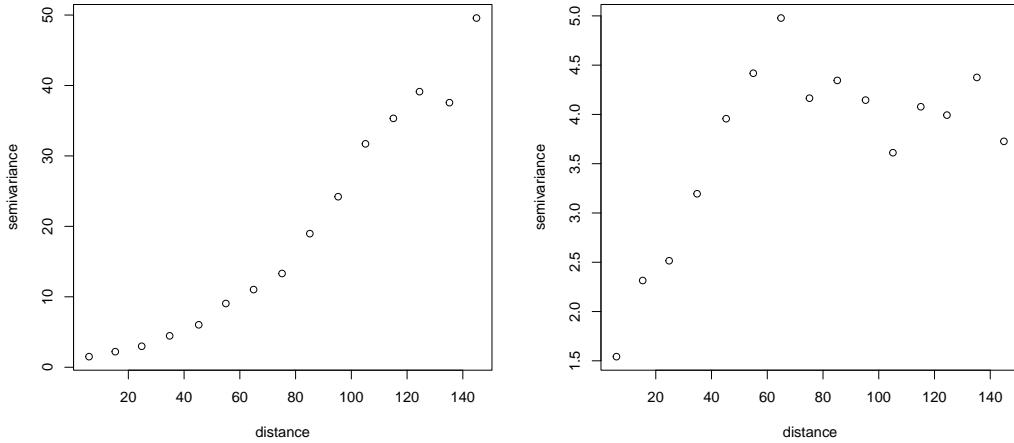


Figura 3.7: Semivariograma empírico obtenido asumiendo media constante (izquierda) y a partir de los residuos de un ajuste lineal de la tendencia (derecha), empleando los datos del acuífero Wolfcamp.

```
par(oldpar)
```

El ajuste por WLS se puede realizar también con la función `fit.variogram()`:

```

modelo <- vgm(model = "Sph", nugget = NA) # Valores iniciales por defecto
# modelo <- vgm(psill = 3, model = "Sph", range = 75, nugget = 0)
fit.resid <- fit.variogram(vario.resid, modelo, fit.method = 2)
fit.resid

##   model   psill    range
## 1   Nug 1.095133  0.00000
## 2   Sph 3.044034 63.39438

# Cuidado con plot.variogramModel() si se pretende añadir elementos
# plot(fit.resid, cutoff = 150, ylim = c(0, 4.5))
# with(vario.resid, points(dist, gamma))
with(vario.resid, plot(dist, gamma, xlab = "distance", ylab = "semivariance",
                      xlim = c(0, 150), ylim = c(0, 5)))
lines(variogramLine(fit.resid, maxdist = 150))

```

Sin embargo, para poder estimar la tendencia de forma eficiente sería necesario conocer la dependencia (i.e. conocer $\gamma(\cdot)$), que dependería a su vez de la estimación de la tendencia. Para solventar este problema circular, Neuman y Jacobson (1984) propusieron una aproximación iterativa, empezar con

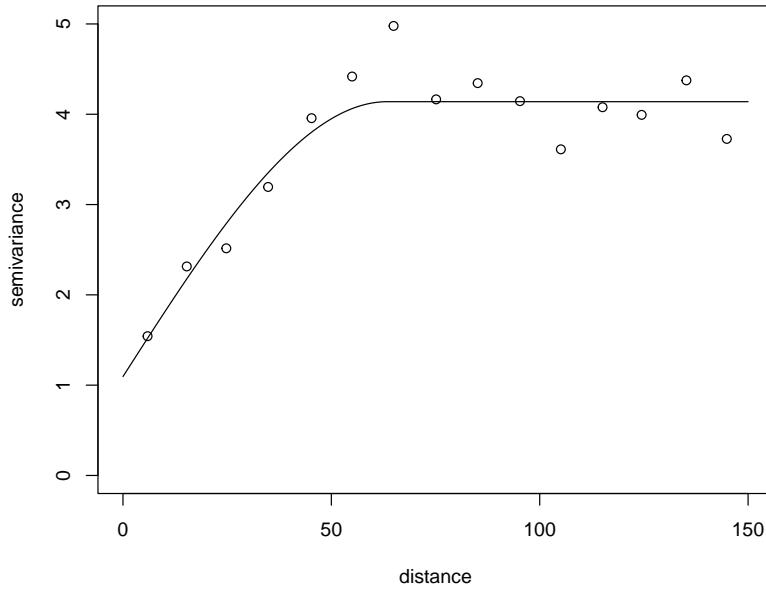


Figura 3.8: Ajuste de un modelo esférico de semivariograma a las estimaciones empíricas obtenidas a partir de los residuos de un ajuste lineal de la tendencia, empleando los datos del acuífero Wolfcamp.

el estimador OLS de θ , estimar el variograma a partir de los residuos, ajustar un modelo de variograma válido, calcular el estimador GLS basado en el modelo ajustado y así sucesivamente hasta convergencia. En la práctica este procedimiento suele converger en pocas iteraciones (normalmente menos de 5). Sin embargo, en el paquete `gstat` solo se realiza una iteración (se reestimará la tendencia empleando GLS al calcular las predicciones kriging).

En el caso de variogramas no acotados, el proceso $\varepsilon(\cdot)$ no sería estacionario de segundo orden, no está disponible la matriz Σ y en principio sería imposible emplear GLS para estimar la tendencia. Sin embargo, normalmente se suele trabajar en un dominio acotado D y podemos encontrar una constante positiva A tal que $C^*(\mathbf{h}) = A - \gamma(\mathbf{h}) \geq 0, \forall \mathbf{h} \in D$ (y por tanto esta función es un covariograma válido en ese dominio). La función $C^*(\mathbf{h})$ se suele denominar *pseudo-covariograma* (o covarianza localmente equivalente; ver e.g. Chilès y Delfiner, 1999, Sección 4.6.2). Si utilizamos $C^*(\mathbf{h})$ en lugar del covariograma en la estimación de la media (o en las ecuaciones del predictor del KU), la constante A se cancela y obtenemos los mismos resultados (sin embargo las varianzas si que dependen de esta constante).

Adicionalmente, habría que tener en cuenta también que la variabilidad de los residuos no es la de los errores teóricos (algo que normalmente se ignora). Para ilustrar este problema supongamos que el proceso de error $\varepsilon(\cdot)$ es estacionario de segundo orden con covariograma conocido $C(\cdot)$ de forma que podemos calcular el estimador lineal óptimo de β :

$$\hat{\beta}_{gls} = (\mathbf{X}^\top \Sigma^{-1} \mathbf{X})^{-1} \mathbf{X}^\top \Sigma^{-1} \mathbf{Z} = \mathbf{P}_{gls} \mathbf{Z},$$

siendo \mathbf{P}_{gls} la matriz de proyección. Empleando este estimador obtenemos el vector de residuos:

$$\mathbf{r} = \mathbf{Z} - \mathbf{X} \hat{\beta}_{gls} = (\mathbf{I}_n - \mathbf{P}_{gls}) \mathbf{Z},$$

cuya matriz de varianzas-covarianzas resulta ser:

$$\begin{aligned} Var(\mathbf{r}) &= (\mathbf{I}_n - \mathbf{P}_{gls}) \Sigma (\mathbf{I}_n - \mathbf{P}_{gls})^\top \\ &= \Sigma - \mathbf{X} (\mathbf{X}^\top \Sigma^{-1} \mathbf{X})^{-1} \mathbf{X}^\top. \end{aligned}$$

De donde se deduce que si utilizamos directamente los residuos, incluso procediendo de la forma más eficiente, se introduce un sesgo en la estimación de la dependencia espacial. Explícitamente, si

denotamos por $\hat{\mu}(\mathbf{s})$ la estimación GLS de la tendencia, puede verse que:

$$\begin{aligned} C_{\mathbf{r}}(\mathbf{s}_i, \mathbf{s}_j) &= Cov(Z(\mathbf{s}_i) - \hat{\mu}(\mathbf{s}_i), Z(\mathbf{s}_j) - \hat{\mu}(\mathbf{s}_j)) \\ &= C(\mathbf{s}_i - \mathbf{s}_j) - Cov(\hat{\mu}(\mathbf{s}_i), \hat{\mu}(\mathbf{s}_j)), \end{aligned}$$

y expresado en función del semivariograma:

$$\gamma_{\mathbf{r}}(\mathbf{s}_i, \mathbf{s}_j) = \gamma(\mathbf{s}_i - \mathbf{s}_j) - \frac{1}{2}Var(\hat{\mu}(\mathbf{s}_i) - \hat{\mu}(\mathbf{s}_j)).$$

Por tanto al utilizar alguno de los estimadores mostrados anteriormente con los residuos estimados obtenemos estimaciones sesgadas del semivariograma teórico. Matheron (1971, pp. 152-155) ya notó que, por lo general, el sesgo del estimador del semivariograma es pequeño en los saltos próximos al origen, pero más sustancial en saltos grandes⁷. Parece ser que este problema provocó una desilusión con el kriging universal y la iniciativa hacia el kriging con funciones intrínsecamente estacionarias (ver e.g. Matheron, 1973; Cressie, 1993, Sección 5.4; o Chilès y Delfiner, 1999, cap. 4).

En cuanto a las consecuencias de que el estimador del variograma no sea insesgado en el kriging universal, hay que tener en cuenta que:

- Al ajustar un modelo de variograma por mínimos cuadrados ponderados o generalizados (Sección 3.3.1), automáticamente los saltos pequeños reciben mayor peso en el ajuste.
- Además si la predicción espacial se lleva a cabo con un criterio de vecindad, el variograma sólo es evaluado en saltos pequeños, donde se tiene una buena estimación y un buen ajuste.
- También hay que tener en cuenta el resultado de Stein (1988), i.e. para una predicción eficiente generalmente sólo es necesario capturar la conducta del variograma cerca del origen

Por lo anterior, el desencanto con el kriging universal ha sido prematuro, el efecto del sesgo del estimador del variograma sobre el predictor del kriging universal es pequeño. Sin embargo la varianza del kriging universal se ve más afectada y es menor de lo que debería ser (para más detalles ver Cressie, 1993, pp. 296-299). Adicionalmente se han propuesto alternativas a los métodos de ajuste basados en mínimos cuadrados que tienen en cuenta el sesgo en la estimación del variograma (e.g. Beckers y Bogaert, 1998; Fernandez-Casal y Francisco-Fernandez, 2014, función `npsp::np.svariso.cor()`).

Otra alternativa sería asumir normalidad y estimar ambos componentes de forma conjunta empleando alguno de los métodos basados en máxima verosimilitud descritos en la siguiente sección (que también tienen problemas de sesgo).

3.3.3 Estimación por máxima verosimilitud

La estimación por máxima verosimilitud (*maximum likelihood*, ML) es un método muy conocido en inferencia estadística paramétrica, aunque su uso en geoestadística ha sido relativamente reciente (a partir de mediados de los 80). Además la estimación ML tiene una conexión directa con la estimación Bayesiana (e.g. Handcock y Wallis, 1994) y el empleo de estas herramientas en estadística espacial ha experimentado un notable aumento en los últimos años (e.g. Wikle et al., 2019; Moraga, 2020).

Si suponemos que la distribución de los datos es normal:

$$\mathbf{Z} \sim \mathcal{N}(\mathbf{X}\beta, \Sigma),$$

donde $\Sigma = \Sigma(\theta)$ (utilizando la notación de secciones anteriores), se puede deducir fácilmente la expresión de la función de verosimilitud y obtener las estimaciones de los parámetros buscando los valores que la maximizan.

En este caso, la función de densidad de los datos es:

$$f(\mathbf{z}) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{X}\beta)^\top \Sigma^{-1} (\mathbf{z} - \mathbf{X}\beta) \right\}.$$

⁷Para un caso particular, Cressie (1993, pp. 166-167) observó que los residuos basados en el estimador GLS dan lugar a un estimador del variograma con sesgo negativo y cuadrático en h.

Además en la mayoría de los casos podemos reparametrizar el covariograma⁸ de forma que:

$$\Sigma = \sigma^2 \mathbf{V}(\theta),$$

siendo σ^2 la varianza desconocida (o umbral total), y se obtiene que la expresión del logaritmo negativo de la función de verosimilitud (*negative log likelihood*, NLL) es:

$$\begin{aligned}\mathcal{L}(\theta, \beta, \sigma^2 | \mathbf{Z}) &= \frac{n}{2} \ln(2\pi) + \frac{n}{2} \ln(\sigma^2) + \frac{1}{2} \ln |\mathbf{V}(\theta)| \\ &\quad + \frac{1}{2\sigma^2} (\mathbf{Z} - \mathbf{X}\beta)^{\top} \mathbf{V}(\theta)^{-1} (\mathbf{Z} - \mathbf{X}\beta),\end{aligned}$$

donde $|\mathbf{V}(\theta)|$ denota el determinante de la matriz $\mathbf{V}(\theta)$. Las estimaciones de los parámetros $(\theta, \beta, \sigma^2)$ se obtendrán minimizando el NLL. Un resultado bien conocido es que el mínimo se obtiene, independientemente de θ , para:

$$\begin{aligned}\hat{\beta} &= (\mathbf{X}^{\top} \mathbf{V}(\theta)^{-1} \mathbf{X})^{-1} \mathbf{X}^{\top} \mathbf{V}(\theta)^{-1} \mathbf{Z}, \\ \hat{\sigma}^2 &= \frac{1}{n} (\mathbf{Z} - \mathbf{X}\hat{\beta})^{\top} \mathbf{V}(\theta)^{-1} (\mathbf{Z} - \mathbf{X}\hat{\beta}).\end{aligned}\tag{3.4}$$

Por tanto la función a minimizar respecto a θ es:

$$\mathcal{L}(\theta | \mathbf{Z}) = \mathcal{L}(\hat{\beta}, \theta, \hat{\sigma}^2 | \mathbf{Z}) = \frac{n}{2} \ln(2\pi) + \frac{n}{2} \ln(\hat{\sigma}^2) + \frac{1}{2} \ln |\mathbf{V}(\theta)| + \frac{n}{2}.$$

Para ello es necesario utilizar algoritmos de minimización no lineal multidimensional. Además está el problema de la posible multimodalidad de esta función (ver e.g. Mardia y Watkins, 1989), por tanto habría que asegurarse de que el algoritmo elegido no converge a un mínimo local. Si $\hat{\theta}$ es la estimación de θ obtenida al resolver este problema, sustituyendo en (3.4) se obtienen las estimaciones del resto de parámetros⁹.

Uno de los principales problemas de la estimación ML es que los estimadores de σ^2 y θ pueden tener un sesgo considerable (especialmente cuando la tendencia no es constante), algo que es bastante conocido en la estimación de la varianza con datos independientes. Este problema se puede resolver (por lo menos en parte) utilizando una variante de este método.

El método de máxima verosimilitud restringida (*restricted maximum likelihood*, REML) se basa en la idea de filtrar los datos de forma que la distribución conjunta no dependa de β . Se trata de maximizar la verosimilitud de $m = n - p - 1$ contrastes de error linealmente independientes:

$$\mathbf{Y} = \Lambda \mathbf{Z},$$

siendo Λ una matriz $m \times n$ de rango m y tal que $\Lambda \mathbf{X} = \mathbf{0}$ (i.e. $E(\mathbf{Y}) = \mathbf{0}$). Estas combinaciones lineales también se denominan habitualmente *incrementos generalizados* y, asumiendo normalidad, su distribución no depende de β :

$$\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \Lambda \Sigma \Lambda^{\top}).$$

De forma análoga al caso anterior podríamos obtener la correspondiente función de verosimilitud. Además, Harville (1974) demostró que las verosimilitudes de distintos incrementos generalizados son iguales salvo una constante y que se pueden obtener expresiones simplificadas seleccionando la matriz Λ de forma que $\Lambda^{\top} \Lambda = \mathbf{I}_n - \mathbf{X}(\mathbf{X}^{\top} \mathbf{X})^{-1} \mathbf{X}^{\top}$ y $\Lambda \Lambda^{\top} = \mathbf{I}_m$, obteniéndose la siguiente expresión para el NLL:

$$\begin{aligned}\mathcal{L}(\theta | \mathbf{Y}) &= \frac{m}{2} \ln(2\pi) + \frac{m}{2} \ln(\hat{\sigma}_Y^2) - \frac{1}{2} |\mathbf{X}^{\top} \mathbf{X}| \\ &\quad + \frac{1}{2} |\mathbf{X}^{\top} \mathbf{V}(\theta)^{-1} \mathbf{X}| + \frac{1}{2} \ln |\mathbf{V}(\theta)| + \frac{m}{2},\end{aligned}$$

⁸Por ejemplo en el caso de los semivariogramas mostrados en la Sección 3.2.1, si c_0 es el efecto nugget y c_1 el umbral parcial, en lugar de éstos parámetros se considerarían la varianza (umbral) $\sigma^2 = c_0 + c_1$ y la proporción de nugget en el umbral total $c_0^* = c_0/(c_0 + c_1)$ (lo que equivale a suponer en las expresiones del covariograma que $c_1 = 1 - c_0$ y $0 \leq c_0 \leq 1$).

⁹El comportamiento asintótico (bajo dominio creciente) de estos estimadores ha sido estudiado por Mardia y Marshal (1984), dando condiciones (no muy fáciles de chequear en la práctica) para su consistencia y normalidad asintótica (ver también Cressie, 1993, Sección 7.3.1).

siendo:

$$\hat{\sigma}_Y^2 = \frac{1}{m}(\mathbf{Z} - \mathbf{X}\hat{\beta})^\top \mathbf{V}(\theta)^{-1}(\mathbf{Z} - \mathbf{X}\hat{\beta}).$$

En general se da por hecho que la estimación REML mejora, a veces significativamente, los resultados obtenidos con la estimación ML (sobre todo si p es grande comparado con n). En numerosos estudios de simulación (e.g. Zimmerman y Zimmerman, 1991; Fernández-Casal et al., 2003b) se ha observado que el sesgo en las estimaciones de los parámetros del variograma es en general menor.

En `gstat` el ajuste mediante REML se podría realizar empleando la función:

```
fit.variogram.reml(formula, locations, data, model, degree = 0, ...)
```

Sin embargo, aparentemente *el método no está bien implementado* (emplea `formula` para un ajuste OLS y después, en el código C interno, considera una tendencia polinómica en las coordenadas determinada por `degree`), actualmente solo admite datos tipo `Spatial*` del paquete `sp` y además es habitual que aparezcan problemas computacionales, por lo que no se recomendaría su uso.

```
model <- vgm(psill = 3, model = "Sph", range = 75, nugget = 0)
fit.variogram.reml(head ~ 1, data = as(aquifer_sf, "Spatial"), model = model, degree = 1)
```

```
##   model      psill range
## 1   Nug -0.7976219    0
## 2   Sph 12.5397527   75
```

Como aparece en la ayuda de esta función, es preferible usar el paquete `geoR` (ver Sección B.3.2 del apéndice; también se podría emplear el paquete `nlme`), empleando la función `geoR::likfit()` para el ajuste y posteriormente `as.vgm.variomodel()` para convertir el modelo ajustado a un objeto de `gstat`.

```
geor.models <- c(Exp = "exponential", Sph = "spherical", Cir = "circular",
                 Gau = "gaussian", Mat = "matern", Pow = "power", Nug = "nugget",
                 Lin = "linear")
pars.ini <- c(psill = 3, range = 75, nugget = 0, kappa = 0.5)
res <- geoR:::likfit(coords = st_coordinates(aquifer_sf), data = aquifer_sf$head,
                      lik.method = "REML", trend = "1st", cov.model = geor.models["Sph"],
                      ini.cov.pars = pars.ini[1:2], nugget = pars.ini[3], kappa = pars.ini[4])

## kappa not used for the spherical correlation function
## -----
## 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(res)

## Summary of the parameter estimation
## -----
## Estimation method: restricted maximum likelihood
##
## Parameters of the mean component (trend):
##   beta0  beta1  beta2
## 26.7704 -0.0701 -0.0634
##
## Parameters of the spatial component:
```

```

##      correlation function: spherical
##      (estimated) variance parameter sigmasq (partial sill) =  4.545
##      (estimated) cor. fct. parameter phi (range parameter)  = 79.16
##      anisotropy parameters:
##          (fixed) anisotropy angle = 0  ( 0 degrees )
##          (fixed) anisotropy ratio = 1
##
## Parameter of the error component:
##      (estimated) nugget =  1.191
##
## Transformation parameter:
##      (fixed) Box-Cox parameter = 1 (no transformation)
##
## Practical Range with cor=0.05 for asymptotic range: 79.16084
##
## Maximised Likelihood:
##      log.L n.params      AIC      BIC
## "-160.4"      "6"    "332.7"   "347.4"
##
## non spatial model:
##      log.L n.params      AIC      BIC
## "-174.5"      "4"    "357.1"   "366.8"
##
## Call:
## geoR::likfit(coords = st_coordinates(aquifer_sf), data = aquifer_sf$head,
##      trend = "1st", ini.cov.pars = pars.ini[1:2], nugget = pars.ini[3],
##      kappa = pars.ini[4], cov.model = geor.models["Sph"], lik.method = "REML")
as.vgm.variomodel(res)

##      model     psill     range
## 1    Nug 1.191129 0.000000
## 2    Sph 4.544502 79.16084

```

3.4 Comentarios sobre los distintos métodos

Podemos afirmar que los métodos de estimación basados en mínimos cuadrados son los utilizados con mayor frecuencia en geoestadística. Por el contrario la estimación por máxima verosimilitud ha sido objeto de debate, con numerosos comentarios en la literatura a favor (e.g. Pardo-Igúzquiza, 1998) y en contra (e.g. Ripley, 1988) de este tipo de métodos.

Una ventaja de los métodos de máxima verosimilitud es que permiten estimar de forma conjunta β y θ directamente de los datos (y no es necesario calcular estimaciones piloto del variograma). Los problemas numéricos relacionados con este tipo de estimación se pueden resolver en la práctica utilizando por ejemplo algoritmos genéticos; aunque el tiempo de computación aumenta notablemente cuando el número de datos es grande (algo que también ocurre con el método GLS). Sin embargo, uno de los principales inconvenientes normalmente achacados a estos métodos es que la hipótesis de normalidad es difícil (o más bien imposible) de chequear en la práctica a partir de una única realización parcial del proceso. Otro problema que también se debe tener en cuenta al utilizar estos métodos es su falta de robustez cuando hay valores atípicos (outliers) en los datos.

No obstante es de esperar que las estimaciones obtenidas con los métodos de máxima verosimilitud sean más eficientes cuando la distribución de los datos se aproxima a la normalidad y el modelo paramétrico está especificado correctamente (especialmente con la estimación REML); aunque no está claro si esta mejora es realmente significativa comparada con otros métodos más sencillos como el de WLS. De hecho, bajo la hipótesis de normalidad, Zimmerman y Zimmerman (1991) observaron, al comparar mediante simulación las estimaciones obtenidas utilizando distintos métodos (entre ellos ML, REML, OLS, WLS y GLS), que el método de WLS era a veces el mejor procedimiento y nunca resultaba

malo (considerando el sesgo, el error en media cuadrática y la cobertura del intervalo de predicción al 95%). Además, los métodos de mínimos cuadrados sólo utilizan la estructura asintótica de segundo orden del estimador piloto (no es necesario hacer suposiciones sobre la distribución completa de los datos), por lo que resultan ser más robustos que los de máxima verosimilitud cuando no se conoce por completo la distribución de Z (e.g. Carroll y Ruppert, 1982) y son adecuados incluso cuando la distribución de los datos no es normal (aunque en ese caso pueden no ser los óptimos). Los comentarios anteriores, además de su fácil implementación, justifican que el método WLS sea el preferible para muchos autores.

Por otra parte, si se asume un modelo paramétrico habrá que asegurarse en la práctica de que esa suposición es adecuada. Dibiasi y Bowman (2001) propusieron un método basado en la estimación no paramétrica para contrastar si el variograma teórico de un proceso estacionario es constante (i.e. si no hay dependencia espacial; ver Figura 3.3). Posteriormente Maglione y Dibiasi (2004) propusieron contrastes de hipótesis para verificar si un determinado modelo paramétrico semivariograma es apropiado (ver también Bowman y Crujeiras, 2013). Para evitar posibles problemas relacionados con la mala especificación del modelo de variograma se puede pensar en su estimación de forma no paramétrica (Capítulo XX; ver paquete `npsp`). En este caso los métodos de mínimos cuadrados serán claramente preferibles a los basados en máxima verosimilitud.

Para comparar el ajuste obtenido con distintos modelos se pueden considerar los correspondientes valores finales de la función objetivo utilizada; por ejemplo los valores WLS (o GLS) correspondientes a su ajuste al estimador piloto o los valores del NLL si se utiliza alguno de los métodos de máxima verosimilitud (en este caso también se pueden emplear criterios para la selección de modelos que tengan en cuenta el número de parámetros, como AIC -*Aikaike Information Criterion*- o BIC -*Bayesian Information Criterion*). Sin embargo en muchas ocasiones el objetivo final es la predicción, por lo que se suele utilizar la técnica de validación cruzada descrita en la Sección 4.6.

Capítulo 4

Predicción Kriging

En este capítulo se comentan brevemente los métodos más conocidos de predicción espacial denominados métodos kriging¹ (ver Sección 1.2 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 (2012, 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:

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

- 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 de predicción (*mean squared prediction error*, MSPE):

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

En este capítulo, 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.
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 $\{X_j(\cdot) : j = 0, \dots, p\}$:

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

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

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 conocido².

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 los inconvenientes que presenta este último; ver Sección 3.3.2). Adicionalmente en la Sección 4.3.2 se tratará una extensión del KU denominada *kriging residual* o *kriging con tendencia externa*.

La suposición de que el variograma (o el covariograma) sólo dependa del salto es conveniente para facilitar el modelado de la dependencia espacial, pero para la predicción espacial no es necesaria esta consideración. Por tanto en las expresiones de las ecuaciones de los distintos métodos kriging se utilizará la notación más general no estacionaria:

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

en lugar de suponer que son funciones de $\mathbf{s}_1 - \mathbf{s}_2$.

4.2 Kriging con media conocida: kriging simple

Supongamos que el proceso $Z(\cdot)$ admite una descomposición de la forma:

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

siendo $\mu(\cdot)$ la función de tendencia conocida y $\varepsilon(\cdot)$ un proceso espacial de media cero con covariograma conocido $C(\mathbf{s}_1, \mathbf{s}_2) = Cov(\varepsilon(\mathbf{s}_1), \varepsilon(\mathbf{s}_2))$ (no necesariamente estacionario, aunque en la práctica se suele suponer que $\varepsilon(\cdot)$ es un proceso estacionario de segundo orden). El predictor lineal óptimo minimiza el MSPE, que puede expresarse como:

$$\begin{aligned} E \left[(p(\mathbf{Z}, \mathbf{s}_0) - Z(\mathbf{s}_0))^2 \right] &= Var(p(\mathbf{Z}, \mathbf{s}_0) - Z(\mathbf{s}_0)) + [E(p(\mathbf{Z}, \mathbf{s}_0) - Z(\mathbf{s}_0))]^2 \\ &= Var \left(Z(\mathbf{s}_0) - \sum_{i=1}^n \lambda_i Z(\mathbf{s}_i) \right) + \left(\mu(\mathbf{s}_0) - \sum_{i=1}^n \lambda_i \mu(\mathbf{s}_i) - \lambda_0 \right)^2, \end{aligned}$$

de donde se deduce que:

$$\lambda_0 = \mu(\mathbf{s}_0) - \sum_{i=1}^n \lambda_i \mu(\mathbf{s}_i),$$

(por tanto el sesgo es nulo). Entonces el predictor es de la forma:

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

(por tanto se puede pensar que se trata de la estimación lineal homogénea de un proceso de media cero) y el MSPE es igual a:

$$\begin{aligned} E \left[(p(\mathbf{Z}, \mathbf{s}_0) - Z(\mathbf{s}_0))^2 \right] &= E \left[\left(\sum_{i=1}^n \lambda_i \varepsilon(\mathbf{s}_i) - \varepsilon(\mathbf{s}_0) \right)^2 \right] \\ &= \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j C(\mathbf{s}_i, \mathbf{s}_j) - 2 \sum_{i=1}^n \lambda_i C(\mathbf{s}_i, \mathbf{s}_0) + C(\mathbf{s}_0, \mathbf{s}_0). \end{aligned}$$

Para minimizar esta función se igualan a cero las derivadas parciales respecto a los pesos, obteniéndose las ecuaciones del kriging simple:

$$\sum_{j=1}^n \lambda_j C(\mathbf{s}_i, \mathbf{s}_j) - C(\mathbf{s}_i, \mathbf{s}_0) = 0, \quad i = 1, \dots, n,$$

²Siempre que una de las funciones explicativas sea idénticamente 1, e.g. $X_0(\cdot) \equiv 1$, en caso contrario las ecuaciones kriging sólo pueden expresarse en función del covariograma (Sección 4.3.1).

que pueden expresarse en forma matricial como:

$$\Sigma \lambda = \mathbf{c},$$

siendo $\lambda = (\lambda_1, \dots, \lambda_n)^\top$, $\mathbf{c} = (C(\mathbf{s}_1, \mathbf{s}_0), \dots, C(\mathbf{s}_n, \mathbf{s}_0))^\top$ y Σ la matriz $n \times n$ de varianzas-covarianzas de los datos (i.e. $\Sigma_{ij} = C(\mathbf{s}_i, \mathbf{s}_j)$). Combinando las expresiones para λ_0 y λ , se obtiene el predictor del kriging simple:

$$p_{KS}(\mathbf{Z}, \mathbf{s}_0) = \mu(\mathbf{s}_0) + \mathbf{c}^\top \Sigma^{-1}(\mathbf{Z} - \mu),$$

donde $\mu = (\mu(\mathbf{s}_1), \dots, \mu(\mathbf{s}_n))^\top$, y el correspondiente valor mínimo del MSPE, también denominado *varianza kriging*:

$$\sigma_{KS}^2(\mathbf{s}_0) = C(\mathbf{s}_0, \mathbf{s}_0) - \mathbf{c}^\top \Sigma^{-1} \mathbf{c}.$$

Una de las principales utilidades de la varianza kriging es la construcción de intervalos de confianza (normalmente basados en la hipótesis de normalidad).

Para que exista una única solución del sistema la matriz Σ debe ser no singular. Una condición suficiente para que esto ocurra es que el covariograma $C(\cdot, \cdot)$ sea una función definida positiva (hay que tener cuidado con la anisotropía zonal, ver Sección 3.2.2) y las posiciones de los datos sean distintas. En la práctica suele interesar la predicción en múltiples posiciones. Teniendo en cuenta que la matriz del sistema no depende de la posición de predicción³, el procedimiento recomendado sería calcular la factorización Cholesky de la matriz Σ y posteriormente emplear esta factorización para resolver el sistema en cada posición de predicción \mathbf{s}_0 .

4.3 Kriging con media desconocida: kriging universal y kriging residual

Como ya se comentó, el kriging universal se basa en el siguiente modelo:

$$Z(\mathbf{s}) = \sum_{j=0}^p X_j(\mathbf{s})\beta_j + \varepsilon(\mathbf{s}),$$

donde $\beta = (\beta_0, \dots, \beta_p)^\top \in \mathbb{R}^{p+1}$ es un vector desconocido, $\{X_j(\cdot) : j = 0, \dots, p\}$ son funciones conocidas y $\varepsilon(\cdot)$ un proceso espacial de media cero con variograma conocido $2\gamma(\mathbf{s}_1, \mathbf{s}_2) = Var(\varepsilon(\mathbf{s}_1) - \varepsilon(\mathbf{s}_2))$ (aunque en la práctica se suele suponer estacionario). Supondremos también que $X_0(\cdot) \equiv 1$, de esta forma además en el caso particular de $p = 0$, se corresponderá con el modelo del kriging ordinario (ver Sección 4.1) muy utilizado en la práctica. Utilizando una notación matricial podemos escribir:

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

siendo $\varepsilon = (\varepsilon(\mathbf{s}_1), \dots, \varepsilon(\mathbf{s}_n))^\top$ y \mathbf{X} una matriz $n \times (p+1)$ con $\mathbf{X}_{ij} = X_{j-1}(\mathbf{s}_i)$, y:

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

con $\mathbf{x}_0 = (X_0(\mathbf{s}_0), \dots, X_p(\mathbf{s}_0))^\top$.

En este caso, como un predictor lineal verifica que:

$$E \left(\sum_{i=1}^n \lambda_i Z(\mathbf{s}_i) + \lambda_0 \right) = \lambda^\top \mathbf{X}\beta + \lambda_0,$$

siendo $\lambda = (\lambda_1, \dots, \lambda_n)^\top$, una condición necesaria y suficiente para que el predictor sea uniformemente insesgado, i.e. $E(p(\mathbf{Z}, \mathbf{s}_0)) = E(Z(\mathbf{s}_0)) = \mathbf{x}_0^\top \beta$, $\forall \beta \in \mathbb{R}^{p+1}$, es que $\lambda_0 = 0$ y:

$$\lambda^\top \mathbf{X} = \mathbf{x}_0^\top. \tag{4.1}$$

³Además, tanto los pesos kriging como la varianza kriging no dependen de los datos observados, solamente de las posiciones y del covariograma (lo que por ejemplo, entre otras cosas, facilita el diseño de la configuración espacial de muestreo).

Además como $X_0(\cdot) \equiv 1$, una de estas restricciones es:

$$\sum_{i=1}^n \lambda_i = 1, \quad (4.2)$$

que es la única condición que deben verificar los pesos en el caso del kriging ordinario.

Por tanto el predictor del kriging universal será de la forma:

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

verificando (4.1) y tal que minimiza el MSPE. Entonces se trata de minimizar:

$$E \left(Z(\mathbf{s}_0) - \sum_{i=1}^n \lambda_i Z(\mathbf{s}_i) \right)^2 - 2 \sum_{j=0}^p m_j \left(\sum_{i=1}^n \lambda_i X_j(\mathbf{s}_i) - X_j(\mathbf{s}_0) \right) \quad (4.3)$$

respecto a $\{\lambda_i : i = 1, \dots, n\}$ y $\{m_j : j = 0, \dots, p\}$, multiplicadores de Lagrange que garantizan (4.1). Teniendo en cuenta que el predictor es insesgado y que los pesos verifican (4.2), entonces:

$$\begin{aligned} \left(\sum_{i=1}^n \lambda_i Z(\mathbf{s}_i) - Z(\mathbf{s}_0) \right)^2 &= \left(\sum_{i=1}^n \lambda_i \varepsilon(\mathbf{s}_i) - \varepsilon(\mathbf{s}_0) \right)^2 \\ &= -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j (\varepsilon(\mathbf{s}_i) - \varepsilon(\mathbf{s}_j))^2 + \sum_{i=1}^n \lambda_i (\varepsilon(\mathbf{s}_i) - \varepsilon(\mathbf{s}_0))^2, \end{aligned}$$

y podemos escribir (4.3) como:

$$-\sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j \gamma(\mathbf{s}_i, \mathbf{s}_j) + 2 \sum_{i=1}^n \lambda_i \gamma(\mathbf{s}_i, \mathbf{s}_0) - 2 \sum_{j=0}^p m_j \left(\sum_{i=1}^n \lambda_i X_j(\mathbf{s}_i) - X_j(\mathbf{s}_0) \right)$$

Derivando respecto a $\{\lambda_i : i = 1, \dots, n\}$ y $\{m_j : j = 0, \dots, p\}$ e igualando a cero se obtienen las $n+p+1$ ecuaciones del kriging universal que, expresadas en forma matricial, resultan ser:

$$\Gamma_U \lambda_U = \gamma_U,$$

con:

$$\Gamma_U = \begin{pmatrix} \Gamma & \mathbf{X} \\ \mathbf{X}^\top & \mathbf{0} \end{pmatrix}, \quad \lambda_U = \begin{pmatrix} \lambda \\ \mathbf{m} \end{pmatrix}, \quad \gamma_U = \begin{pmatrix} \gamma \\ \mathbf{x}_0 \end{pmatrix},$$

donde $\gamma = (\gamma(\mathbf{s}_1, \mathbf{s}_0), \dots, \gamma(\mathbf{s}_n, \mathbf{s}_0))^\top$, $\mathbf{m} = (m_0, \dots, m_p)^\top$ y Γ es una matriz $n \times n$ con $\Gamma_{ij} = \gamma(\mathbf{s}_i, \mathbf{s}_j)$. Además el MSPE mínimo, o varianza kriging:

$$\sigma_{KU}^2(\mathbf{s}_0) = 2 \sum_{i=1}^n \lambda_i \gamma(\mathbf{s}_0, \mathbf{s}_i) - \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j \gamma(\mathbf{s}_i, \mathbf{s}_j)$$

se puede obtener como:

$$\begin{aligned} \sigma_{KU}^2(\mathbf{s}_0) &= \sum_{i=1}^n \lambda_i \gamma(\mathbf{s}_0, \mathbf{s}_i) + \sum_{j=0}^p m_j X_j(\mathbf{s}_0) \\ &= \lambda_U^\top \gamma_U. \end{aligned}$$

En el caso particular del kriging ordinario ($p = 0$), la expresión de la varianza kriging resulta ser:

$$\sigma_{KO}^2(\mathbf{s}_0) = \sum_{i=1}^n \lambda_i \gamma(\mathbf{s}_0, \mathbf{s}_i) + m_0.$$

4.3.1 Ecuaciones en función del covariograma

Cuando existe el covariograma $C(\mathbf{s}_1, \mathbf{s}_2) = Cov(\varepsilon(\mathbf{s}_1), \varepsilon(\mathbf{s}_2))$ del proceso $\varepsilon(\cdot)$ y es conocido (una suposición más fuerte), podemos expresar las ecuaciones del kriging universal (o del KO) en función de $C(\cdot, \cdot)$. Además, si ninguna de las funciones explicativas es idénticamente 1, las ecuaciones del kriging universal sólo pueden expresarse en función del covariograma.

El proceso sería análogo al caso anterior, el sistema del kriging universal equivalente es:

$$\Sigma_U \lambda_U = \mathbf{c}_U,$$

donde:

$$\Sigma_U = \begin{pmatrix} \Sigma & \mathbf{X} \\ \mathbf{X}^\top & \mathbf{0} \end{pmatrix}, \quad \lambda_U = \begin{pmatrix} \lambda \\ \mathbf{m} \end{pmatrix}, \quad \mathbf{c}_U = \begin{pmatrix} \mathbf{c} \\ \mathbf{x}_0 \end{pmatrix},$$

y la varianza kriging es:

$$\begin{aligned} \sigma_{KU}^2(\mathbf{s}_0) &= C(\mathbf{s}_0, \mathbf{s}_0) - \sum_{i=1}^n \lambda_i C(\mathbf{s}_0, \mathbf{s}_i) + \sum_{j=0}^p m_j X_j(\mathbf{s}_0) \\ &= C(\mathbf{s}_0, \mathbf{s}_0) - \lambda_U \mathbf{c}_U. \end{aligned}$$

Muchos de los algoritmos utilizados para la solución de los sistema kriging están diseñados y optimizados para covariogramas (e.g. Chilès y Delfiner, 2012, p. 170). En el caso de variogramas no acotados se podrían emplear pseudo-covarianzas (ver Sección 3.3.2).

4.3.2 Kriging residual

Otra forma, que puede ser más interesante, de obtener las ecuaciones del KU es a partir del predictor del kriging simple. Suponiendo que β es conocido en el modelo del KU, el predictor del kriging simple es:

$$\begin{aligned} p_{KS}(\mathbf{Z}, \mathbf{s}_0) &= \mathbf{x}_0^\top \beta + \mathbf{c}^\top \Sigma^{-1} (\mathbf{Z} - \mathbf{X}\beta) \\ &= \mathbf{c}^\top \Sigma^{-1} \mathbf{Z} + (\mathbf{x}_0 - \mathbf{X}^\top \Sigma^{-1} \mathbf{c})^\top \beta. \end{aligned}$$

Cuando β no es conocido es lógico pensar en utilizar en su lugar su estimador lineal óptimo

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

obteniéndose el predictor:

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

Puede verse (Goldberger, 1962) que este predictor, lineal e insesgado, es óptimo (en el sentido de que minimiza el MSPE sobre todos los predictores lineales e insesgados) y por tanto coincide con el predictor del kriging universal. Además, teniendo en cuenta que el error ($p_{KS}(\mathbf{Z}, \mathbf{s}_0) - Z(\mathbf{s}_0)$) tiene covarianza nula con cualquier combinación lineal de \mathbf{Z} (ver e.g. Chilès y Delfiner, 2012, p. 161), esta relación también se extiende a la varianza kriging:

$$\sigma_{KU}^2(\mathbf{s}_0) = \sigma_{KS}^2(\mathbf{s}_0) + (\mathbf{x}_0 - \mathbf{X}^\top \Sigma^{-1} \mathbf{c})^\top (\mathbf{X}^\top \Sigma^{-1} \mathbf{X})^{-1} (\mathbf{x}_0 - \mathbf{X}^\top \Sigma^{-1} \mathbf{c}),$$

donde el segundo término cuantifica la precisión en la estimación de la media. Estas expresiones son conocidas como la relación de aditividad entre el KS y el KU.

Los resultados anteriores permiten pensar en la predicción lineal con media desconocida como un proceso de dos etapas: en la primera estimar la media desconocida, y en la segunda realizar la predicción lineal óptima con media supuestamente conocida. En el caso de una tendencia lineal obtenemos el predictor de KU, mientras que en el caso general se obtiene el denominado predictor del *kriging residual* (o *kriging con tendencia externa*).

4.4 Kriging con el paquete gstat

Los métodos kriging (KS, KO y KU) están implementados en la función `krige()` del paquete `gstat`. Normalmente la sintaxis empleada es:

```
krige(formula, locations, newdata, model, ..., beta)
```

- `formula`: fórmula que define la tendencia como un modelo lineal de la respuesta en función de las variables explicativas (para KO será de la forma `z ~ 1`).
- `locations`: objeto `sf` o `Spatial*` que contiene las observaciones espaciales (incluyendo las variables explicativas).
- `newdata`: objeto `sf`, `stars` o `Spatial*`, que contiene las posiciones de predicción (incluyendo las variables explicativas).
- `model`: modelo de semivariograma (generado con la función `vgm()` o `fit.variogram()`).
- `beta`: vector con los coeficientes de tendencia (incluida la intersección) si se supone conocida (se empleará para KS, en lugar de las estimaciones GLS empleadas en KO y KU).

Esta función, además de kriging univariante puntual con vecindario global, también implementa kriging por bloques, cokriging (predicción multivariante), predicción local y simulación condicional. Para predicción (o simulación) local se pueden establecer los siguientes parámetros adicionales (ver Sección 4.5.3):

- `maxdist`: solo se utilizarán las observaciones a una distancia de la posición de predicción menor de este valor.
- `nmax`, `nmin` (opcionales): número máximo y mínimo de observaciones más cercanas. Si el número de observaciones más cercanas dentro de la distancia `maxdist` es menor que `nmin`, se generará un valor faltante.
- `omax` (opcional): número máximo de observaciones por octante (3D) o cuadrante (2D).

Los parámetros (opcionales) para simulación condicional (ver e.g. Fernández-Casal y Cao, 2020, Sección 7.5) son:

- `nsim`: número de generaciones. Si se establece un valor distinto de cero, se emplea simulación condicional en lugar de predicción kriging.
- `indicators`: valor lógico que determina el método de simulación. Por defecto (`FALSE`) emplea simulación condicional gaussiana, en caso contrario (`TRUE`) simula una variable indicadora.

Como ejemplo consideraremos los datos del acuífero Wolfcamp con el modelo del kriging universal ajustado en la Sección 3.3.2 (en s100 se tiene un ejemplo adicional empleando KO).

```
load("datos/aquifer.RData")
library(sf)

## Linking to GEOS 3.9.1, GDAL 3.2.1, PROJ 7.2.1
aquifer$head <- aquifer$head/100 # en cientos de pies...
aquifer_sf <- st_as_sf(aquifer, coords = c("lon", "lat"), remove = FALSE, agr = "constant")
library(gstat)
vario <- variogram(head ~ lon + lat, aquifer_sf, cutoff = 150)
fit <- fit.variogram(vario, vgm(model = "Sph", nugget = NA), fit.method = 2)
```

Como se mostró en el Ejemplo 2.1, para generar la rejilla de predicción podemos utilizar la función `st_as_stars()` del paquete `stars` considerando un buffer de radio 40 en torno a las posiciones espaciales:

```
buffer <- aquifer_sf %>% st_geometry() %>% st_buffer(40)
library(stars)
```

```
## Loading required package: abind
## Registered S3 methods overwritten by 'stars':
##   method           from
##   st_bbox.SpatRaster sf
```

```
##   st_crs.SpatRaster sf
grid <- buffer %>% st_as_stars(nx = 50, ny = 50)
```

Como suponemos un modelo (no constante) para la tendencia, es necesario añadir los valores de las variables explicativas a la rejilla de predicción:

```
coord <- st_coordinates(grid)
grid$lon <- coord$x
grid$lat <- coord$y
```

Además, en este caso recortamos la rejilla para filtrar predicciones alejadas de las observaciones:

```
grid <- grid %>% st_crop(buffer)
```

Obtenemos las predicciones mediante kriging universal (Sección 4.3):

```
pred <- krige(formula = head ~ lon + lat, locations = aquifer_sf, model = fit,
newdata = grid)
```

```
## [using universal kriging]
```

Aparentemente hay un **ERROR** en **krige()** y cambia las coordenadas del objeto **stars**:

```
summary(st_coordinates(grid))
```

```
##      x           y
## Min. :-181.86   Min. :-28.03
## 1st Qu.:-100.73 1st Qu.: 33.25
## Median : -16.22 Median : 97.09
## Mean   : -16.22 Mean   : 97.09
## 3rd Qu.:  68.29 3rd Qu.:160.93
## Max.   : 149.42 Max.   :222.21
```

```
summary(st_coordinates(pred))
```

```
##      x           y
## Min. :-181.86   Min. : 53.83
## 1st Qu.:-100.73 1st Qu.:115.11
## Median : -16.22 Median :178.95
## Mean   : -16.22 Mean   :178.95
## 3rd Qu.:  68.29 3rd Qu.:242.79
## Max.   : 149.42 Max.   :304.08
```

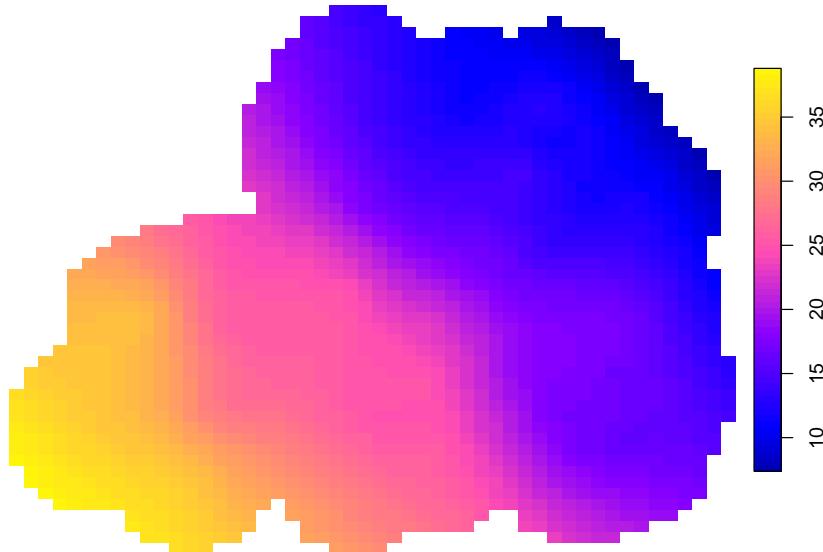
Para evitar este problema podemos añadir los resultado al objeto **grid** y emplearlo en lugar de **pred**:

```
grid$var1.pred <- pred$var1.pred
grid$var1.var <- pred$var1.var
```

Podemos representar las predicciones y las varianzas kriging empleando **plot.stars()**:

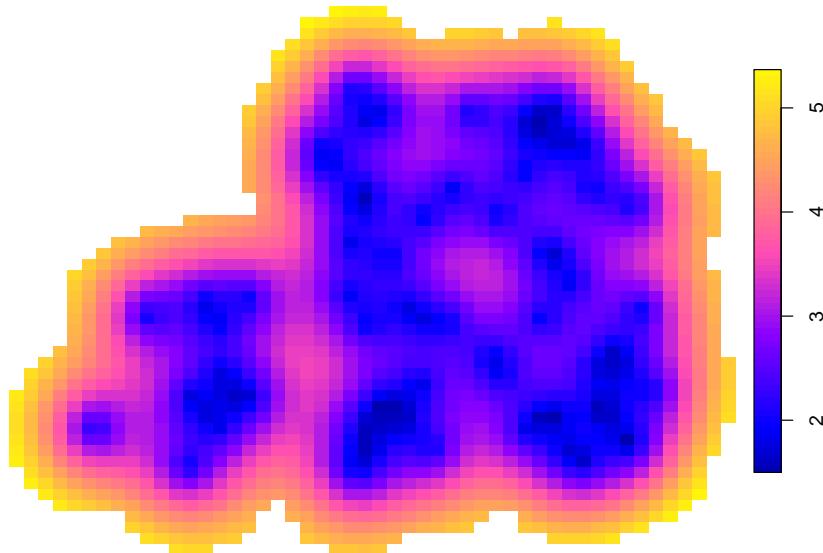
```
plot(grid["var1.pred"], breaks = "equal", col = sf.colors(64), key.pos = 4,
main = "Predicciones kriging")
```

Predicciones kriging



```
plot(grid["var1.var"], breaks = "equal", col = sf.colors(64), key.pos = 4,
     main = "Varianzas kriging")
```

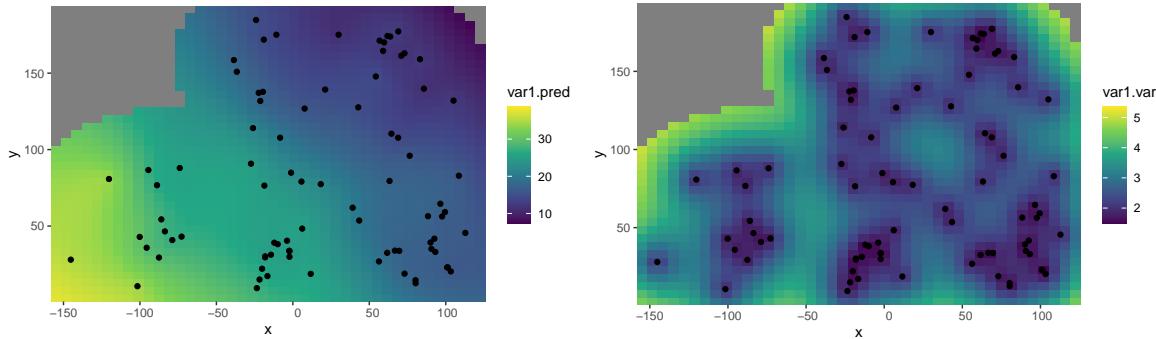
Varianzas kriging



También podríamos emplear el paquete `ggplot2`:

```
library(ggplot2)
library(gridExtra)
p1 <- ggplot() + geom_stars(data = grid, aes(fill = var1.pred, x = x, y = y)) +
  scale_fill_viridis_c() + geom_sf(data = aquifer_sf) +
  coord_sf(lims_method = "geometry_bbox")
p2 <- ggplot() + geom_stars(data = grid, aes(fill = var1.var, x = x, y = y)) +
  scale_fill_viridis_c() + geom_sf(data = aquifer_sf) +
```

```
coord_sf(lims_method = "geometry_bbox")
grid.arrange(p1, p2, ncol = 2)
```



4.5 Consideraciones acerca de los métodos kriging

Los métodos kriging descritos anteriormente permiten obtener el mejor predictor lineal insesgado (BLUP, *best linear unbiased predictor*). Como 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 e.g. Rivoirard, 1994) y del kriging trans-normal (ver Sección 4.7.2).

En estos métodos se supone que el variograma (covariograma) es conocido, sin embargo en la práctica en realidad el variograma es estimado (kriging estimado). Puede verse (Yakowitz y Szidarovszky, 1985) que, bajo condiciones muy generales, el predictor kriging con variograma estimado converge al valor correcto si la densidad de datos tiende a infinito, incluso cuando la estimación del variograma no es muy buena. Además, Stein (1988) probó que para una predicción asintóticamente (de relleno) eficiente lo que se necesita generalmente es capturar la conducta del variograma cerca del origen. Por tanto, en cuanto a las predicciones, el factor más importante es que la aproximación al variograma verdadero cerca del origen no sea muy mala.

Al contrario que en el caso del predictor, la estimación del variograma afecta directamente a la varianza kriging y en general no es un estimador consistente del error en media cuadrática del predictor con variograma estimado. En general, por ejemplo si el proceso es normal, es de esperar que subestime la verdadera varianza de predicción y por tanto debería incrementarse para que tenga en cuenta el efecto de la estimación del variograma. Para más detalles sobre este problema, ver por ejemplo Cressie (1993, Sección 5.3), Christensen (1991, Sección 6.5), Stein (1999, Sección 6.8) o Chilès y Delfiner (2012, Sección 3.4.3). En la Sección 4.6 se sugiere una posible corrección de la varianza kriging a partir del método de validación cruzada.

4.5.1 Kriging como interpolador

Las ecuaciones kriging tienen en cuenta varios aspectos del problema de interpolación⁴:

- La configuración espacial de los datos, a través de las matrices Σ (o Γ), donde el covariograma (o el variograma) actúa como una “distancia estadística” entre las observaciones y de forma que se tiene en cuenta la información redundante presente en los datos.
- La situación de la posición de predicción respecto a los datos, a través de \mathbf{c} (o γ).
- La presencia de una función determinística de tendencia.

⁴Para un tratamiento más detallado ver por ejemplo Chilès y Delfiner (2012), secciones 3.3.2 y 3.4.2.

Adicionalmente también tienen en cuenta propiedades estadísticas del proceso $Z(\cdot)$, a través del variograma o el covariograma (que como se comentó en la Sección 1.3, entre otras cosas, determina las propiedades de continuidad del proceso). Esta es la principal diferencia con otros métodos de interpolación que no tienen en cuenta la estructura de segundo orden del proceso.

Una propiedad importante de los predictores kriging es que son interpoladores exactos (suponiendo que no hay error de medida⁵), en el sentido de que $p(\mathbf{Z}, \mathbf{s}_0) = Z(\mathbf{s}_0)$ cuando $\mathbf{s}_0 = \mathbf{s}_i$ (la solución de los sistemas es $\lambda_i = 1$ y $\lambda_j = 0$, $\forall j \neq i$), y naturalmente en ese caso la estimación de la varianza es 0. Además, por lo general no son continuos en las posiciones de los datos, para que lo sean el efecto nugget debe ser nulo.

Otra característica de la interpolación kriging (y que no aparece en otros métodos como los que asignan pesos inversamente proporcionales a la distancia) es el denominado *efecto pantalla*. En general se observa (ver comentarios en la siguiente sección) que la influencia de un valor es menor si está oculto detrás de otro valor (e.g. Journel y Huijbregts, 1978, p. 346). Esto produce, por ejemplo en el caso del KO (incluso con un variograma isotrópico), que puntos situados a la misma distancia de la posición de predicción puedan tener distintos pesos y que los datos cercanos no apantallados reciban los mayores pesos, reduciéndose considerablemente (llegando a ser negativos) los pesos de los datos que quedan ocultos⁶.

La aparición de pesos negativos (o mayores que 1) en el KO como consecuencia del efecto pantalla, puede provocar (incluso suponiendo media constante) que el predictor kriging no esté necesariamente comprendido entre el valor mínimo y máximo de los datos. Esto que en principio puede ser una propiedad muy interesante puede conducir a resultados extraños en ciertas ocasiones, como por ejemplo dar lugar a predicciones negativas en casos en los que la variable considerada es necesariamente positiva. Para solucionar estos problemas se han propuesto numerosas alternativas, entre ellas la inclusión de restricciones adicionales sobre los pesos de forma que sean positivos (e.g. Chilès y Delfiner, 2012, Sección 3.9.1) (lo cual puede dar lugar a un incremento considerable del MSPE de predicción) o sobre el predictor (ver p.e. Goovaerts, 1997, Sección 7.4.2). Otra alternativa que puede ser preferible es la transformación del proceso $Z(\cdot)$ a otra escala (de forma que se aproxime a la normalidad), realizar la predicción kriging del proceso transformado y volver a la escala original (pero asegurándose de que al hacer la transformación inversa el resultado tenga las propiedades de optimalidad deseadas); más detalles sobre este procedimiento se tienen en la Sección 4.7.2.

4.5.2 Efecto del variograma (covariograma) en el kriging

El variograma (o el covariograma) tiene un efecto determinante en la predicción espacial. Como ejemplo, a continuación se incluyen algunas observaciones acerca de la influencia en el kriging de las tres principales características de un variograma estacionario (normalmente tratadas como parámetros) definidas en la Sección 1.3.

Rango

Supongamos que la posición de predicción \mathbf{s}_0 está a una distancia mayor que el rango de las posiciones de los datos $\{\mathbf{s}_1, \dots, \mathbf{s}_n\}$ (i.e. la posición de predicción está fuera de la zona de influencia de los datos), entonces $\mathbf{c} = \mathbf{0}$ en las ecuaciones kriging, obteniéndose que:

$$p_{KS}(\mathbf{Z}, \mathbf{s}_0) = \mu(\mathbf{s}_0), \quad p_{KU}(\mathbf{Z}, \mathbf{s}_0) = \mathbf{x}_0^\top \hat{\beta}_{gls},$$

por lo tanto la predicción kriging se reduce a la media⁷ (estimada en el caso del KU).

Nugget y umbral

Resulta claro que las estimaciones obtenidas de la varianza de los predictores kriging dependen en gran medida de estos parámetros. Es importante destacar que la escala del variograma (o del covariograma)

⁵En caso contrario interesaría predecir el proceso libre de ruido y habría que modificar ligeramente las ecuaciones kriging. Para más detalles, ver por ejemplo Cressie (1993, pp. 128-130) o Chilès y Delfiner (2012, Sección 3.7.1)

⁶En la literatura se muestran numerosos ejemplos sobre el comportamiento de los pesos kriging en distintos escenarios; una colección bastante completa se tiene en Wackernagel (1998, Capítulo 13).

⁷Puede verse fácilmente que la ecuaciones del kriging universal con $\mathbf{c} = \mathbf{0}$, son las obtenidas en el denominado método kriging de estimación de la tendencia (e.g. Wackernagel, 1998, pp. 212-213; Chilès y Delfiner, 2012, Sección 3.4.5).

no influye en las predicciones obtenidas, solamente en la varianza kriging. Si se multiplica el variograma (o el covariograma) por una constante, las ecuaciones de los predictores kriging quedan invariantes y consecuentemente los pesos kriging no cambian, aunque la varianza kriging resulta multiplicada por esa constante. Para estudiar su influencia en la predicción resulta de utilidad la proporción del efecto nugget en el umbral total c_0/σ^2 (que como ya se comentó al final de la Sección 1.3, 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. el proceso $Z(\cdot)$ es ruido blanco) entonces $\Sigma = \sigma^2 \mathbf{I}_n$ y $\mathbf{c} = \mathbf{0}$ (suponiendo que $\mathbf{s}_0 \neq \mathbf{s}_i, \forall i$), y los predictores kriging se reducen a la estimación OLS de la tendencia:

$$p_{KU}(\mathbf{Z}, \mathbf{s}_0) = \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Z} = \mathbf{x}_0^\top \hat{\boldsymbol{\beta}}_{ols}.$$

En el caso del KO se obtiene la media muestral:

$$p_{KO}(\mathbf{Z}, \mathbf{s}_0) = \bar{Z} = \frac{1}{n} \sum_{i=1}^n Z(\mathbf{s}_i),$$

predictor bien conocido que asigna igual peso a todos los datos (por tanto el efecto pantalla es nulo). Además, teniendo en cuenta los resultados de la Sección 4.3.2, como $\sigma_{KS}^2(\mathbf{s}_0) = \sigma^2$ (caso más desfavorable), entonces:

$$\sigma_{KO}^2(\mathbf{s}_0) = \sigma^2 + \frac{\sigma^2}{n},$$

es decir, la varianza del KO para el caso de procesos incorrelados es igual a la varianza del proceso más la varianza de la media muestral. En general (al menos en KS y KO), se puede ver que al aumentar el porcentaje de efecto nugget en el umbral total disminuye el efecto pantalla y aumenta la varianza kriging (ver e.g. Isaaks y Srivastava, 1989, pp. 305-306). Hay que tener en cuenta que cuando la media no es constante (e.g. KU), puede ocurrir incluso lo contrario del efecto pantalla, y observaciones alejadas pueden tener una gran influencia en la estimación (como es bien conocido en la estimación de la tendencia).

Teniendo en cuenta los comentarios anteriores, podemos afirmar que todos los parámetros (características) del variograma influyen en el kriging (aunque quizás el rango es el que tiene un menor efecto, ya que pequeñas variaciones en este parámetro producen resultados casi idénticos). Estas observaciones no contradicen el resultado de que asintóticamente lo importante es capturar el comportamiento del variograma cerca del origen (Stein, 1988). Es difícil determinar cuando los datos están suficientemente cerca como para tener sólo en cuenta el efecto nugget y la pendiente del variograma cerca del origen.

4.5.3 Elección del vecindario

Una práctica habitual en geoestadística, en lugar de considerar todas las observaciones disponibles, es incluir en las ecuaciones kriging únicamente los $n(\mathbf{s}_0)$ datos más “próximos” a la posición de predicción \mathbf{s}_0 . Esto puede justificarse por varias razones:

- Utilizar todos los datos puede dar lugar a un sistema de difícil solución debido a problemas numéricos. Por ejemplo, entre otros, el tiempo de computación (aproximadamente proporcional a $n(\mathbf{s}_0)^3$) aumenta drásticamente al aumentar el numero de datos.
- Las estimaciones del variograma son normalmente eficientes (o incluso el propio modelo geostadístico válido) únicamente en pequeñas distancias.
- El uso de vecindarios locales permite la relajación de las hipótesis del modelo (como la estacionariedad intrínseca en el caso del KO) o su simplificación (por ejemplo, en el caso del KU se puede suponer que localmente la estructura de la tendencia es más simple o incluso constante y utilizar en su lugar KO local).
- En muchos casos los datos cercanos apantallan a los más alejados reduciendo su influencia (aunque no siempre de forma significativa; ver observaciones siguientes).

La selección “optima” de un vecindario local resulta no obstante un problema algo complejo. Por ejemplo, se acostumbra a pensar que el rango del variograma permite determinar por sí solo un criterio de vecindad, como incluir en las ecuaciones sólo los datos que estén dentro del rango de \mathbf{s}_0 , sin embargo esto puede no ser adecuado en muchos casos (aunque en la práctica el valor de este parámetro puede ser de gran utilidad como referencia inicial). Teniendo en cuentas las observaciones realizadas en secciones anteriores, cuando aumenta la proporción de efecto nugget disminuye el efecto pantalla,

el predictor kriging se reduce a la media y observaciones a más distancia que el rango de la posición de predicción contribuyen (a veces de forma significativa) a la estimación de la tendencia. Hay que tener en cuenta que los pesos kriging dependen de la configuración espacial de todas las observaciones y observaciones fuera del rango pueden tener influencia en la predicción a través de su correlación con observaciones dentro del rango (esto es conocido en la literatura como efecto *relay*, que podríamos traducir por efecto transmisión).

Se han propuesto algunos criterios para la selección de un vecindario óptimo (e.g. Cressie, 1993, pp. 176-177), que son de utilidad cuando los datos están regularmente espaciados y el vecindario se puede fijar de antemano. Por ejemplo, se pueden ir incluyendo observaciones en el vecindario hasta que no se produzca una disminución “significativa” en la estimación de la varianza kriging.

En la práctica, la densidad de datos y su configuración espacial en torno a las posiciones de predicción pueden ser muy irregulares. Teniendo en cuenta esto se han desarrollado distintos algoritmos, algunos bastante sofisticados, para la selección de vecindarios (ver e.g. Isaaks y Srivastava, 1989, Capítulo 14; Deutsch y Journel, 1992, secciones II.4 y IV.6). Para la selección de los datos típicamente se fija un radio máximo de búsqueda y únicamente se consideran los datos dentro de una circunferencia (esfera) centrada en la posición de predicción. En el caso de anisotropía (Sección 2.2.2), se considera una elipse (elipsoide) con el radio mayor orientado en la dirección de máxima variación. Además suele interesar disponer de observaciones en todas direcciones, por lo que se divide la zona de búsqueda en sectores angulares (por ejemplo cuadrantes en el caso bidimensional o octantes en el caso tridimensional) y se selecciona un número mínimo de datos en todos o en la mayoría de esos sectores (esto evita también que clusters de datos tengan una excesiva influencia sobre predicciones en su entorno). Si se sospecha de la presencia de una tendencia en los datos (KU), puede ser deseable la inclusión de observaciones más alejadas de la posición de predicción para poder estimarla de forma más eficiente.

Utilizando un algoritmo de búsqueda que tenga en cuenta todas o alguna de las consideraciones anteriores, típicamente se selecciona un número pequeño de datos como vecindario (e.g. entre 10 y 20 observaciones) para cada posición de predicción. Sin embargo esto puede causar que las superficies de predicción presenten discontinuidades (especialmente cuando los datos están regularmente espaciados y se utiliza búsqueda por octantes). Una aproximación distinta sería la selección un único vecindario más grande (e.g. de 20 a 40 observaciones) para pequeños conjuntos de posiciones de predicción, de esta forma en condiciones normales los vecindarios correspondientes a conjuntos de predicción próximos se solapan y es de esperar que no aparezcan discontinuidades. Además el incremento en tiempo de computación debido a la inclusión de un número mayor de observaciones se compensa por el hecho de que sólo es necesario factorizar una vez la matriz sistema kriging para obtener las predicciones en el conjunto considerado⁸.

Otra forma de proceder que puede ser de interés en la práctica, es usar el semivariograma como distancia en lugar de la tradicional distancia euclídea; de esta forma los datos son seleccionados preferentemente en la dirección de máxima continuidad (y se evita tener que considerar elipsoides en el caso de anisotropía). Adicionalmente, cuando el número de datos es grande y sus posiciones irregulares, es recomendable utilizar alguna técnica de búsqueda, como el denominado *kd-tree* (e.g. ver paquete FNN), de forma que se puedan determinar eficientemente las observaciones más próximas a la posición de predicción s_0 (en lugar de comprobar todas las posiciones).

Independientemente del algoritmo de búsqueda que se vaya a utilizar, puede ser recomendable realizar antes algunas pruebas utilizando por ejemplo la técnica de validación cruzada descrita a continuación.

4.6 Validación cruzada del modelo ajustado

El método de validación cruzada es la técnica normalmente utilizada en geoestadística para diagnosticar si un modelo (de tendencia y variograma) describe adecuadamente la variabilidad espacial de los datos (ver e.g. Cressie, 1993, Sección 2.6.4). Si se asume que la tendencia es constante, permitiría verificar si el modelo de variograma describe adecuadamente la dependencia espacial de los

⁸Por ejemplo, si se pretenden obtener predicciones en una rejilla bidimensional, el tiempo de computación considerando un vecindario distinto con 16 observaciones para cada nodo resulta similar a considerar un vecindario con 25 (o incluso más) observaciones para grupos de 4 nodos (dos por fila y columna).

datos. Aunque también es utilizada para otros fines (ver e.g. Isaaks y Srivastava, 1989, Capítulo 15), entre ellos: comparar distintas hipótesis sobre el modelo geoestadístico (tipo de modelo, vecindarios, etc.), detectar observaciones atípicas o incluso para la estimación de los parámetros del variograma. La idea básica es eliminar una parte de los datos y utilizar el resto de los datos para predecir los datos eliminados, entonces el error de predicción puede deducirse del valor que se predice menos el observado; repitiendo esto sobre varios conjuntos de datos se tiene una idea sobre la variabilidad del error de predicción. En su versión más simple, validación cruzada dejando uno fuera (*Leave-one-out cross-validation*, LOOCV), para cada observación de la muestra se obtiene una predicción empleando el resto de observaciones (para más detalles, ver e.g. Fernández-Casal et al., 2021, Sección 1.3.3). En el caso de datos geoestadísticos no sólo interesa analizar las predicciones, en general son también de interés las estimaciones del error cuadrático de predicción (varianza kriging).

Supongamos que $\hat{Z}_{-j}(\mathbf{s}_j)$ es un predictor de $Z(\mathbf{s}_j)$ obtenido, utilizando alguno de los métodos de predicción espacial, a partir de $\{Z(\mathbf{s}_i) : i \neq j\}$ y el variograma ajustado $2\gamma(\cdot, \hat{\theta})$ (calculado utilizando todos los datos), y que $\sigma_{-j}^2(\mathbf{s}_j)$ es el correspondiente error en media cuadrática de predicción. Hay varias formas de medir la aproximación de las predicciones a los verdaderos valores, por ejemplo:

- La media de los errores tipificados (*dimensionless mean error*)

$$\text{DME} = \frac{1}{n} \sum_{j=1}^n (\hat{Z}_{-j}(\mathbf{s}_j) - Z(\mathbf{s}_j)) / \sigma_{-j}(\mathbf{s}_j)$$

debería ser próxima a cero. Este no es un criterio muy adecuado (sobre todo en el caso del KO) ya que los predictores kriging son insesgados independientemente del modelo de variograma utilizado (ver e.g. Yakowitz y Szidarovski, 1985).

- El error cuadrático medio adimensional (*dimensionless mean squared error*):

$$\text{DMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n ((\hat{Z}_{-j}(\mathbf{s}_j) - Z(\mathbf{s}_j)) / \sigma_{-j}(\mathbf{s}_j))^2}$$

debería ser próximo a uno. El valor de este estadístico puede interpretarse como una medida de la concordancia entre las varianzas kriging y las varianzas observadas. Teniendo en cuenta que si reescalamos el variograma multiplicándolo por una constante, las predicciones con el variograma reescalado son idénticas y las varianzas kriging serán las mismas multiplicadas por esa constante. Podemos pensar en “corregir” las varianzas kriging obtenidas con un modelo de variograma estimado de forma que el DMSE sea igual a 1, multiplicándolas por DMSE^2 .

- El error cuadrático medio (*mean squared error*):

$$\text{MSE} = \frac{1}{n} \sum_{j=1}^n (\hat{Z}_{-j}(\mathbf{s}_j) - Z(\mathbf{s}_j))^2$$

debería ser pequeño. El principal problema de este estadístico es que asigna igual peso a todos los datos y no tiene en cuenta las posiciones espaciales. Por lo general los errores son mayores en los puntos más alejados del resto de los datos (observaciones exteriores) y pueden tener un efecto dominante en la media global. Se podría pensar en calcular una media ponderada con pesos inversamente proporcionales a la varianza kriging o a alguna medida de la distancia de una posición al resto de los datos.

- Diversos criterios gráficos pueden ser también de interés como herramientas de diagnóstico, como un gráfico de tallo-hojas de los residuos tipificados o gráficos de normalidad.

Después de la validación cruzada del variograma, si esta resultó ser satisfactoria, se puede confiar en que la predicción basada en el modelo ajustado es aproximadamente óptima y que las estimaciones del error en media cuadrática de predicción son bastante buenas (i.e. el modelo ajustado no es muy incorrecto).

Uno de los principales problemas de esta técnica es el elevado coste computacional (Cressie, 1993, p. 104), sin embargo, se han desarrollado métodos (normalmente ignorados) que permiten realizar

la validación cruzada de un modelo de variograma de forma rápida y sencilla (ver Fernández-Casal, 2003c, Sección 4.4). Alternativamente se podría emplear validación cruzada en k grupos (*k-fold cross-validation*), considerando típicamente 10 o 5 grupos⁹.

La validación cruzada en `gstat` está implementada en la función `krige.cv()`. La sintaxis de esta función es casi idéntica a la de la función `krige()` (Sección 4.4, salvo que incluye un argumento `nfold` en lugar de `newdata`:

```
krige.cv(formula, locations, model, nfold = nrow(data), ...)
```

El resultado (un `data.frame` o un objeto del mismo tipo que `locations`), además de las predicciones y varianzas kriging de validación cruzada (en las posiciones de observación), contiene los componentes `observed` (valor observado), `residual` (residuos), `zscore` (residuos divididos por el error estándar kriging) y `fold` (grupo de validación cruzada).

Como ejemplo continuaremos con los datos del acuífero Wolfcamp (en `s100` se tiene un ejemplo adicional empleando KO). Como ya se comentó, la función `krige.cv()` emplea LOOCV por defecto y puede requerir de mucho tiempo de computación (no implementa eficientemente esta técnica):

```
system.time(cv <- krige.cv(formula = head ~ lon + lat, locations = aquifer_sf,
                             model = fit))

##      user  system elapsed
##     0.58    0.00    0.57

str(cv)

## Classes 'sf' and 'data.frame':   85 obs. of  7 variables:
## $ var1.pred: num  15 23.5 22.9 24.6 17 ...
## $ var1.var : num  3.08 2.85 2.32 2.81 2.05 ...
## $ observed : num  14.6 25.5 21.6 24.6 17.6 ...
## $ residual : num  -0.3357 1.9962 -1.3101 -0.0792 0.5478 ...
## $ zscore   : num  -0.1914 1.1821 -0.8608 -0.0472 0.3829 ...
## $ fold     : int  1 2 3 4 5 6 7 8 9 10 ...
## $ geometry :sfc_POINT of length 85; first list element: 'XY' num  42.8 127.6
## - attr(*, "sf_column")= chr "geometry"
## - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",... NA NA NA NA NA
## ..- attr(*, "names")= chr [1:6] "var1.pred" "var1.var" "observed" "residual" ...
```

Si el número de observaciones es grande puede ser preferible emplear k-fold CV (y como la partición en grupos es aleatoria se recomendaría fijar previamente la semilla de aleatorización):

```
set.seed(1)
system.time(cv <- krige.cv(formula = head ~ lon + lat, locations = aquifer_sf,
                           model = fit, nfold = 10))

##      user  system elapsed
##     0.06    0.00    0.07
```

Como ya se comentó, podemos considerar distintos estadísticos, por ejemplo los implementados en la siguiente función (los tres últimos tienen en cuenta la estimación de la varianza kriging):

```
summary_cv <- function(cv.data, na.rm = FALSE,
                        tol = sqrt(.Machine$double.eps)) {
  err <- cv.data$residual      # Errores
  obs <- cv.data$observed
  z <- cv.data$zscore
  w <- 1/pmax(cv.data$var1.var, tol) # Ponderación según varianza kriging
  if(na.rm) {
```

⁹LOOCV sería un caso particular considerando un número de grupos igual al número de observaciones. La partición en k-fold CV se suele realizar al azar. Hay que tener en cuenta la aleatoriedad al emplear k-fold CV, algo que no ocurre con LOOCV.

```

is.a <- !is.na(err)
err <- err[is.a]
obs <- obs[is.a]
z <- z[is.a]
w <- w[is.a]
}
perr <- 100*err/pmax(obs, tol) # Errores porcentuales
return(c(
  # Medidas de error tradicionales
  me = mean(err),           # Error medio
  rmse = sqrt(mean(err^2)), # Raíz del error cuadrático medio
  mae = mean(abs(err)),     # Error absoluto medio
  mpe = mean(perr),         # Error porcentual medio
  mape = mean(abs(perr)),   # Error porcentual absoluto medio
  r.squared = 1 - sum(err^2)/sum((obs - mean(obs))^2), # Pseudo R-cuadrado
  # Medidas de error que tienen en cuenta la varianza kriging
  dme = mean(z),            # Error estandarizado medio
  dmse = sqrt(mean(z^2)),   # Error cuadrático medio adimensional
  rwmse = sqrt(weighted.mean(err^2, w)) # Raíz del ECM ponderado
))
}

summary_cv(cv)

```

```

##          me        rmse        mae        mpe        mape      r.squared
## 0.058039856 1.788446500 1.407874022 -0.615720059 7.852363328 0.913398424
##          dme        dmse       rwmse
## 0.001337332 1.118978878 1.665958815

```

Estas medidas podrían emplearse para seleccionar modelos (de tendencia y variograma), y también para ayudar a establecer los parámetros del vecindario para kriging local.

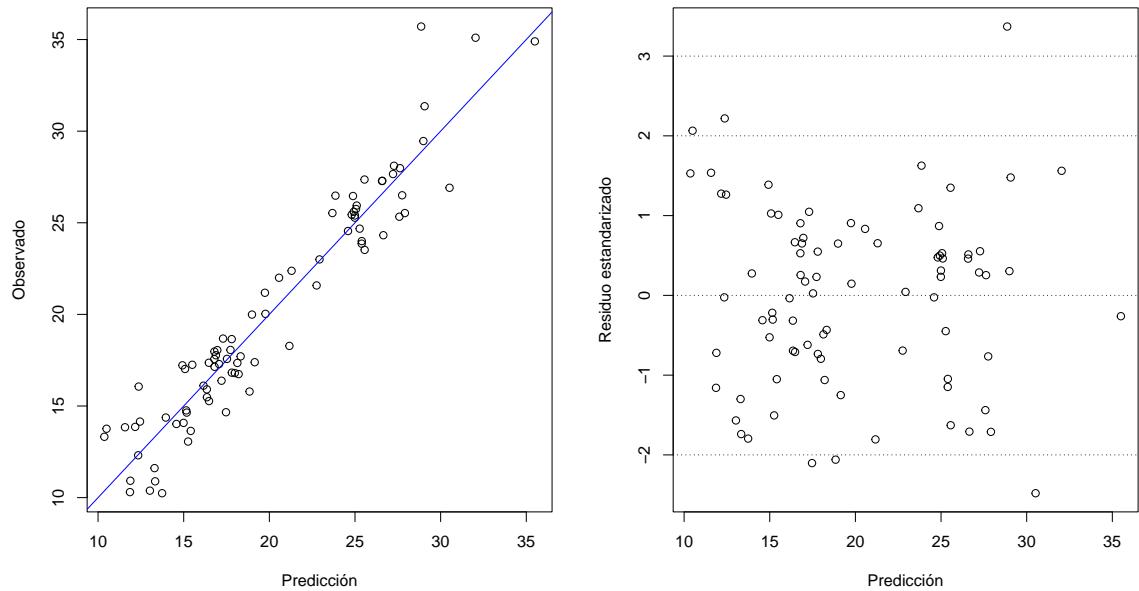
Para detectar datos atípicos, o problemas con el modelo, podemos generar distintos gráficos. Por ejemplo, gráficos de dispersión de valores observados o residuos estandarizados frente a predicciones:

```

old_par <- par(mfrow = c(1, 2))
plot(observed ~ var1.pred, data = cv, xlab = "Predicción", ylab = "Observado")
abline(a = 0, b = 1, col = "blue")

plot(zscore ~ var1.pred, data = cv, xlab = "Predicción", ylab = "Residuo estandarizado")
abline(h = c(-3, -2, 0, 2, 3), lty = 3)

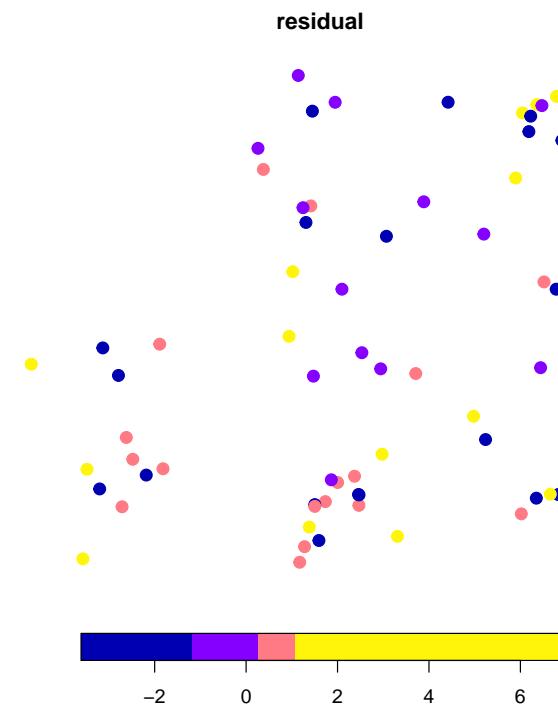
```



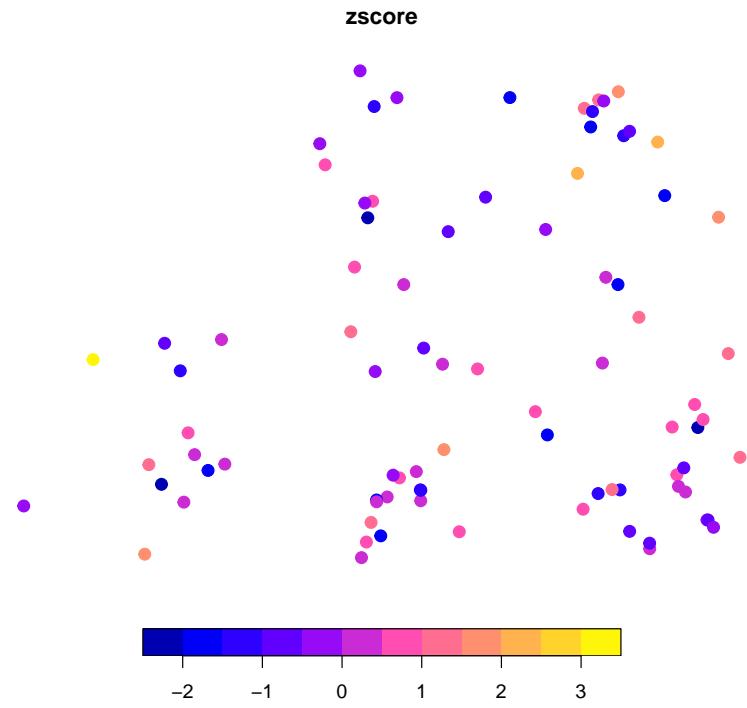
```
par(old_par)
```

Gráficos con la distribución espacial de los residuos:

```
plot(cv["residual"], pch = 20, cex = 2, breaks = "quantile", nbreaks = 4)
```

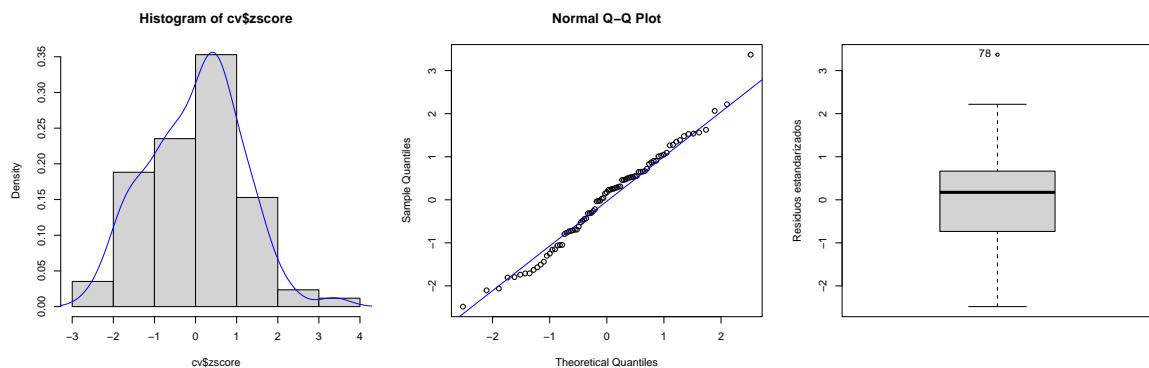


```
plot(cv["zscore"], pch = 20, cex = 2)
```



Además de los gráficos estándar para analizar la distribución de los residuos estandarizados o detectar atípicos:

```
# Histograma
old_par <- par(mfrow = c(1, 3))
hist(cv$zscore, freq = FALSE)
lines(density(cv$zscore), col = "blue")
# Gráfico de normalidad
qqnorm(cv$zscore)
qqline(cv$zscore, col = "blue")
# Boxplot
car:::Boxplot(cv$zscore, ylab = "Residuos estandarizados")
```



```
## [1] 78
par(old_par)
```

4.7 Otros métodos kriging

4.7.1 Block kriging

Aunque en los métodos kriging descritos anteriormente se asumía que el soporte era puntual, serían igualmente válidos para el caso de distintos soportes. Simplemente habría que sustituir las semivarianzas (o covarianzas) puntuales por las del proceso agregado $\text{Var}(Z(B_1) - Z(B_2))$ (ver Sección 1.3.3).

La función `krige()` del paquete `gtsat` empleará este método de forma automática cuando la geometría del argumento `newdata` sean polígonos (incluyendo datos raster).

4.7.2 Kriging trans-normal

Como se comentó en la Sección 4.5, en el caso de normalidad el predictor óptimo $E(Z(\mathbf{s}_0)|\mathbf{Z})$ de $Z(\mathbf{s}_0)$ es lineal y coincide con los predictores kriging. Pero si el proceso no es normal este predictor puede ser altamente no lineal, en esos casos la transformación del proceso $Z(\cdot)$ a otra escala puede producir que se aproxime a la normalidad. De esta forma se puede pensar en realizar la predicción lineal en la escala transformada (donde también puede ser más eficiente realizar el modelado) y posteriormente hacer la transformación inversa (pero asegurándose de que el resultado tenga las propiedades de optimalidad deseadas). En esta Sección simplemente se muestran algunos resultados sobre este método, para un tratamiento más detallado ver por ejemplo Cressie (1993, Sección 3.2.2) o Chilès y Delfiner (2012, Sección 3.4.10).

Una de las transformaciones más utilizadas en geoestadística es la transformación logarítmica, asumiendo que el proceso $Z(\cdot)$ sigue una distribución lognormal. Un proceso aleatorio lognormal es un proceso $\{Z(\mathbf{s}) : \mathbf{s} \in D\}$ (que toma valores positivos) tal que:

$$Y(\mathbf{s}) = \log(Z(\mathbf{s})); \quad \mathbf{s} \in D,$$

es un proceso normal.

El *kriging simple lognormal* (KSL) se basa en la suposición adicional de que el proceso $Y(\cdot)$ verifica las hipótesis del kriging simple (media y covariograma conocidos). En ese caso, utilizando el método del KS, a partir de $\mathbf{Y} = (Y(\mathbf{s}_1), \dots, Y(\mathbf{s}_n))^{\top}$ podemos obtener el predictor $p_{KS}(\mathbf{Y}, \mathbf{s}_0)$ de $Y(\mathbf{s}_0)$ y la correspondiente varianza kriging $\sigma_{KS}^2(\mathbf{s}_0)$. Si transformamos de nuevo este valor a la escala de $Z(\cdot)$, obtenemos $\exp(p_{KS}(\mathbf{Y}, \mathbf{s}_0))$ que no es un predictor insesgado de $Z(\mathbf{s}_0)$ (es un predictor insesgado en mediana). El predictor óptimo de $Z(\mathbf{s}_0)$ resulta ser:

$$p_{KSL}(\mathbf{Z}, \mathbf{s}_0) = \exp\left(p_{KS}(\mathbf{Y}, \mathbf{s}_0) + \frac{1}{2}\sigma_{KS}^2(\mathbf{s}_0)\right),$$

y la correspondiente varianza kriging:

$$\sigma_{KSL}^2(\mathbf{s}_0) = p_{KS}(\mathbf{Y}, \mathbf{s}_0)^2 (\exp(\sigma_{KS}^2(\mathbf{s}_0)) - 1).$$

En el caso de media no conocida, i.e. suponiendo que el proceso $Y(\cdot)$ verifica las hipótesis del KU (kriging universal lognormal, KUL), se complica aún más el problema. No basta con sustituir la media teórica en por su predictor óptimo ya que se obtendría un predictor sesgado. Si se hace una corrección para obtener un predictor insesgado de $Z(\mathbf{s}_0)$, el resultado sería:

$$p_{KUL}(\mathbf{Z}, \mathbf{s}_0) = \exp\left(p_{KU}(\mathbf{Y}, \mathbf{s}_0) + \frac{1}{2}\sigma_{KU}^2(\mathbf{s}_0) - \mathbf{m}^{\top} \mathbf{x}_0\right),$$

utilizando la notación de la Sección 4.3 (y suponiendo también que una de las funciones explicativas es idénticamente 1). Hay que destacar que el predictor no es un predictor óptimo en el sentido de que minimice el MSPE (este predictor minimiza $E(\log p(\mathbf{Z}, \mathbf{s}_0) - Y(\mathbf{s}_0))^2$, sujeto a las correspondientes restricciones de insesgadez y forma del predictor).

La varianza kriging tiene una expresión considerablemente más complicada que en el caso de media conocida (ver Cressie, 1993, p. 136; para el caso de media constante). Sin embargo, si el objetivo es

la construcción de intervalos de confianza, se pueden transformar directamente de la escala $Y(\cdot)$. Por ejemplo:

$$(\exp(p_K(\mathbf{Y}, \mathbf{s}_0) - 1.96\sigma_K^2(\mathbf{s}_0)), \exp(p_K(\mathbf{Y}, \mathbf{s}_0) + 1.96\sigma_K^2(\mathbf{s}_0))),$$

es un intervalo de confianza al 95% para $Z(\mathbf{s}_0)$.

La aproximación anterior puede generalizarse para una transformación cualquiera:

$$Z(\mathbf{s}) = \phi(Y(\mathbf{s})); \quad \mathbf{s} \in D,$$

siendo $Y(\cdot)$ un proceso normal y $\phi(\cdot)$ una función medible dos veces diferenciable. En general no se dispone de expresiones exactas como en el caso del kriging lognormal, aunque a partir de un predictor kriging $p_K(\mathbf{Y}, \mathbf{s}_0)$ de $Y(\mathbf{s}_0)$ se pueden obtener un predictor aproximadamente insesgado de $Z(\mathbf{s}_0)$ teniendo en cuenta que:

$$\begin{aligned} \phi(Y(\mathbf{s}_0)) &\simeq \phi(p_K(\mathbf{Y}, \mathbf{s}_0)) + (Y(\mathbf{s}_0) - p_K(\mathbf{Y}, \mathbf{s}_0))\phi'(p_K(\mathbf{Y}, \mathbf{s}_0)) \\ &+ \frac{1}{2}(Y(\mathbf{s}_0) - p_K(\mathbf{Y}, \mathbf{s}_0))^2\phi''(p_K(\mathbf{Y}, \mathbf{s}_0)), \end{aligned}$$

si el error kriging $p_K(\mathbf{Y}, \mathbf{s}_0) - Y(\mathbf{s}_0)$ es pequeño. A partir de esto, si $\sigma_K^2(\mathbf{s}_0)$ es la correspondiente varianza kriging, se obtiene el predictor (aproximadamente insesgado) del *kriging trans-normal* (KT):

$$p_{KT}(\mathbf{Z}, \mathbf{s}_0) = \phi(p_K(\mathbf{Y}, \mathbf{s}_0)) + \frac{1}{2}\sigma_K^2(\mathbf{s}_0)\phi''(p_K(\mathbf{Y}, \mathbf{s}_0)),$$

que en el caso del KS se aproxima al predictor óptimo $E(Z(\mathbf{s}_0)|\mathbf{Z})$ de $Z(\mathbf{s}_0)$. Como aproximación de la varianza kriging de este predictor se puede utilizar:

$$\sigma_{KT}^2(\mathbf{s}_0) = \sigma_K^2(\mathbf{s}_0)\phi'(p_K(\mathbf{Y}, \mathbf{s}_0)).$$

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 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. En este apéndice se incluyen algunos ejemplos que pueden servir como introducción a las herramientas implementadas en este paquete. Para más detalles ver Bivand et al. (2013), consultar las viñetas *sp: classes and methods for spatial data* y *Plotting maps with sp* o la chuleta.

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
```

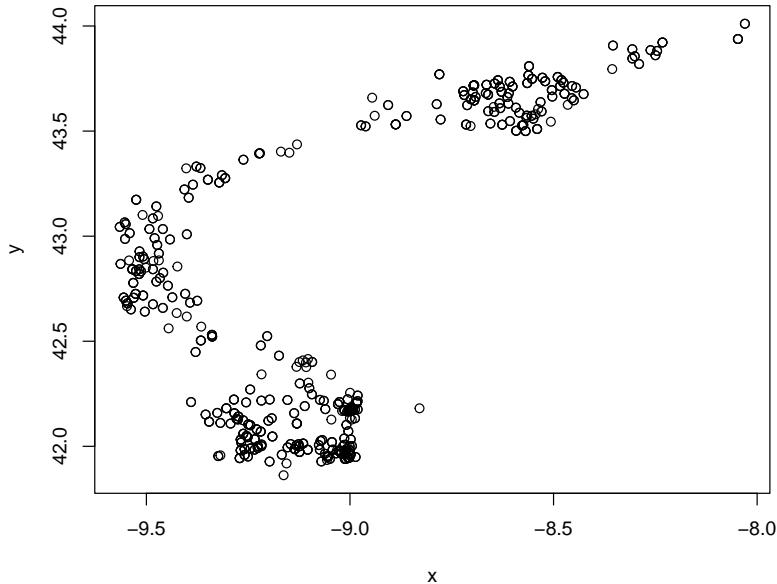
A.1.1 SpatialPoints y SpatialPointsDataFrame

- 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)

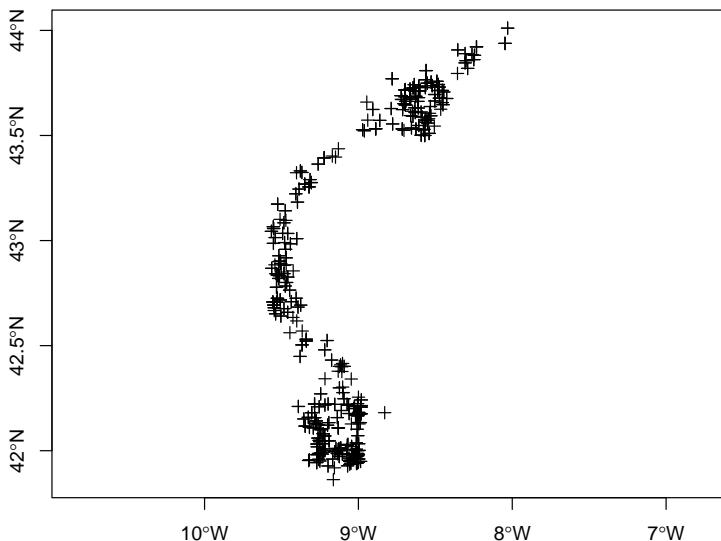
## Object of class SpatialPoints
## Coordinates:
##      min      max
## x -9.56538 -8.030065
## y 41.86240 44.010800
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84 +no_defs]
## Number of points: 676
```

```
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

```

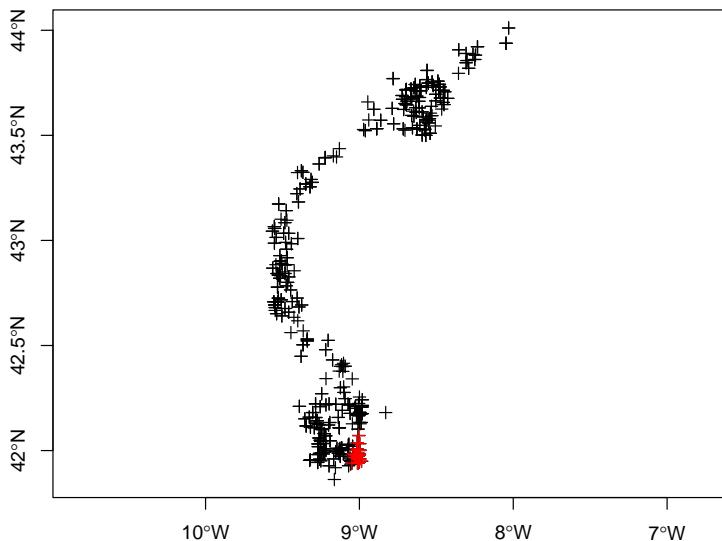
```

## [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 C4 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 F1 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):

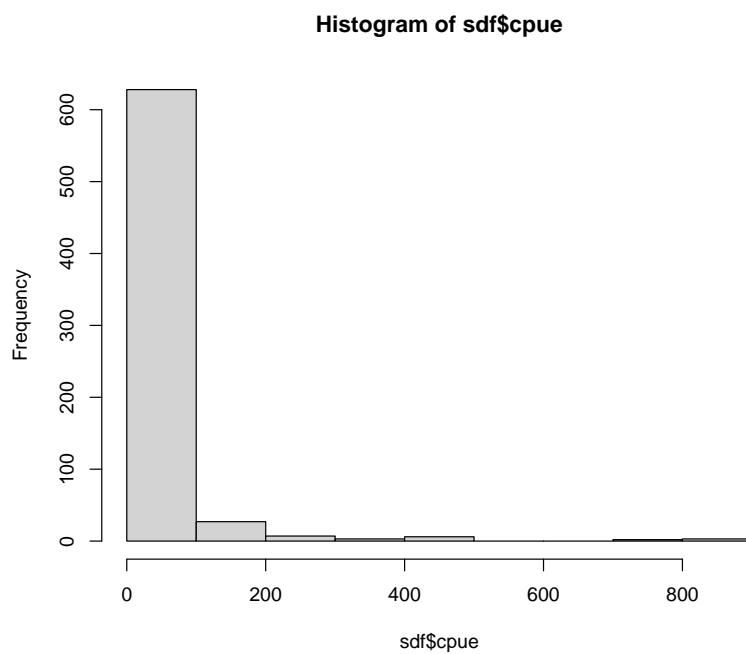
```

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

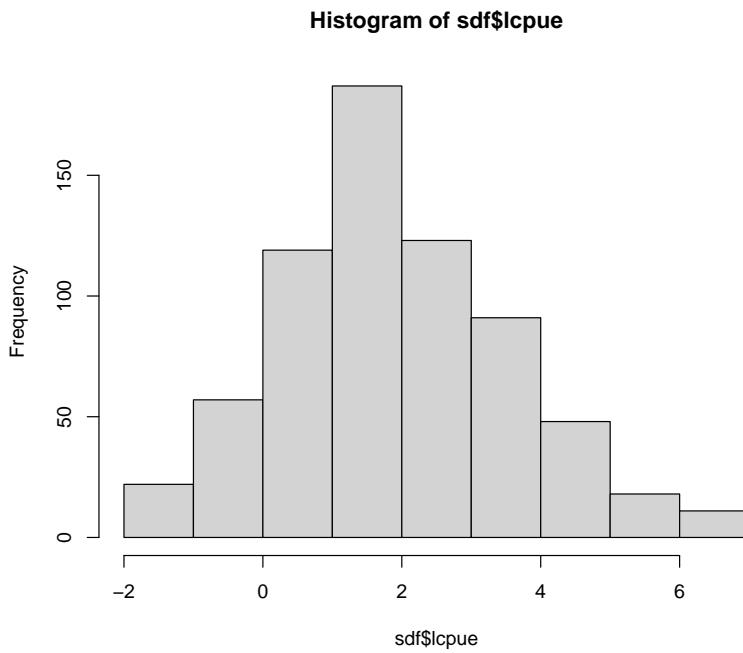
## Object of class SpatialPointsDataFrame

```

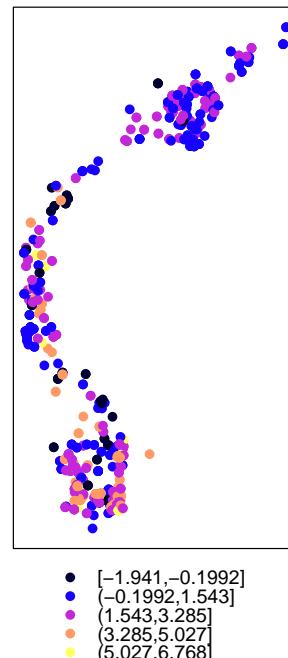
```
## Coordinates:  
##      min      max  
## x -9.56538 -8.030065  
## y 41.86240 44.010800  
## Is projected: FALSE  
## proj4string : [+proj=longlat +ellps=WGS84 +no_defs]  
## Number of points: 676  
## Data attributes:  
##      cpue      lcpue  
## Min.   : 0.1435   Min.   :-1.9411  
## 1st Qu.: 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

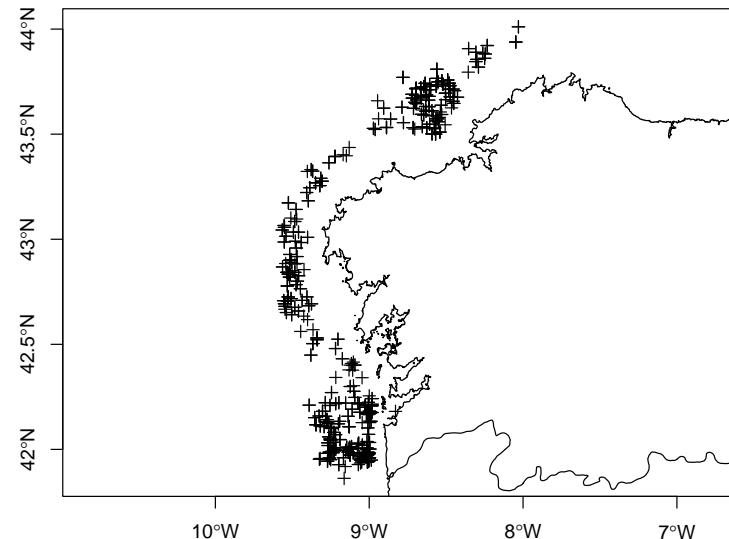
El fichero `costa.galicia.txt` contiene la costa de Galicia en formato Mapgen. Descargada del (difunto) Coastline Extractor

```
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)
```

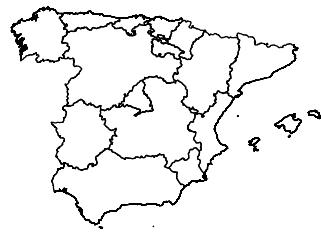
Carga de un objeto *gadm*:

```
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)
```



• ↗

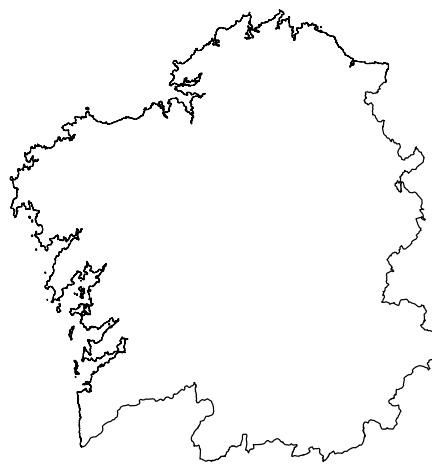
```
# Cuidado objeto muy grande: str(gadm)
# Mejor emplear str(gadm, 3)
```

Extraer autonomía de Galicia:

```
names(gadm)
```

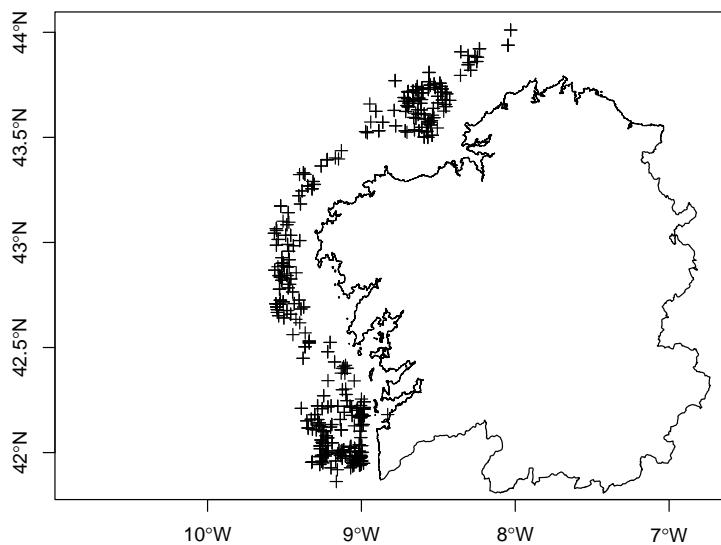
```
## [1] "OBJECTID"    "ID_0"        "ISO"         "NAME_0"       "ID_1"        "NAME_1"
## [7] "HASC_1"       "CCN_1"       "CCA_1"       "TYPE_1"      "ENGTYPE_1"  "NL_NAME_1"
## [13] "VARNAME_1"

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

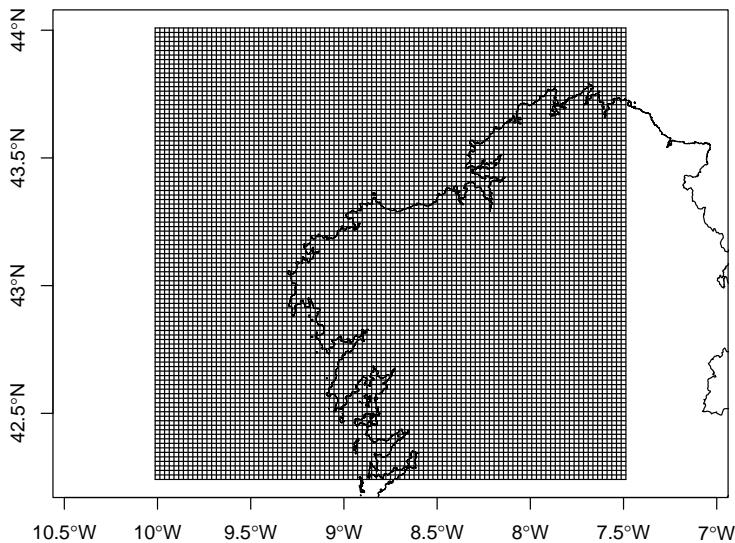
Importante si se utiliza el paquete `gstat...`

```
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)
```



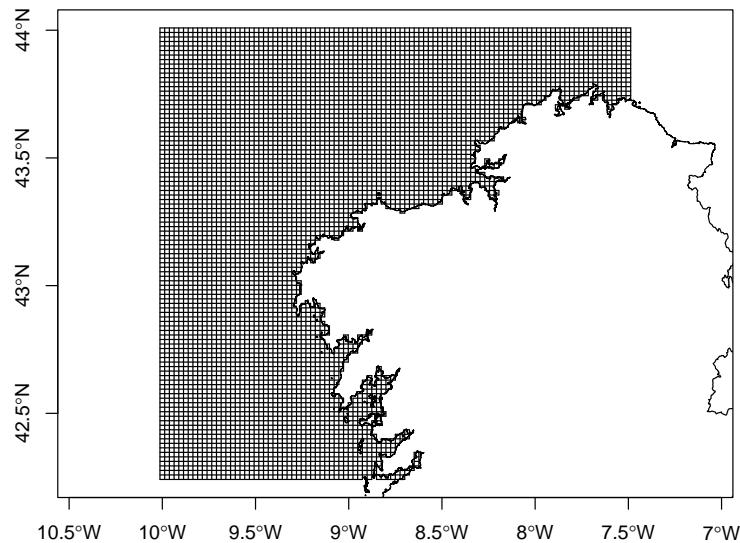
```

# 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

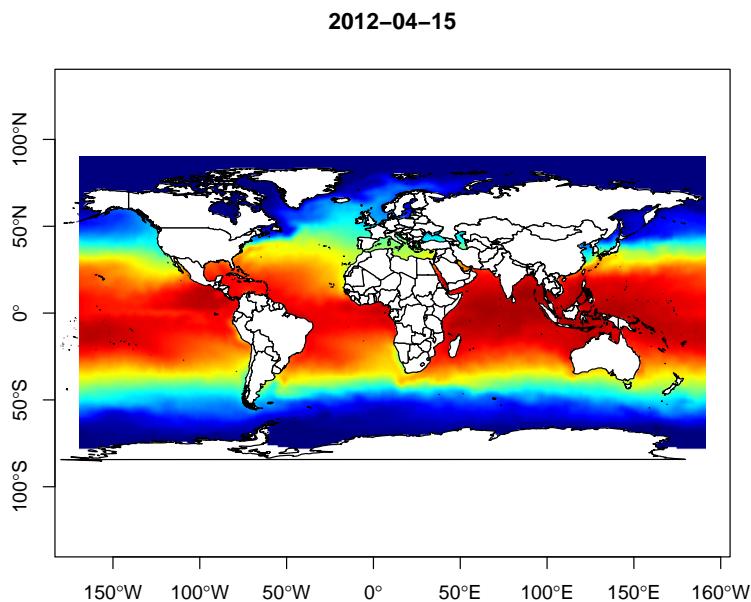
A través de R (paquetes que emplean sp) es fácil exportar datos a otras aplicaciones (e.g. Google Earth).

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*:

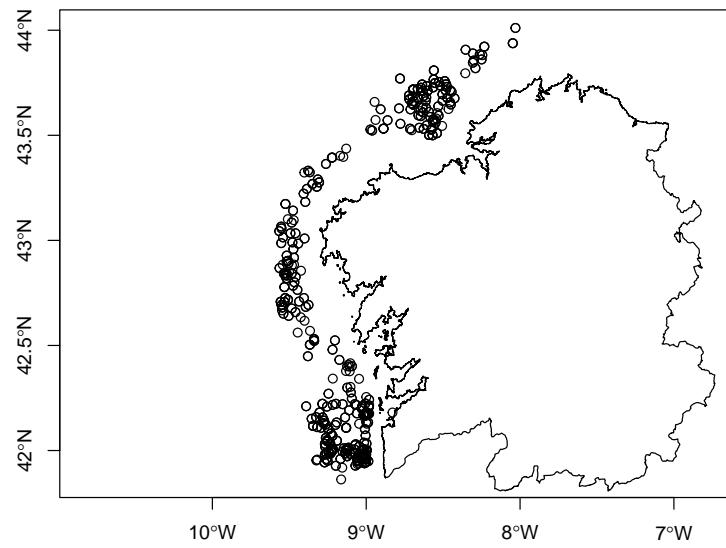
```
load("datos/sstsp.RData") # SST 15-04-2012
jet.colors <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow", "#FF7F00")
image(sstsp, col = jet.colors(128), axes = TRUE)
title(attr(sstsp@data, "label"))

# 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)
```



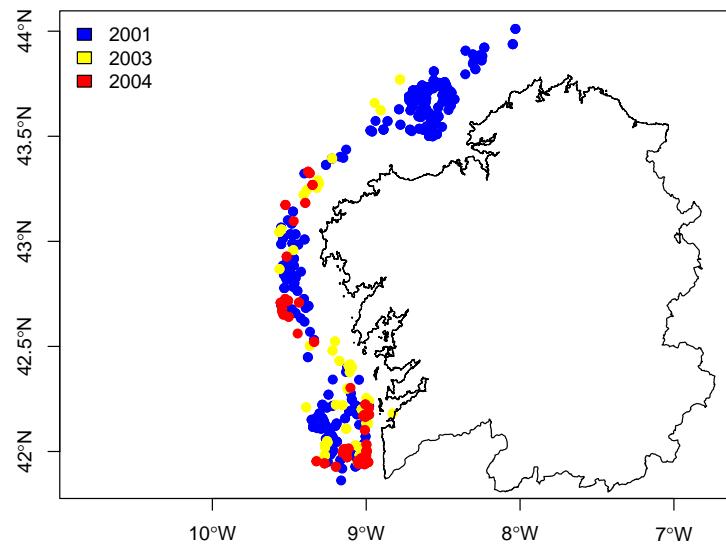
A.3 Representaciones gráficas

```
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)
```



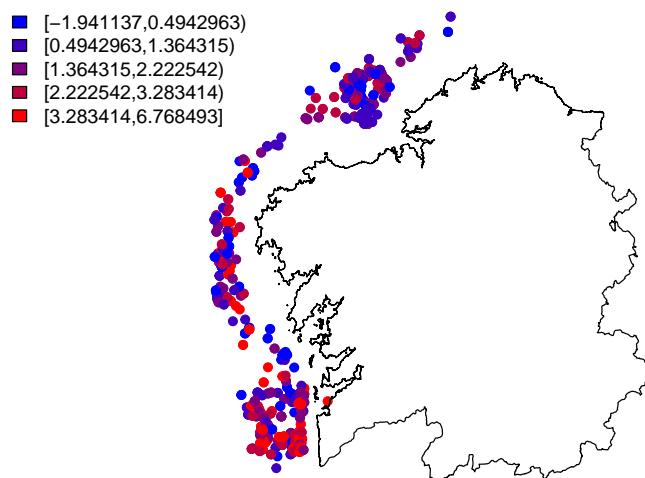
Usando p.e. la función classIntervals del paquete classInt se puede establecer los colores en función de una variable continua:

```
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)
```

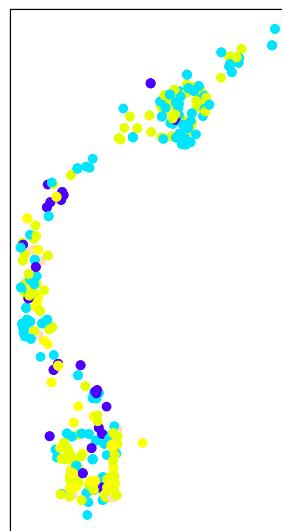
A.3.2 Gráficos lattice: spplot

- 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)
```

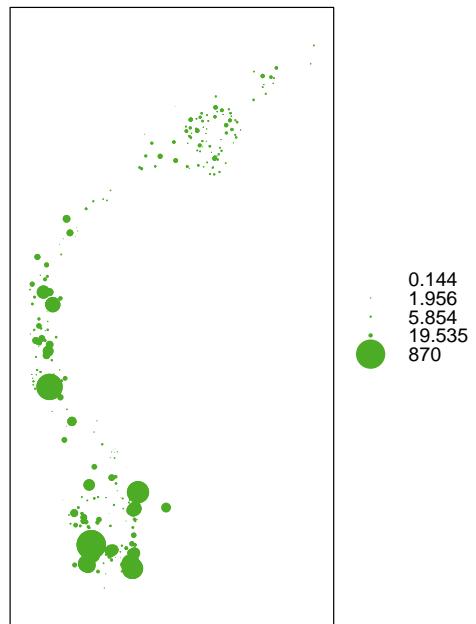
CPUE (escala logarítmica)



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

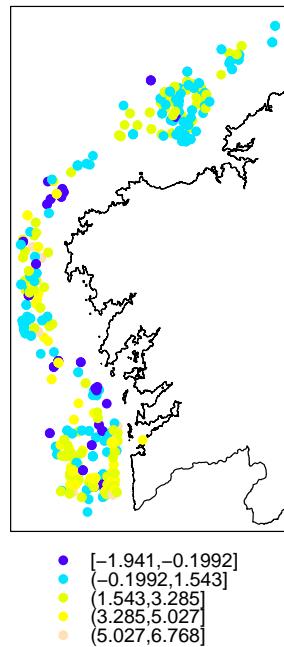
```
bubble(sdf, "cpue", main = "CPUE")
```

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 (alternativa al paquete `gstat`). 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)
```

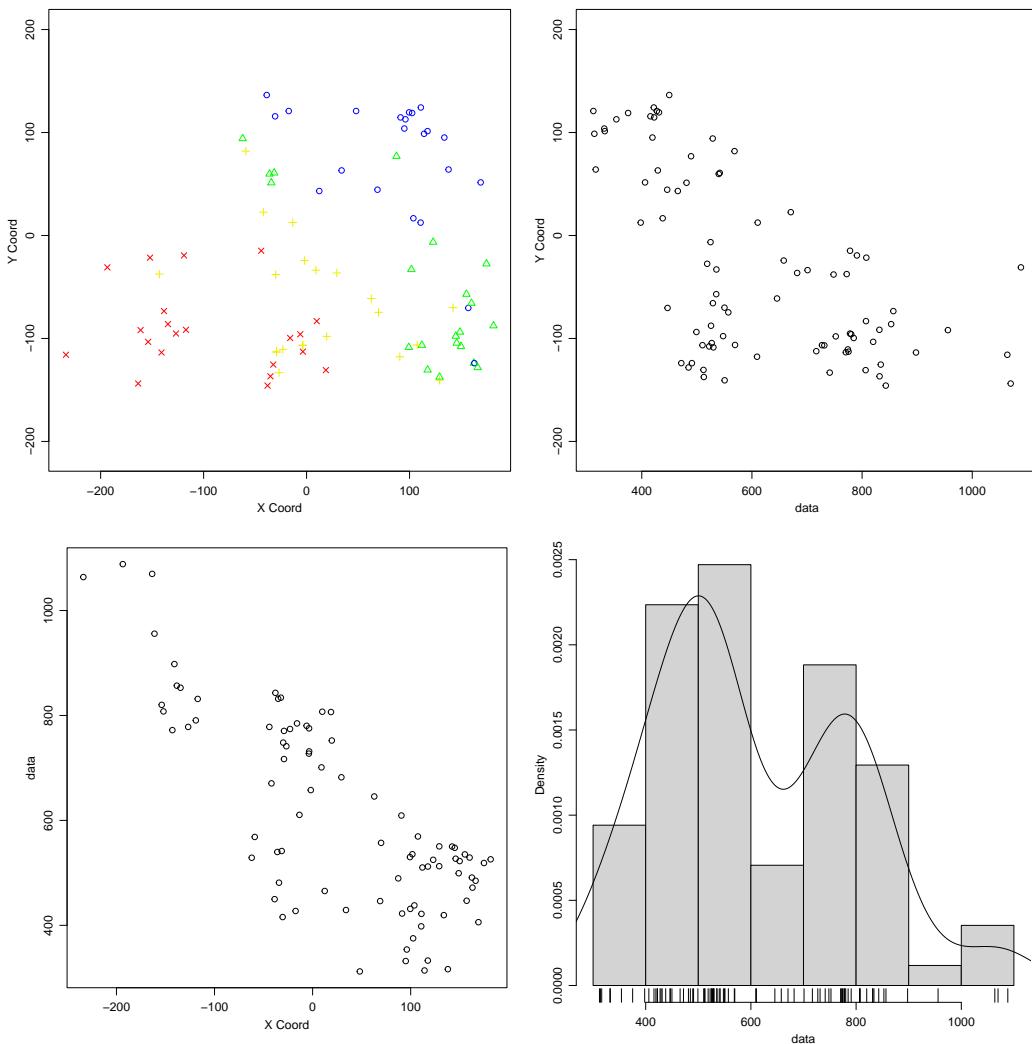
En la documentación de las funciones `as.geodata` y `read.geodata` hay más información sobre cómo importar/convertir datos.

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`).

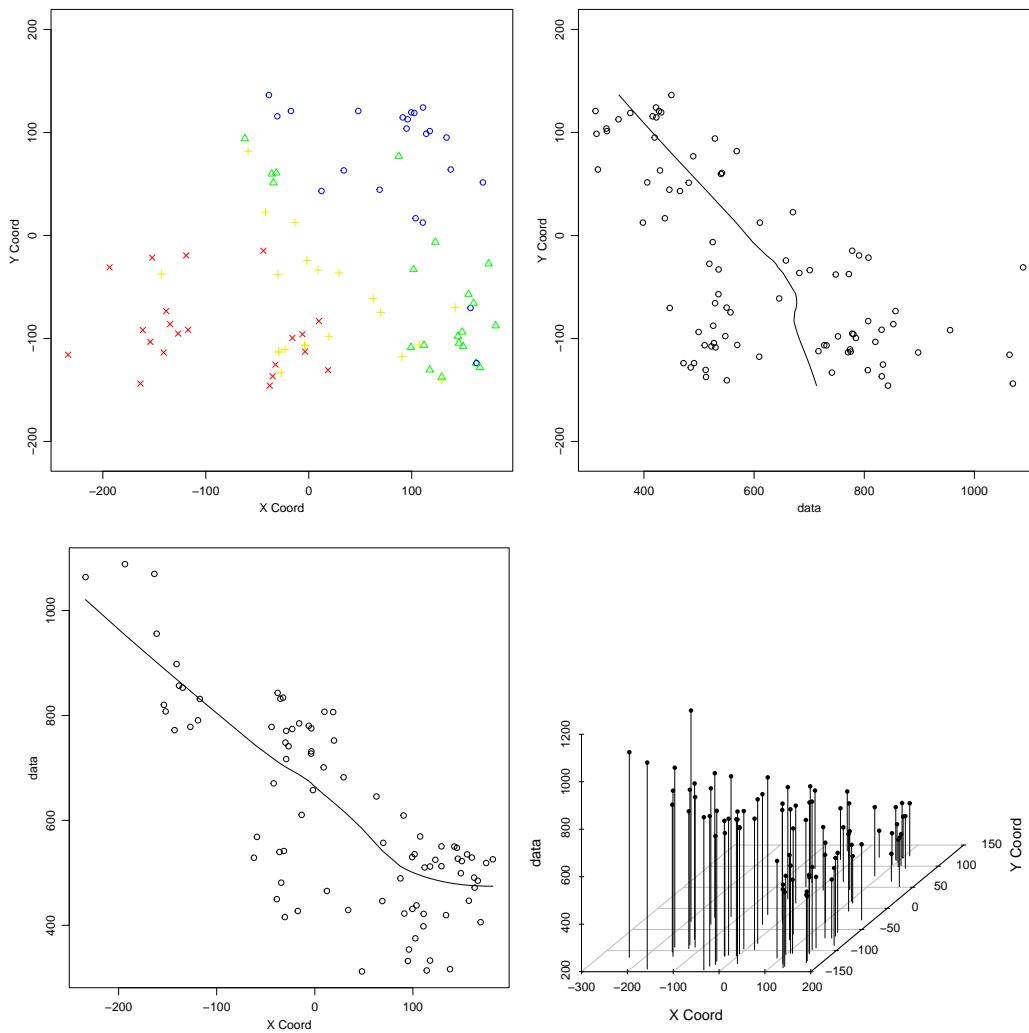
La función `plot()` genera por defecto gráficos de los valores en las posiciones espaciales (distinguiendo según cuartiles), los datos frente a las coordenadas y un histograma de los datos:

```
plot(wolfcamp)
```



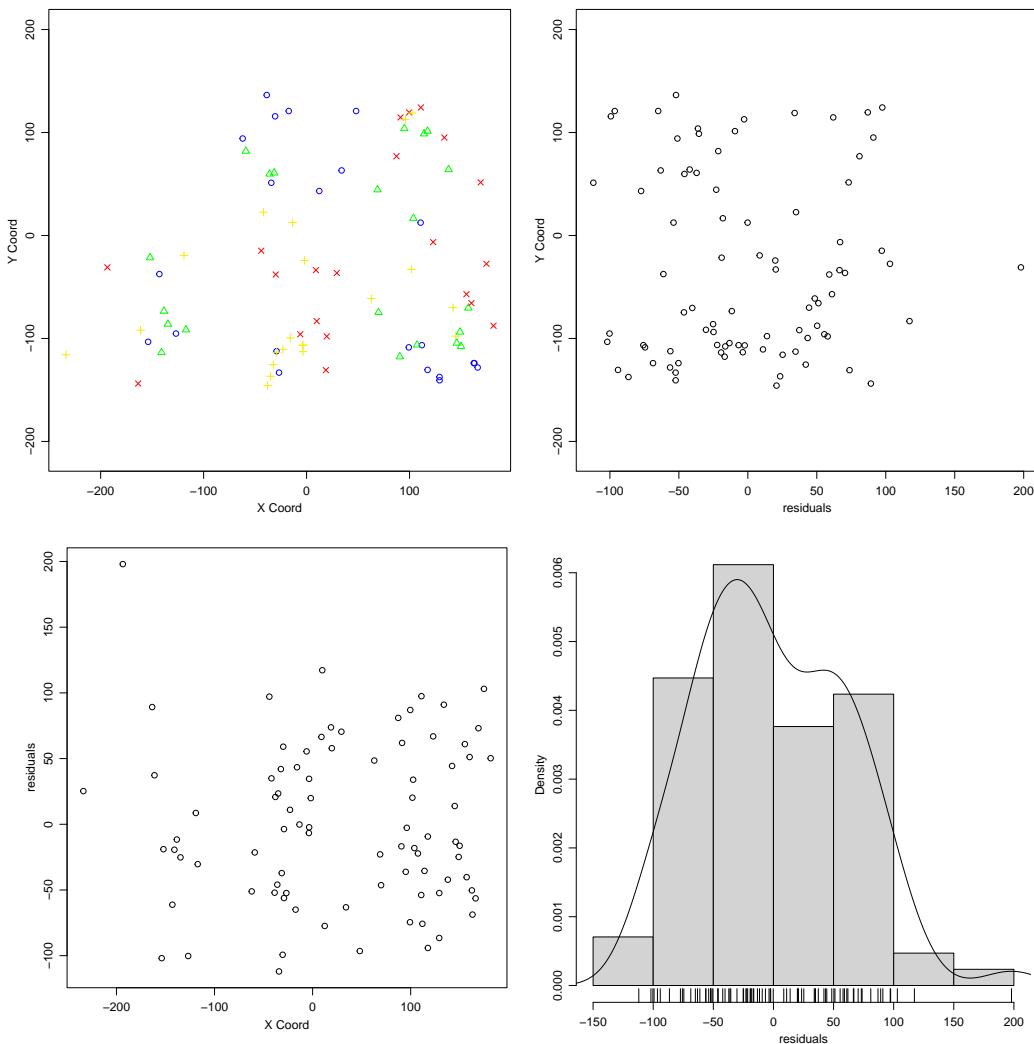
Los gráficos de dispersión de los datos frente a las coordenadas nos pueden ayudar a determinar si hay una tendencia. También, en lugar del histograma, nos puede interesar un gráfico de dispersión 3D

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



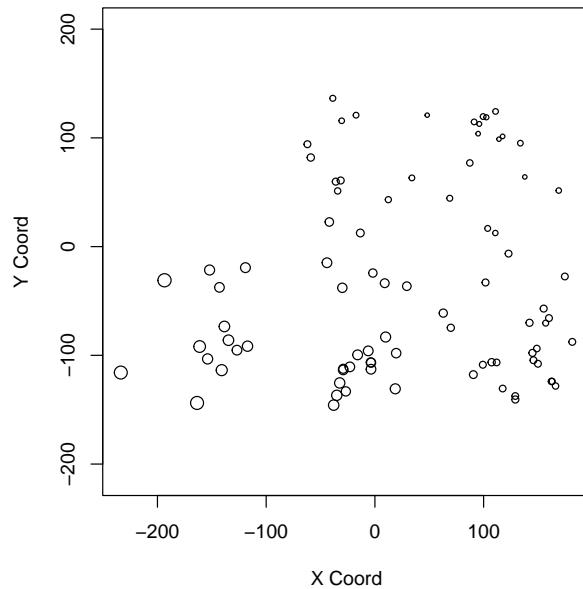
Si se asume que hay una tendencia puede interesar eliminarla:

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



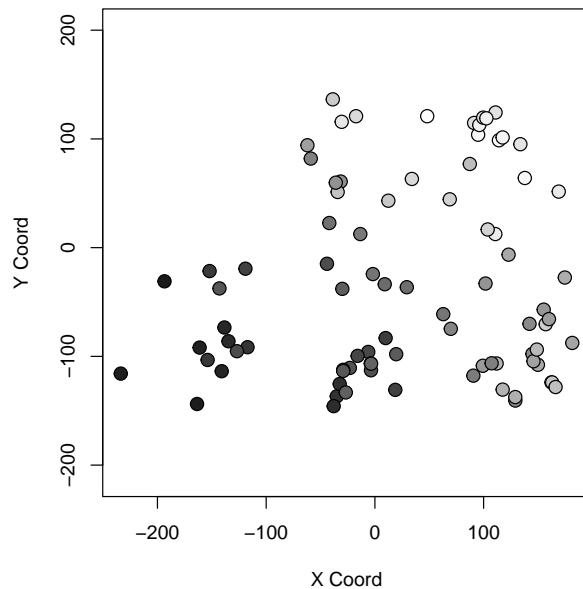
El comando `points(geodata)` (función `points.geodata`) genera un gráfico con las posiciones de los datos (y por defecto con el tamaño de los puntos proporcional al valor):

```
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

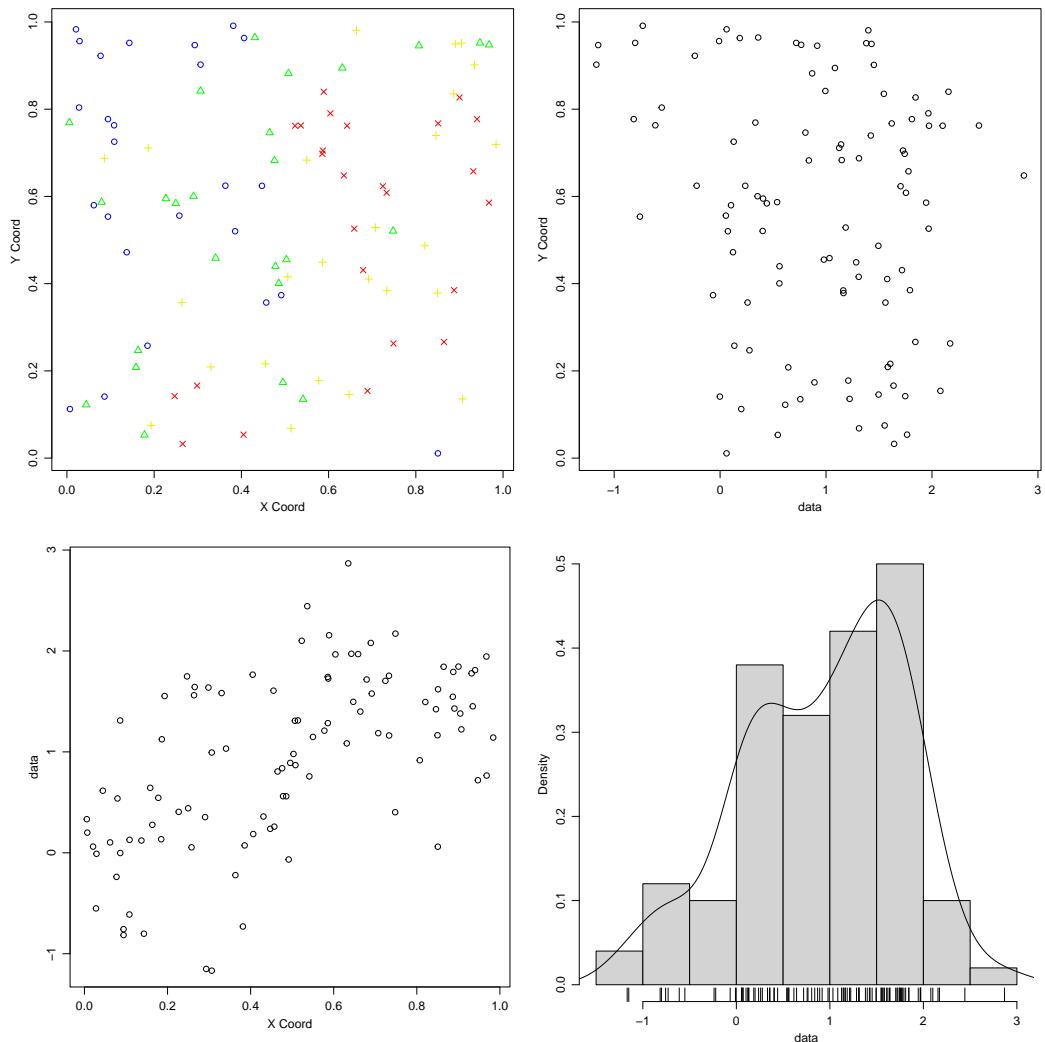
En la primera parte de esta sección consideraremos un proceso espacial sin tendencia:

```

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.

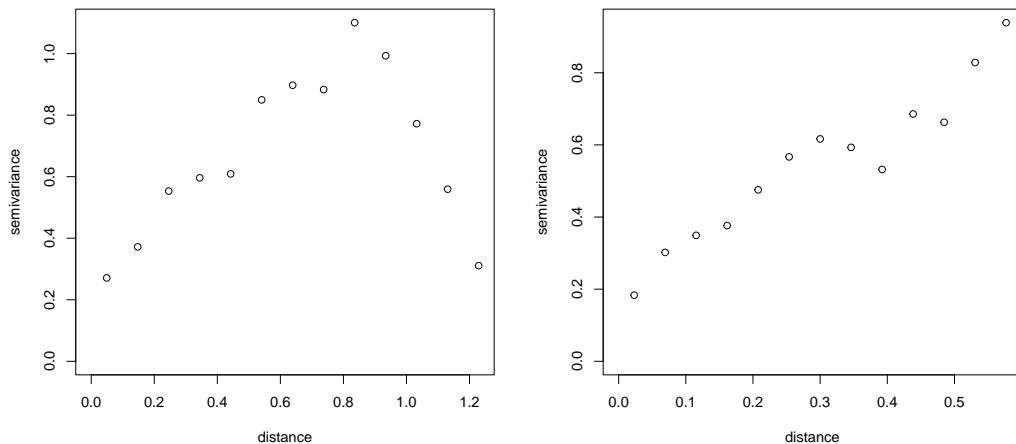
B.3.1 Variogramas empíricos

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
```

```

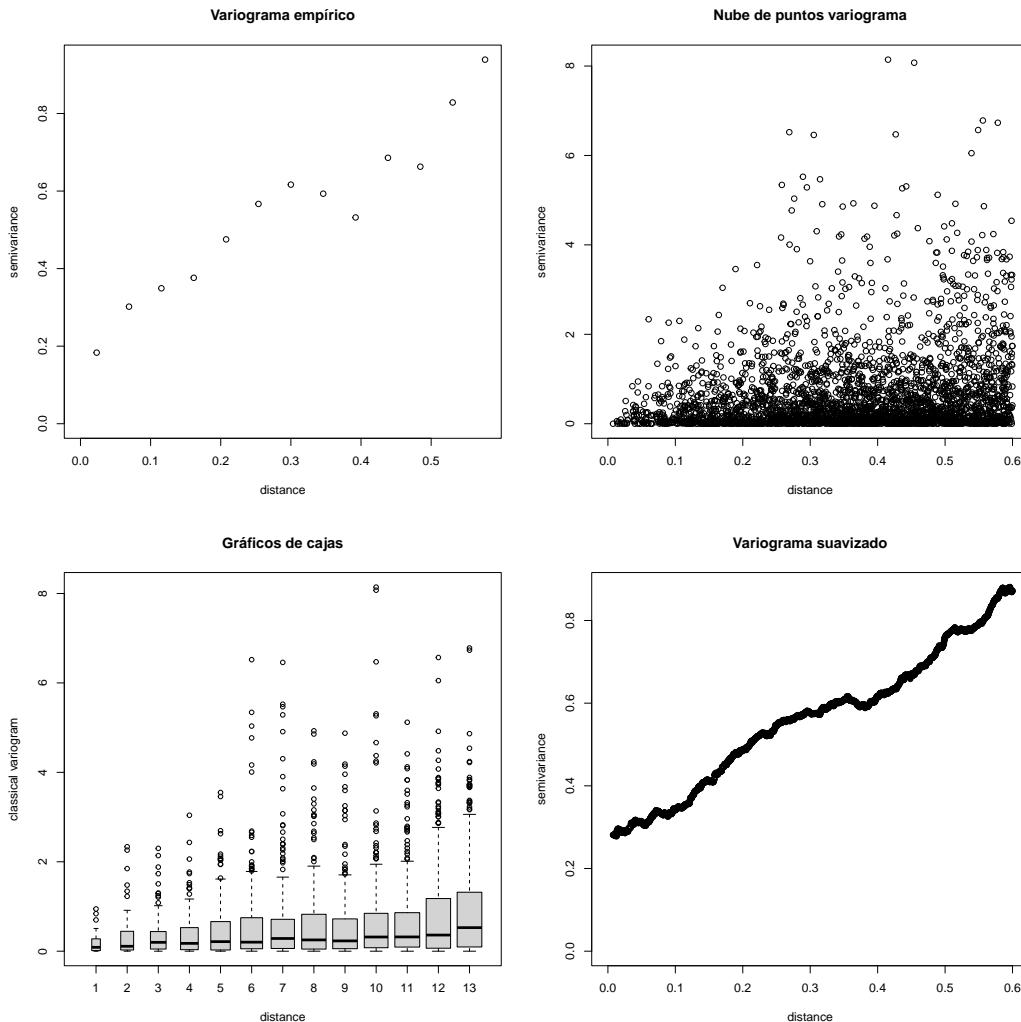
vario.c <- variog(s100, max.dist=0.6, op="cloud") #nube

## 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"`:

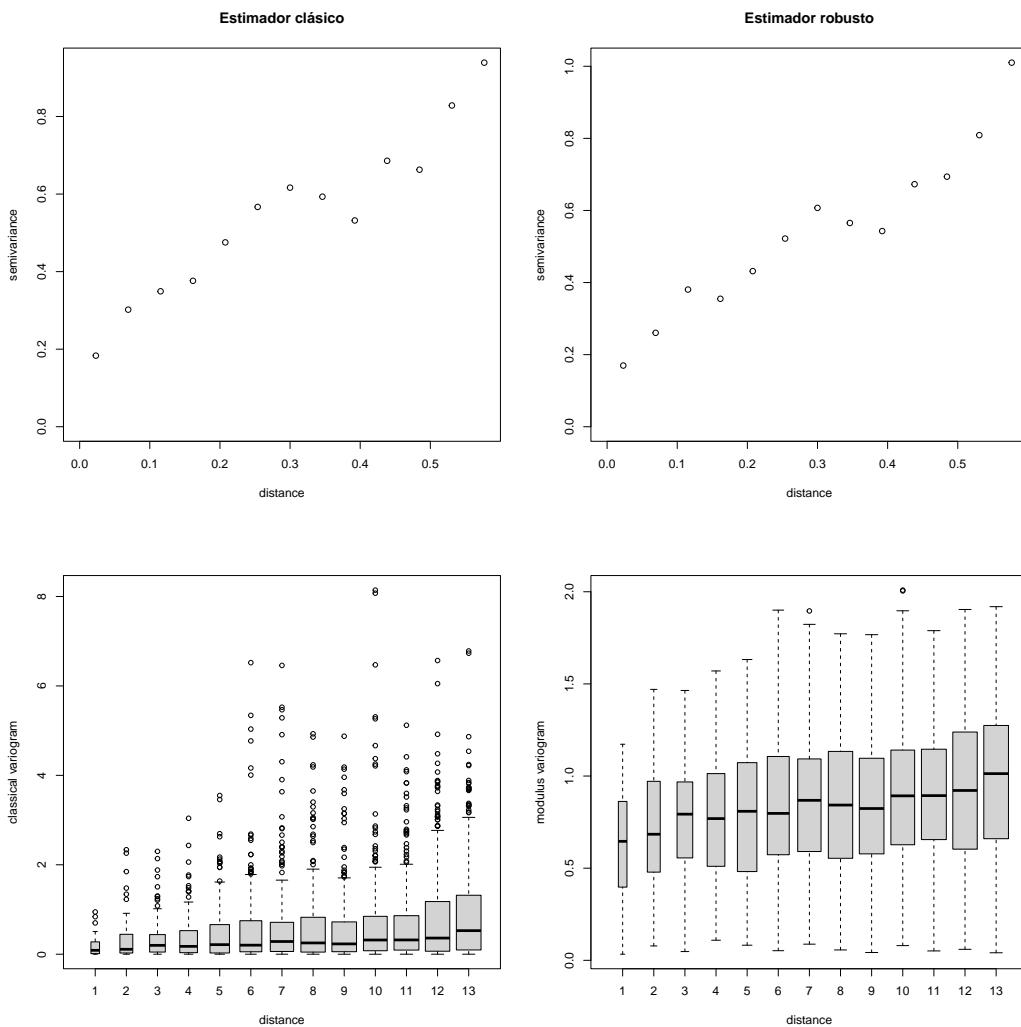
```

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)

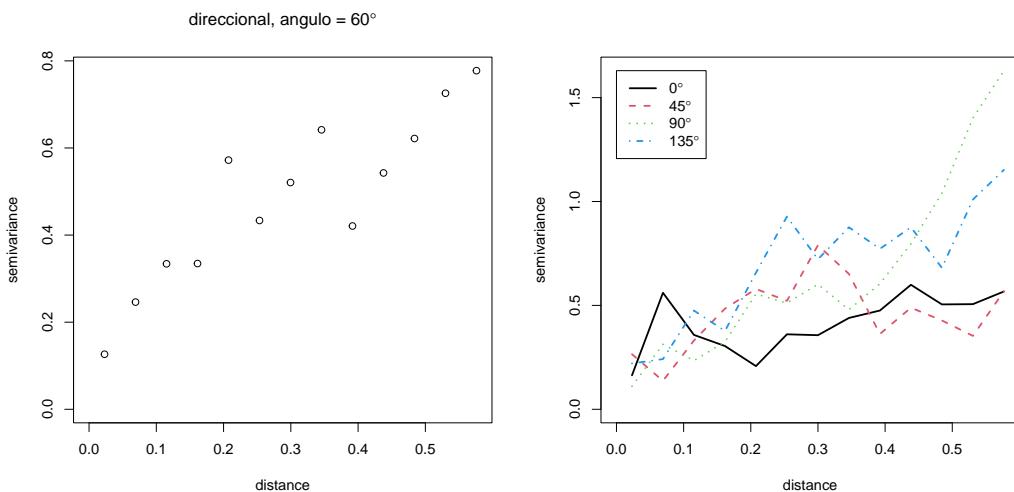
```

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`).

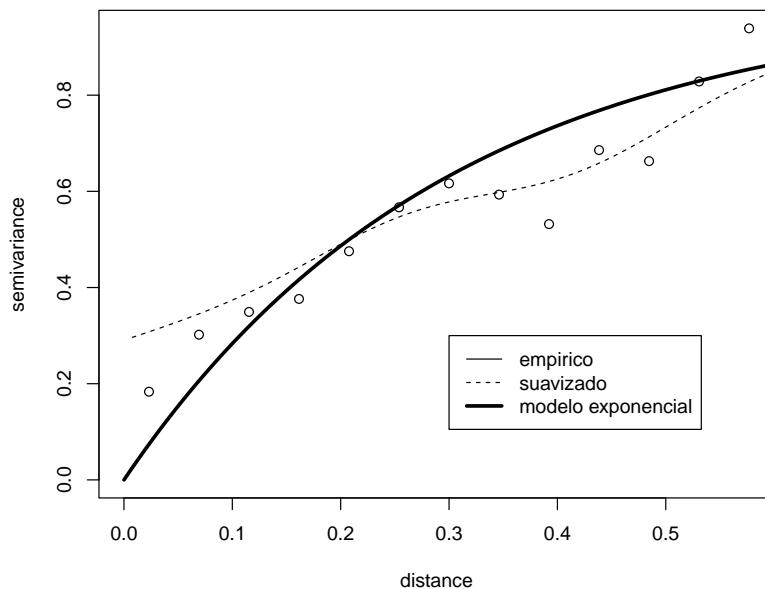
Ejemplo de ajuste “a ojo”:

```
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

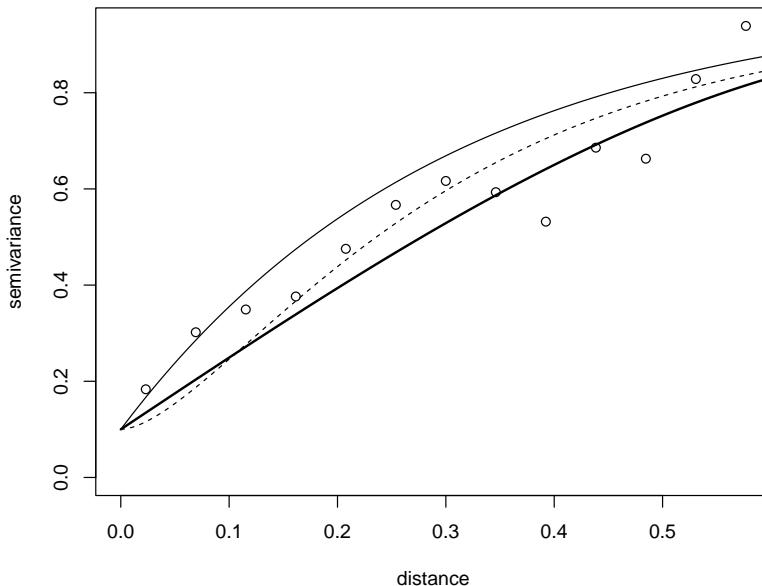
## variog: computing omnidirectional variogram
plot(vario.b)
lines(vario.s, type = "l", lty = 2)

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.)

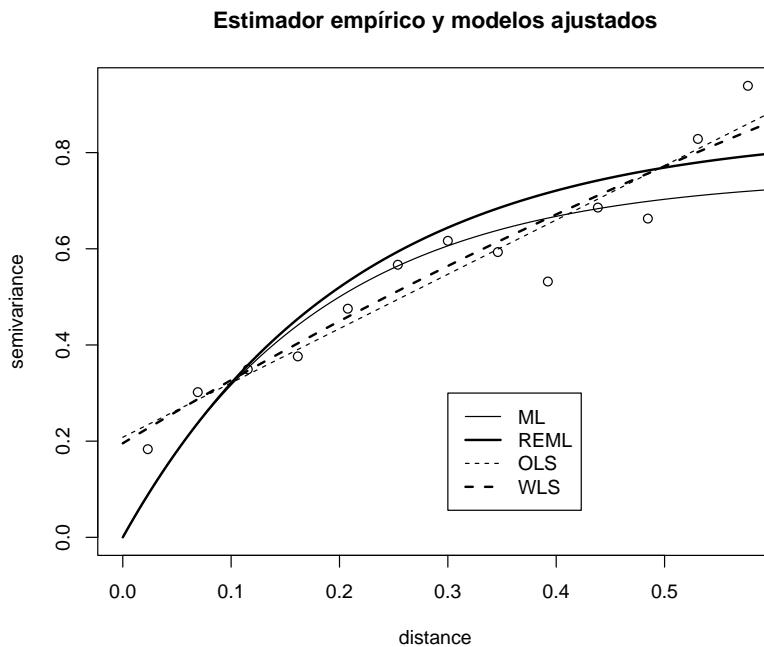
Representación gráfica junto al estimador empírico:

```

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

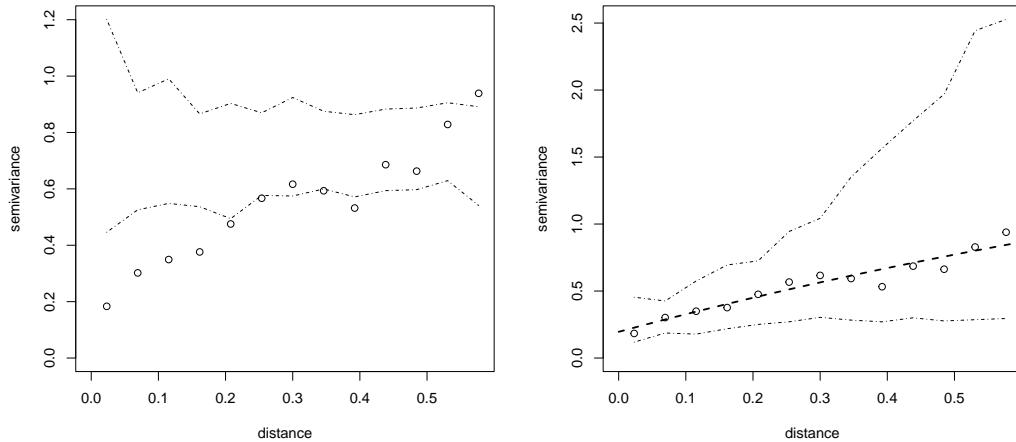
Se pueden obtener dos tipos de envolventes (envelopes, i.e. valores máximos y mínimos aproximados) del variograma empírico mediante simulación:

- 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)
```

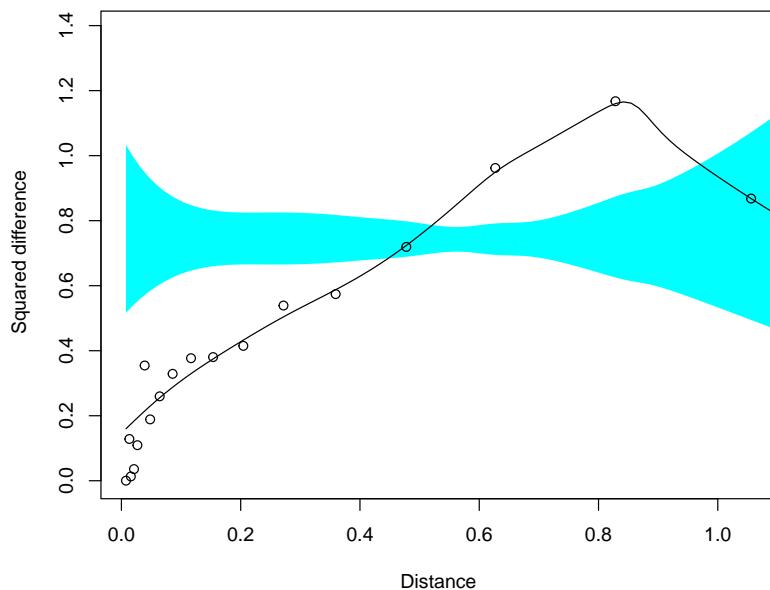


```
par(oldpar)
```

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".

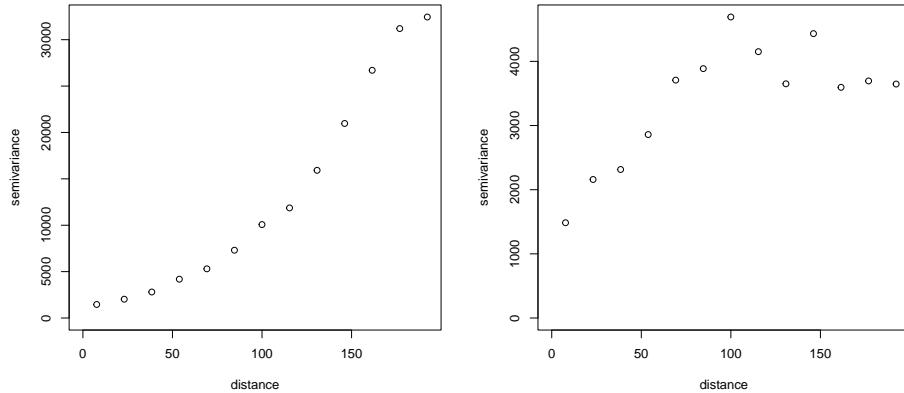
B.3.4 Estimación del variograma en procesos no estacionarios

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)

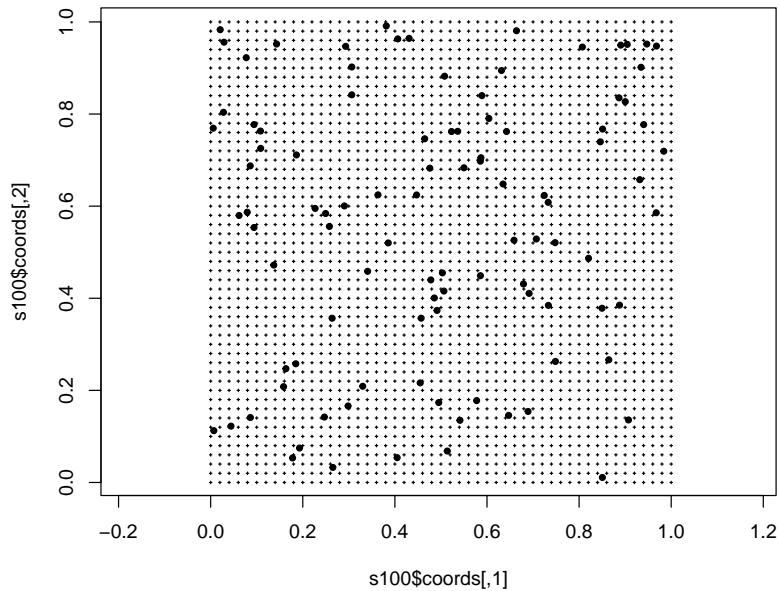
El paquete `geoR` dispone de opciones para los métodos kriging tradicionales, que dependiendo de las suposiciones acerca de la función de tendencia se clasifican en:

- *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):

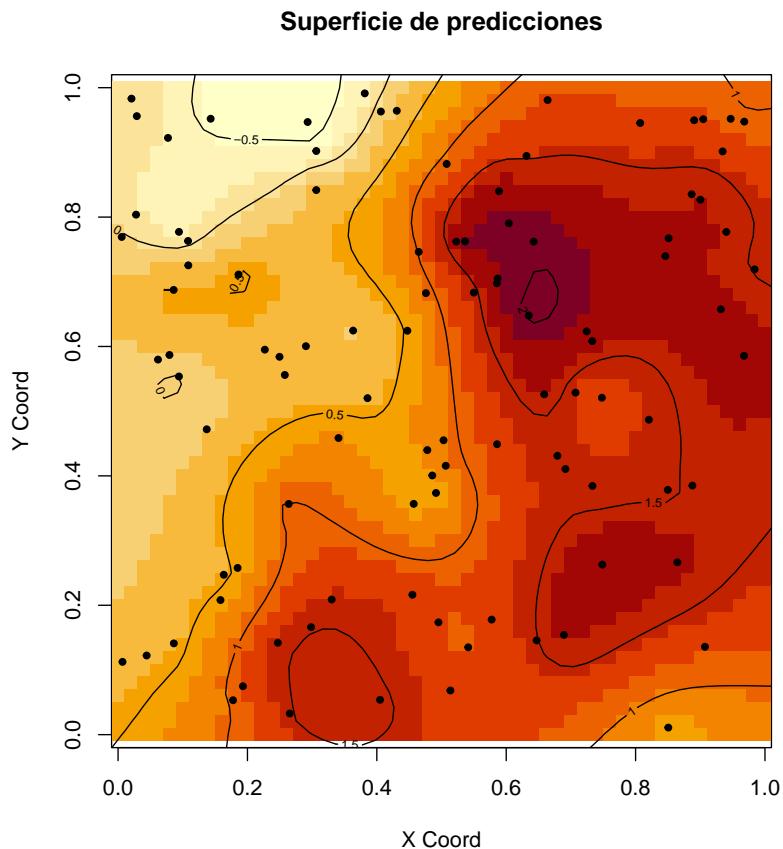
```
names(ko.wls)
```

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

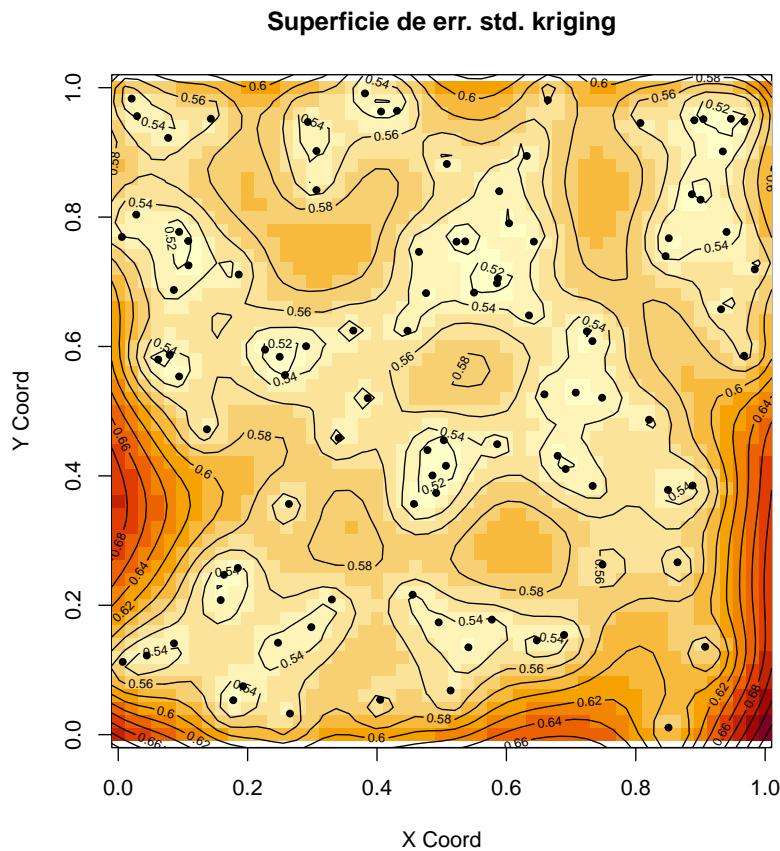
Para ver todas las opciones de kriging disponibles ejecutar ?krige.control. Para kriging con vecindario local (archivos de datos grandes) se puede utilizar la función ksline.

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
```



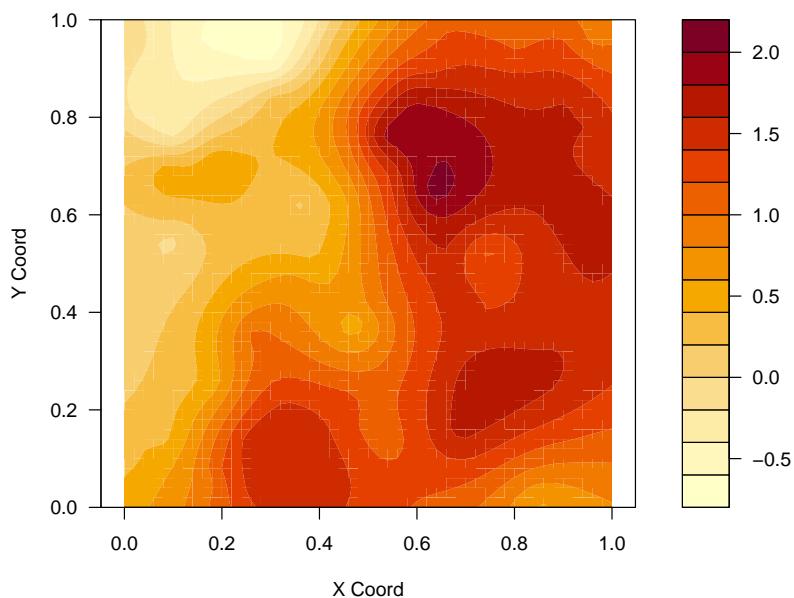
```
image(ko.wls, coords.data=s100$coords, values = sqrt(ko.wls$krige.var), main = "Superficie de errores")
contour(ko.wls, values = sqrt(ko.wls$krige.var), add = TRUE)
```



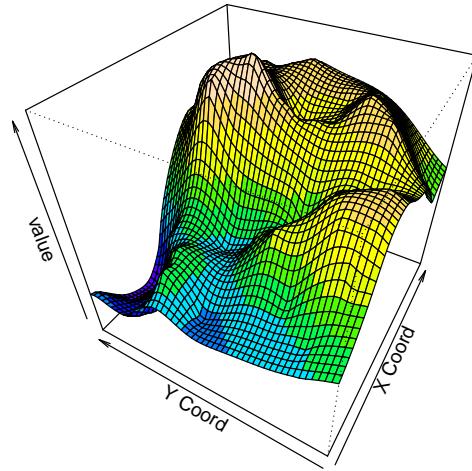
```
# par(oldpar)
```

Otras opciones:

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

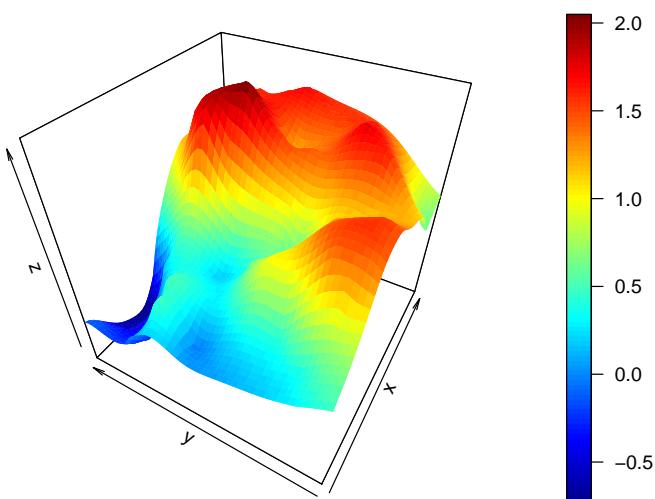


```
fcol <- topo.colors(10)[cut(matrix(ko.wls$pred, nrow=51, ncol=51)[-1,-1], 10, include.lowest=TRUE)]
persp(ko.wls, theta=-60, phi=40, col=fcol)
```



```
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)

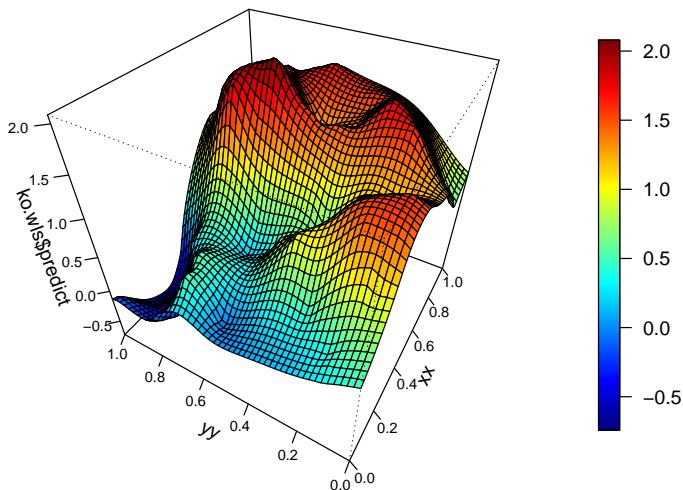
## Loading required package: npsp

## 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)

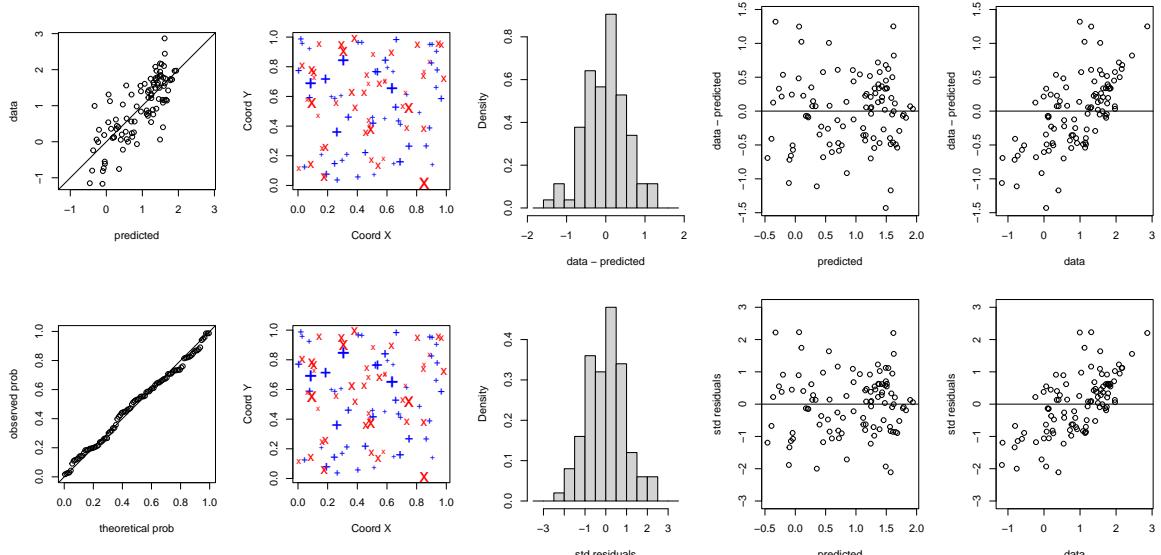
## xvalid: number of data locations      = 100
## xvalid: number of validation locations = 100
## xvalid: performing cross-validation at location ... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
## xvalid: end of cross-validation

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

- Tobler, W. (1970). A computer movie simulating urban growth in the Detroit region. *Economic Geography*, **46**, 234-240.
- 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/>