FACULTAD DE ESTUDIOS ESTADÍSTICOS

GRADO EN ESTADÍSTICA APLICADA

Curso 2024/2025

Trabajo de Fin de Grado

TÍTULO:* *Predicción espacial de la presencia del lobo ibérico (Canis lupus signatus) en la Península Ibérica usando factores ecológicos y humanos

***Alumno*: Sofía Rodríguez Rodríguez**

***Tutor*: Javier Portela García-Miguel**

Junio de 2025

UNIVERSIDAD COMPLUTENSE
MADRID

Índice

1. Introducción

1.1. Justificación del estudio

El lobo ibérico (*Canis lupus signatus*) es una de las especies más emblemáticas y controvertidas de la fauna peninsular. Su nombre científico tiene origen en las manchas oscuras que presenta en la parte anterior, la cruz y la cola. Su rol ecológico como depredador le confiere una gran importancia en el equilibrio de los ecosistemas, al tiempo que su presencia ha generado históricamente conflictos con actividades humanas como la ganadería.

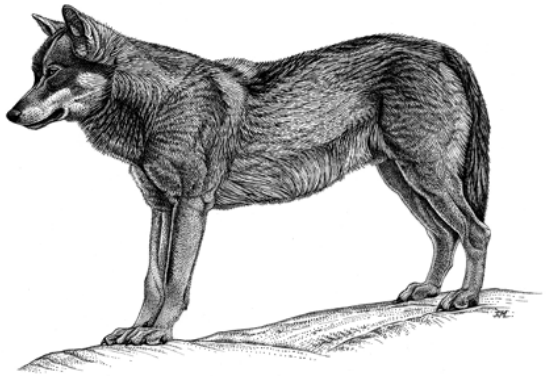

Figura 1: Lobo ibérico (*Canis lupus signatus*).

Fuente: Ministerio para la Transición Ecológica y el Reto Demográfico (MITECO).

Entender qué factores son los que determinan su distribución en la actualidad es esencial para poder establecer mecanismos o estrategias de conservación eficaces, especialmente en un periodo histórico de expansión humana y de cambio climático.

Este trabajo contribuye a ese objetivo mediante la elaboración de modelos predictivos creados tanto con variables ambientales (como el clima y la topografía), como con variables de población. Se combina el enfoque ecológico con el estadístico, con el objetivo de conocer la distribución de la probabilidad de presencia en la peninsula, y así poder ayudar a elaborar mecanismos de preservación, gestión territorial o ganadera u otros objetivos. (National Geographic España, 2023)

1.2. Contexto ecológico y distribución del lobo ibérico

Como he mencionado anteriormente, es una subespecie endémica, clave para el equilibrio del ecosistema de la península. Sin embargo, la amenaza creciente de la actividad humana ha hecho que este se vea gravemente afectado.

Su población experimentó un retroceso importante desde comienzos del siglo XX, llegando a estar incluso al borde de la extinción, y no es hasta pasado 1970, cuando la población de lobos empieza a crecer hasta hoy en día, donde se puede decir que se encuentra en expansión.

Históricamente, esta subespecie se distribuía por toda la península. Hoy en día, aunque se encuentra legalmente protegido en gran parte de ella, su distribución es mucho más desigual, concentrándose principalmente en el noroeste peninsular.

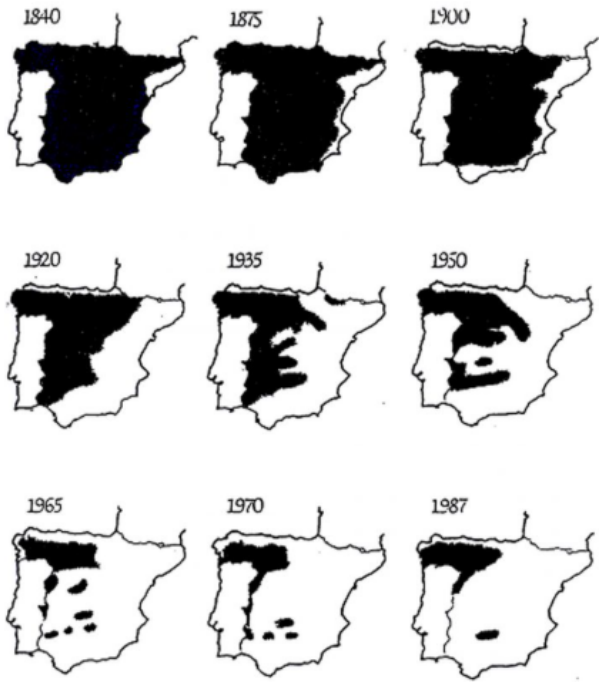

Figura 2: Evolución histórica de la distribución del lobo en España desde 1840 hasta 1987. Fuente: [48].

Las principales poblaciones actuales se encuentran en Galicia, Castilla y León, el norte de Portugal, Cantabria y Asturias; aunque también encontramos poblaciones más reducidas en el Pais Vasco, en la Rioja y incluso en las zonas montañosas de Andalucía (Sierra Nevada).

Se alimenta principalmente de las presas que caza, desde grandes ungulados (jabalíes, corzos o ciervos) hasta mamíferos de menor tamaño (conejos u ovejas), aunque también puede alimentarse de animales muertos que hayan encontrado o restos humanos que puedan encontrar en vertederos o lugares donde se deposite basura.

Diversos factores determinan la presencia o ausencia del lobo en determinadas zonas: condiciones climáticas, disponibilidad de hábitat adecuado, cobertura forestal, presencia humana y conectividad ecológica, entre otros. La modelización de estos factores permite entender los patrones actuales y proyectar escenarios futuros bajo distintas condiciones ambientales (National Geographic España, 2023).

1.3. Objetivos del trabajo

El presente Trabajo de Fin de Grado tiene como objetivo principal elaborar modelos predictivos para comprender mejor los factores que influyen en su presencia y elaborar mapas de probabilidad que reflejen la distribuión actual del lobo en la península ibérica.

Los objetivos específicos son:

- Recopilar y procesar información geográfica, ambiental y humana relevante sobre la presencia/ausencia del lobo.
- Aplicar técnicas de selección de variables para identificar los factores más determinantes.

- Evaluar diferentes modelos predictivos (regresión logística, Random Forest, Gradient Boosting) y comparar su rendimiento.
- Observar la distribución espacial de la probabilidad de presencia a lo largo de la Península Ibérica.

2. Marco teórico

2.1. Factores que influyen en la distribución de especies

Según Lobo (2000), la distribución de una especie a lo largo de un territorio nunca es aleatoria, siempre hay condicionantes ecológicos y evolutivos detrás; en este trabajo trataremos los primeros. Las condiciones ambientales actúan como filtros, definiendo dónde puede sobrevivir una especie. Se entienden por filtros las barreras ambientales que determinan si una especie puede sobrevivir y reproducirse en un determinado entorno (Guisan & Zimmermann, 2000).

Distinguimos entre factores bióticos (depredadores, presas, etc.), factores abióticos (clima, altitud, etc.) y factores antrópicos (presión humana). Estos factores no actúan de forma aislada, sino que se interrelacionan, modulando conjuntamente las condiciones de hábitat disponibles para una especie. Este trabajo se centrará en analizar cómo estas variables, especialmente las abióticas y antrópicas, las cuales se detallarán en secciones posteriores, influyen en la distribución y presencia del lobo.

2.2. El concepto de nicho ecológico

Cuando hablamos de nicho ecológico, no solo hablamos del lugar donde vive una especie, sino de todo el conjunto de condiciones en las que una especie puede sobrevivir, crecer y reproducirse; hablamos de la relación de la especie con su ambiente.

En 1917, Joseph Grinnel introduce el concepto de nicho ecológico como el conjunto de condiciones abióticas necesarias para la existencia de una especie, basándose en el estudio del zorzal californiano. Observó que la distribución de este, la determinaban un conjunto muy específico de condiciones, tanto climáticas como geográficas y por ello su rango de habitabilidad era tan restringido. Una parte importante de su tesis es que no creía que dos especies pudieran ocupar nichos similares si estaban en ecosistemas diferentes; para Grinnell, el nicho era una proyección geográfica donde una especie podía vivir.

Más adelante, en 1927, Charles S. Elton establece las bases de la ecología animal, creando así una ciencia independiente. Gracias a un gran estudio de campo y a diversos ejemplos concretos, este introduce conceptos nuevos que son pilares en la ecología actual y, entre ellos, reexplora el concepto de nicho ecológico. Elton define el nicho de una especie como su "lugar de ambiente biótico", dicho de otra manera, su función dentro de la comunidad (especialmente en términos de sus relaciones alimenticias y con los depredadores). Este le daba al nicho ecológico un enfoque más funcional, lo cual contrastaba con la visión más abiótica de Grinell. Para Elton, dos especies diferentes sí podrían ocupar nichos parecidos, siempre y cuando estas desempeñaran en su ecosistema correspondiente un rol ecológico parecido.

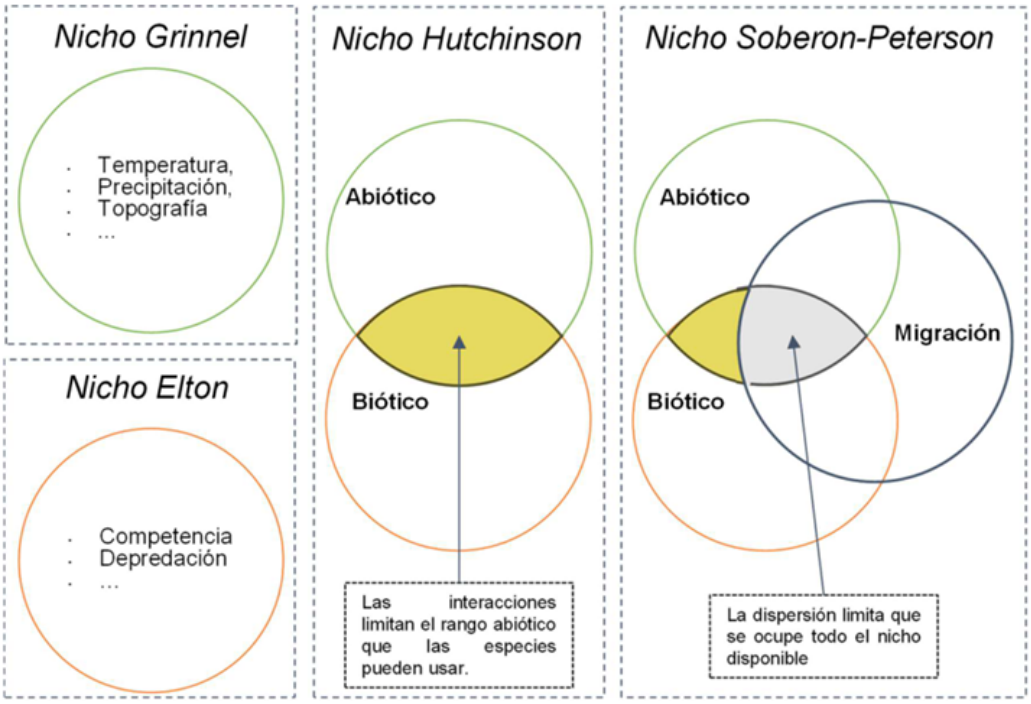

Figura 3: Representación comparativa de los conceptos de nicho según Grinnell, Elton, Hutchinson y Soberón-Peterson. Fuente: [21].

Sin embargo, la definición más utilizada actualmente en ecología es la de Hutchinson (1957), con una visión que, en contraste con la naturaleza más descriptiva de las aportaciones de Grinnel o Elton, posée un enfoque más matemático y multidimensional. Hutcherson define el nicho ecológico como un hipervolumen n-dimensional en un espacio abstracto, donde cada dimensión representa una variable ambiental relevante. Este concepto ha sido fundamental para el desarrollo de modelos de distribución de especies.

Para entender mejor las tres visiones, podemos usar un ejemplo cercano. Un enfoque grinnelliano de nicho diría: *"vive en zonas frías con cierto tipo de suelo"*; el enfoque funcional eltoniano lo expresaría como: *"es un depredador de roedores nocturnos"*; mientras que, desde la perspectiva hutchinsoniana, sería: *"la especie sobrevive si se cumplen simultáneamente las condiciones X, Y y Z"*.

2.3. Modelos de distribución de especies (SDMs)

Hasta este punto, se ha desarrollado un marco teórico-ecológico centrado en los conceptos clave que explican la distribución de las especies desde distintas perspectivas: el nicho, los factores abióticos, bióticos y antrópicos, y su interacción con el entorno. Sin embargo, si ya conocemos las condiciones que determinan la presencia de una especie en un entorno, ¿Cómo podemos predecir dónde podría estar presente y dónde no? Esta es precisamente la pregunta que sirve de nexo entre la teoría ecológica y la modelización estadística, y constituye el principal objetivo del trabajo. Y es en este cruce es donde se sitúa mi interés: aplicar herramientas estadísticas para abordar un problema ecológico complejo desde una perspectiva cuantitativa y predictiva. En este contexto, surgen los Modelos de Distribución de Especies (SDMs) como herramienta clave para integrar ecología y modelización.

En el año 2000, los ya anteriormente mencionados Guisan & Zimmermann definen los SDM's como una herramienta que relaciona observaciones de presencia/ausencia (o abundancia) de especies con variables ambientales, para estimar la distribución potencial de una especie en el espacio. Lo que busca el modelo es una relación estadística o matemática entre las observaciones biológicas y los predictores ambientales.

Para desarrollar esta idea, se basaron en el concepto de nicho ecológico de Hutchinson (1957) y en su distinción entre nicho fundamental y nicho realizado. Se entiende por nicho fundamental al conjunto teórico de todas las condiciones ambientales donde una especie podría sobrevivir y reproducirse, si no existiera ningún obstáculo u impedimento; en su lugar, el nicho realizado es el subconjunto real del nicho fundamental, donde la especie efectivamente vive (teniendo en cuenta factores externos como la presencia humana, otros animales o barreras físicas como ríos o carreteras).

En el mito de la caverna, Platón describe a unos prisioneros que solo ven sombras proyectadas en una pared. Ellos creen que esas sombras son la realidad, pero en realidad están viendo sólo una parte distorsionada y limitada de lo que hay "fuera de la caverna"; algo muy parecido ocurre con los SDM's. Estos no modelan directamente el nicho fundamental, sino que trabajan con el nicho realizado.

Por tanto, las predicciones están limitadas por la distribución actual observada, y no reflejan necesariamente todas las áreas donde la especie verdaderamente podría existir si no hubiera barreras; si hay zonas donde la especie podría estar pero no está, el modelo podría no detectarlas. Esto implica que los modelos no revelan "toda la verdad" ecológica sobre una especie, sino que capturan la porción observable del nicho, la que está representada en los datos disponibles de campo (Soberón, J. & Peterson, 2005). En la siguiente sección se detallarán las herramientas estadísticas específicas utilizadas para aplicar los SDMs en la práctica.

2.4. Técnicas estadísticas y de machine learning

En el ámbito del modelado ecológico (especialmente en los SDMs), la elección de las herramientas analíticas correctas desempeña un papel fundamental. Dado que el objetivo último de este trabajo es estimar la distribución del lobo ibérico, es necesario combinar tanto técnicas estadísticas más tradicionales, como técnicas más actuales de aprendizaje automático (lo que se conoce como machine learning), técnicas que son capaces de captar comportamientos o patrones más complejos o escondidos si se buscan con técnicas más restrictivas.

Por un lado, los modelos estadísticos, como por ejemplo los GLM (Modelos Lineales Generalizados), permiten interpretar directamente el efecto que tienen las variables independientes en la probabilidad de presencia. Por otro lado, los modelos de machine learning, como por ejemplo Random Forest o Gradient Boosting, son mucho más potentes predictivamente, pero más difíciles de interpretar.

Modelo	**Ventajas principales**	**Limitaciones**
GLM / GAM	Interpretabilidad, control de errores, y claridad en los efectos de las variables	Supone relaciones lineales o suavizadas; puede infraajustar si la realidad es más compleja
Random Forest	Robusto ante ruido, detecta interacciones y no requiere supuestos previos	Menor interpretabilidad y dificultad para explicar decisiones del modelo
Gradient Boosting	Alta precisión, permite ajustar modelos complejos y reducir el error de forma progresiva	Mayor riesgo de sobreajuste y necesidad de ajustar varios hiperparámetros

Cuadro 1: Comparativa de modelos estadísticos y de *machine learning* empleados

Estas diversas metodologías no solo ayudan a encontrar un modelo más robusto, sino también a identificar patrones recurrentes de las diferentes variables en las distintas zonas, las cuales condicionan la presencia de la especie (Elith & Leathwick, 2009).

A continuación, vamos a hablar de los diferentes tipos de modelos que vamos a utilizar en este trabajo dentro de los estadísticos y los de machine learning anteriormente mencionados, así como de sus ventajas y limitaciones. Por un lado, dentro de los modelos estadísticos tenemos los modelos lineales generalizados y los aditivos generalizados, modelos con gran interpretabilidad basados en hipótesis y en el control de los errores, pero que asumen linealidad y pueden infraajustar. Por otro lado, dentro de los modelos de machine learning tenemos: el Random Forest (Breiman, 2001), un modelo robusto, no lineal y que gestiona interacción entre variables, aunque tiene menor interpretabilidad y a veces sus resultados son difíciles de explicar; y el Gradient Boosting (Friedman, 2001), un modelo muy preciso que permite ajustar modelos complejos y minimizar errores, aunque requiere más ajuste de los parámetros y tiene más riesgo de sobreajuste.

Cada uno de estos algoritmos aborda los datos desde una perspectiva distinta, usando varios, buscamos un equilibrio entre interpretabilidad y rendimiento del modelo.

Una vez asentadas las bases teóricas y metodológicas, en la siguiente seción se abordarán los datos empleados y el proceso de preprocesamiento necesario para su análisis.

3. Datos y preprocesamiento

Antes de desarrollar y aplicar estos modelos de distribución de especies, debemos trabajar primero con los datos que servirán como base para nuestro análisis, y eso es de lo que nos ocuparemos en esta sección.

En primer lugar, hablaremos de la delimitación de nuestra área geográfica de estudio, además de la aplicación sobre esta de una malla hexagonal que nos permite estructurar y agrupar nuestras observaciones, haciéndolas más fácilmente interpretables. Detallaremos también todas las fuentes de datos utilizadas para nuestras variables, tanto la de presencia del lobo como aquellas ambientales, antrópicas o de vegetación. Por último, asociaremos cada una de estas variables a los hexágonos de la malla y las transformaciones (feature engineering) que se han llevado a cabo en el preprocesamiento.

Dentro del apartado de feature engineering, hablaremos sobre una de las decisiones metodológicas aplicadas en este trabajo: la construcción de dos escenarios de modelado. En uno de ellos se tendrá en cuenta el porcentaje de vecinos adyacentes con presencia del lobo, y en el otro no. Esto se hace porque, si nuestro objetivo es predecir la presencia del lobo en zonas donde conocemos su presencia en áreas cercanas, esta se convierte en una variable muy relevante que debemos utilizar (un ejemplo sería un pastor que quiere mover sus ovejas por otra zona donde aún no se conoce información directa, pero sí la de su entorno circundante). Sin embargo, hay muchos casos donde no se conoce la presencia en zonas vecinas, en cuyo caso debemos contar con un modelo que no dependa de esa información.

3.1. Área de estudio

El área de estudio utilizada para este trabajo es la Península Ibérica, es decir, España y Portugal. En un inicio se pensó trabajar solo con España, pero debido a la notable presencia del lobo en Portugal, no tenía sentido delimitar el territorio únicamente por una frontera administrativa.

A pesar de no ser un territorio especialmente extenso, la Península Ibérica representa un enclave ecológico de gran importancia, debido a su elevada biodiversidad y a su heterogeneidad climática y altitudinal. Presenta además una amplia representación de ecosistemas mediterráneos y atlánticos, reflejada incluso en su nombre, en alusión a las dos principales masas de agua que la rodean. Esta diversidad de hábitats la convierte en un territorio clave para la coexistencia de miles de especies diferentes, entre ellas el lobo ibérico, el cual lleva su nombre precisamente por su presencia única en esta región. Por todo ello, se ha seleccionado este área de estudio, en coherencia con la disponibilidad de datos ambientales y biológicos.

A la hora de procesar y modelar los datos en RStudio, se ha trabajado sobre el mapa de la Península; la geometría base utilizada para definirla (el contorno de España y Portugal) fue obtenida a través del paquete `rnaturalearth` en R, el cual accede a datos geográficos del proyecto Natural Earth. En el mapa se excluyeron las Islas Canarias debido a la ausencia de presencia del lobo en esta zona, mientras que se mantuvieron las Islas Baleares por cuestiones técnicas, a pesar de que tampoco se registra presencia en estas islas.

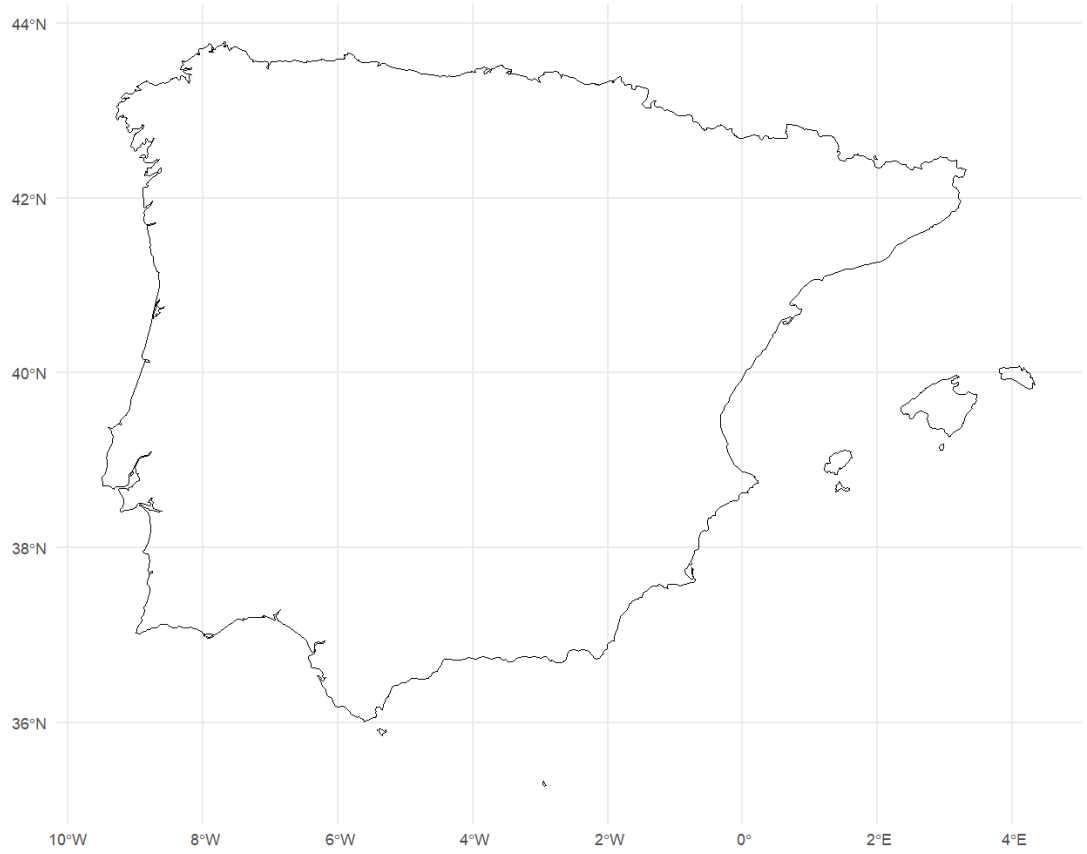

Figura 4: Contorno geográfico de la Península Ibérica. Fuente: Elaboración propia.

3.2. División en hexágonos

Una vez definida el área de estudio, ahora nos preguntamos cómo debemos tratar con nuestros datos en este espacio. Tratar con ellos individualmente sería tedioso y poco eficiente, sobre todo computacionalmente, lo que nos lleva a pensar en una forma de dividir el espacio, para la cual necesitamos una unidad espacial regular y neutra que agregue datos y facilite el posterior análisis espacial y la modelización (Guisan & Zimmermann, 2000). En este contexto surge la búsqueda de la forma espacial óptima.

Podríamos inicialmente pensar en dividirla en una cuadrícula, pero con las cuadrículas surge un problema: cada celda tendría 4 vecinos por lado y, a su vez, cuatro vecinos con los que comparte vértice. En total tenemos 8 vecinos, pero no todos tienen la misma relación geométrica; los de lado tienen conexión real (adyacencia), mientras que los de vértice (diagonales) no comparten borde, solo punto, lo que puede llevar a ambigüedad topológica.

Esto se resuelve a partir de un problema de optimización muy antiguo, conocido como la conjetura del panal, la cual fue finalmente demostrada por el matemático Thomas C. Hales en 1999 y publicada en 2001. En su demostración, Hales estableció que: Çualquier partición del plano en regiones de igual área tiene un perímetro total al menos igual al de la teselación hexagonal regular.", resultado que confirma matemáticamente lo que, por ejemplo, las abejas ya aplicaban en la naturaleza: la eficiencia del patrón hexagonal.

En una malla de hexágonos, cada celda tiene exactamente 6 vecinos, todos conectados por lado, no por vértice. En este caso no hay vecinos ambiguos, conectados solo por ün punto", como sí ocurre en cuadrados. Otra ventaja es que la distancia entre centros

de celdas vecinas es constante, la conectividad se vuelve mucho más uniforme y, como consecuencia, tenemos transiciones espaciales más suaves (Birch et al., 2007).

Aplicando esto a nuestro trabajo, hemos dividido el mapa de la Península Ibérica en hexágonos, buscando esta optimización del espacio.

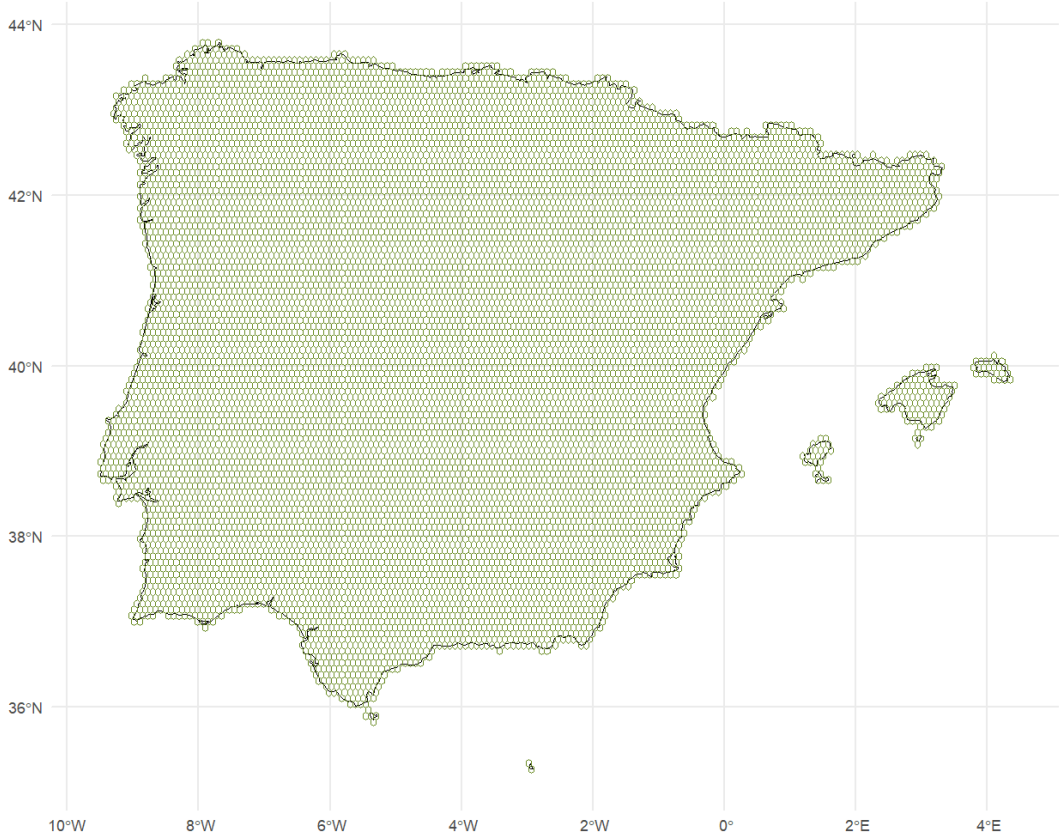

Figura 5: División de la península en hexágonos. Fuente: Elaboración propia.

La malla de hexágonos regulares se creó mediante la función `st_make_grid()` del paquete `sf`, definiendo una resolución espacial uniforme. Posteriormente, se filtraron aquellos hexágonos que no intersectaban con el contorno peninsular y se les asignó un identificador único (`hex_id`) para su posterior integración con los datos de presencia del lobo.

Se seleccionó un tamaño de celda de 0.08 grados para los hexágonos, lo cual representa aproximadamente 50 km^2 por celda. Esta resolución fue elegida como un equilibrio entre nivel de detalle espacial y eficiencia computacional. Es lo suficientemente pequeña como para captar variaciones ambientales y patrones de presencia del lobo, pero lo bastante grande como para reducir el ruido local y mantener la eficiencia en el procesamiento de los datos.

3.3. Fuentes de datos

En la siguiente sección hablaremos sobre las diferentes fuentes de datos utilizadas en este trabajo, desde el conjunto de datos de presencia del lobo, hasta las variables climáticas, topográficas, atrópicas y de vegetación, que servirán como las predictoras de nuestro modelo.

Cada conjunto de datos ha sido seleccionado en función de su disponibilidad, resolución espacial y relevancia ecológica. Además, se explica cómo estas capas de información han sido integradas y adaptadas a la malla hexagonal construida y explicada anteriormente.

3.3.1. Presencia/ausencia del lobo

Los datos de presencia del lobo ibérico (*Canis lupus*) fueron obtenidos a través de la plataforma GBIF, utilizando el paquete `rgbif` en R. Se extrajeron observaciones, tanto para España como para Portugal, mediante la función `occ_data()`, especificando el nombre científico y el país correspondiente.

Posteriormente, se combinaron los datos recogidos de ambos países y se filtraron aquellos que se colaban en nuestra muestra erróneamente, como por ejemplo registros correspondientes a perros domésticos (Canis lupus familiaris), a través del nombre científico.

Una vez hecha esta pequeña depuración, los datos se transformaron a objeto espacial del tipo sf, utilizando las coordenadas geográficas y transformándolos al sistema EPSG:4326. Esto permite garantizar la coherencia espacial de todos los datos, especialmente si se integran con otras fuentes que pueden usar sistemas distintos. En este caso, se conservaron únicamente las coordenadas en grados decimales (latitud y longitud), ya que es el formato más estándar y de fácil utilización en mapas interactivos, y el más apropiado para los cálculos espaciales posteriores. Solo se conservaron las columnas necesarias para el análisis, como el identificador único de cada observación y la geometría (coordenadas de cada lobo).

Finalmente, se procedió a la asociación de la presencia de lobos a los hexágonos. Esto se hizo utilizando una unión espacial entre los puntos de observación del lobo y los hexágonos. Esta operación asigna a cada punto de presencia el identificador del hexágono en el que se encuentra. Con esta información, se realizó un conteo de individuos por hexágono y se unió este conteo a la capa original de hexágonos. Por último, se creó una nueva variable binaria presencia, con valor 1 si se observó al menos un individuo de lobo en el hexágono, y 0 en caso contrario.

A continuación, graficamos de nuevo la malla de hexágonos anterior, tiñendo de negro aquellos hexágonos donde hay presencia (presencia = 1).

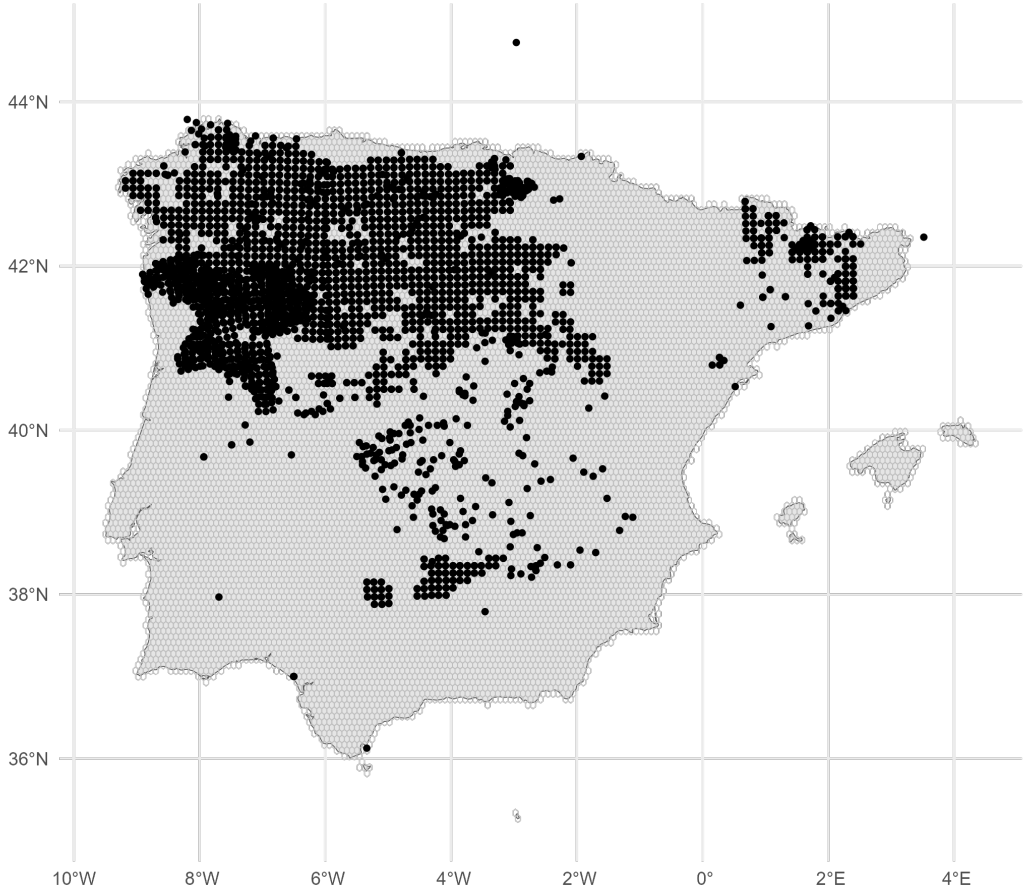

Figura 6: Distribución del lobo ibérico en la Península Ibérica.

Fuente: Elaboración propia.

3.3.2. Variables ambientales y topográficas

En primer lugar, tenemos las variables ambientales o climáticas, en este caso extraídas del conjunto de datos bioclimáticos de WorldClim (versión 2.1), a una resolución espacial de 2.5 minutos (aproximadamente 4.5 km). De nuevo, al igual que con los hexágonos, la elección del tamaño se ha hecho de forma que sea lo suficientemente pequeña como para captar variaciones ambientales y patrones de presencia del lobo, pero lo bastante grande como para reducir el ruido local y mantener la eficiencia en el procesamiento de los datos. Estas variables representan promedios climáticos del periodo 1970–2000 y se presentan en formato raster georreferenciado.

Esta página tiene una gran cantidad de variables, de las cuales se ha hecho una preselección, dejando fuera algunas que a priori sabemos que son indiferentes para nuestro trabajo. Las variables seleccionadas son consideradas relevantes para modelar la distribución del lobo ibérico, ya que influyen en su fisiología, comportamiento, disponibilidad de presas y cobertura vegetal. En la literatura, es habitual utilizar BIO1 y BIO12 como variables base para modelos de distribución de grandes carnívoros; por ejemplo, Lesmerises et al. (2012) usaron variables como la temperatura mínima del invierno y la precipitación anual para modelar la distribución del lobo en zonas boreales, mostrando su relevancia como restricciones climáticas sobre la presencia de la especie.

En segundo lugar, de la misma página pero en un dataset diferente, hemos descargado la elevación o altitud. Esta fue incluida como variable predictora debido a su fuerte relación con múltiples factores ecológicos relevantes. Afecta directamente a parámetros climáticos como la temperatura y la humedad, y de forma indirecta a la vegetación, la accesibilidad y la distribución de presas. Además, la altitud puede limitar la presencia del lobo en zonas montañosas por debajo o por encima de ciertos umbrales, influyendo así en sus patrones de distribución y desplazamiento. En la literatura, la elevación se considera una variable clave en modelos de distribución de grandes carnívoros, ya que contribuye a explicar gradientes ambientales importantes en el paisaje; por ejemplo, Said et al. (2016) encontraron que la elevación influía significativamente en la distribución del lobo ibérico, afectando su selección de hábitat y conectividad entre poblaciones en la Península Ibérica.

En tercer y último lugar, de nuevo de la misma página obtuvimos la variable velocidad del viento. La inclusión de esta variable en estudios acerca de la distribución del lobo es poco común: sin embargo, unos pocos estudios sí han explorado su relevancia en contextos específicos: la comunicación y la percepción del entorno.

El principal método de comunicación de los lobos, especialmente a distancia, es a través de los aullidos. Es lógico pensar que, si la velocidad con la que viaja el sonido se ve afectada por el viento, la comunicación entre miembros de la manada podría verse alterada, en cuyo caso el viento muy elevado o grandes variaciones de viento a lo largo del año podrían actuar como una barrera ecológica, reduciendo la idoneidad del entorno y afectando, por tanto, su distribución espacial. En esa línea, Ausband et al. (2020) encontraron que los lobos eran menos propensos a responder a aullidos simulados en condiciones de viento fuerte, lo que sugiere que el viento puede interferir en la comunicación acústica entre individuos. Por tanto, aunque no se trate de una variable ampliamente estudiada, su inclusión en este trabajo se justifica con motivos exploratorios.

Finalmente, una observación relevante en este punto es que, dado que la resolución espacial es de aproximadamente 4.5 km, puede haber más de un valor bioclimático dentro de una misma celda. Efectivamente, esto ocurre, del mismo modo que puede haber más de una observación de lobo por hexágono en numerosas unidades.

Para asociar a cada celda hexagonal las variables climáticas y de viento, se emplearon operaciones de extracción espacial sobre capas ráster, utilizando funciones de resumen.

En el caso de las variables climáticas (como la temperatura máxima del mes más cálido, BIO5), se aplicó una función de extracción de media sobre cada hexágono. Esto permitió asignar a cada celda un valor promedio representativo del área cubierta por dicha celda. Finalmente esta variable fue añadida al dataset de hexágonos como una columna más.

Para la variable viento se utilizó un procedimeinto ligeramente diferente, ya que provenian de capas mensuales. Se trabajó con múltiples rásters mensuales de velocidad, sobre los que se calcularon dos evariables diferentes, una que recogía la media por hexágono y otra que calculaba la desviación típica (para una posterior utilización en el feature engineering).

A continuación, una tabla resumen de todas ellas:

Variable	Descripción ecológica
BIO1	Temperatura media anual. Representa el nivel térmico general del área de estudio.
BIO4	Estacionalidad de la temperatura (desviación estándar ×100). Mide la variabilidad térmica anual, relevante para especies sensibles al cambio estacional.
BIO5	Temperatura máxima del mes más cálido. Indica la tolerancia térmica máxima del hábitat.
BIO6	Temperatura mínima del mes más frío. Refleja los límites térmicos inferiores, críticos para la supervivencia en invierno.
BIO12	Precipitación anual total. Estima la disponibilidad hídrica, fundamental para ecosistemas y presas del lobo.
Velocidad del viento	Velocidad media anual del viento. Puede influir en la sensación térmica, la dispersión de olores, la comunicación y el movimiento de presas.
viento_sd	Variabilidad estacional del viento (desviación típica mensual). Refleja el grado de fluctuación intermensual de la velocidad del viento, lo cual podría influir en la comunicación o en la percepción del entorno por parte del lobo.
Elevación	Altitud del terreno. Afecta directamente a la temperatura, humedad, vegetación disponible y distribución de especies.

Cuadro 2: Variables climáticas y ambientales seleccionadas y su relevancia ecológica en el modelado de la distribución del lobo ibérico.

3.3.3. Vegetación y proporción de bosque

Uno de los factores clave en el hábitat del lobo ibérico es la vegetación, especialmente la cobertura forestal, aportando refugio, áreas de descanso, escondite a la hora de cazar o incluso protección frente a perturbaciones humanas. Numerosos estudios han demostrado que los lobos prefieren hábitats con mayor cobertura forestal, especialmente en zonas donde la presión humana es elevada. Sin embargo, no es solo gran cobertura lo que buscan, también la heterogeneidad del terreno.

Se exploraron varias formas de incluir esta variable y finalmente se decidió trabajar con el porcentaje de bosque en cada hexágono (número de píxeles forestales sobre total por celda). Se consideraron como "bosque"las clases correspondientes a masas forestales continuas o mixtas.

Los datos de cobertura del suelo se obtuvieron a partir del producto raster del programa CORINE Land Cover (CLC) 2018, en su versión validada de 2020, con una resolución espacial de 100 metros. Esta base de datos, elaborada por la Agencia Europea de Medio Ambiente (EEA) en el marco del programa Copernicus, proporciona una clasificación temática detallada del uso del suelo en toda Europa. En particular, se utilizaron las clases

correspondientes a masas forestales (códigos 23, 24 y 25) para identificar zonas boscosas.

Código CLC	**Tipo de cobertura**	**Descripción**
23	Bosque de frondosas	Bosques dominados por árboles de hoja ancha, como robles o hayas.
24	Bosque de coníferas	Bosques dominados por coníferas, como pinos o abetos.
25	Bosque mixto	Bosques con mezcla significativa de frondosas y coníferas.

Cuadro 3: Tipos de bosque seleccionados a partir de CORINE Land Cover (CLC).

Esta información fue reproyectada al sistema de coordenadas EPSG:4326, recortada al área de estudio y utilizada para calcular, mediante técnicas de extracción espacial, la proporción de superficie forestal contenida en cada hexágono del análisis. Esta información se unió finalmente a una nueva columna a nuestro dataset de hexágonos, conteniendo un número de 0 a 1 que representaría el porcentaje de bosque.

3.3.4. Variables antrópicas: población y accesibilidad

Uno de los grandes condicionantes de la presencia del lobo ibérico es la actividad humana. El lobo evita, en la medida de lo posible, las zonas con alta densidad poblacional, ya que estas suelen implicar mayor riesgo de mortalidad, menor disponibilidad de refugio y mayor fragmentación del hábitat. Por ello, las variables antrópicas son fundamentales para capturar estos efectos en los modelos de distribución.

Hay diferentes formas de tener en cuenta la actividad humana. En este trabajo se han incorporado dos tipos. Por un lado, la densidad de población: se utilizaron datos de población para el año 2020 procedentes del proyecto *WorldPop*, que proporcionan estimaciones de población por píxel asociadas a unidades administrativas oficiales. Se descargaron los conjuntos asociados a España y Portugal, y se combinaron mediante una operación mosaico para construir una capa raster continua sobre la Península Ibérica. A partir de esta capa, se extrajo el número total de habitantes por hexágono, y posteriormente se dividió por el área de cada hexágono (misma área para todos) para obtener la densidad de población (habitantes por km^2).

Sin embargo, no siempre la densidad de población es suficiente para explicar la ausencia del lobo u otras especies salvajes. Puede darse el caso de que una zona tenga una ciudad densamente poblada, pero el entorno próximo esté poco habitado; o lo contrario: una zona dispersa con muchos pueblos pequeños y conectados. Ambos escenarios pueden influir en la presencia del lobo, aunque la densidad media sea similar.

Para solventar esto, se ha creado una nueva variable denominada *accesibilidad*. Esta se construyó a partir de una matriz de distancias entre el centroide de cada hexágono y los centroides de aquellos que presentaban una población superior a 500 habitantes, considerados como núcleos poblados. La accesibilidad se estimó mediante una suma de distancias inversas ponderadas, de forma que los pueblos más cercanos aportan mayor

peso que los más lejanos. Esta variable no solo tiene en cuenta los asentamientos humanos cercanos, sino también su proximidad relativa, lo que puede influir en la ocupación del lobo ibérico.

La accesibilidad A_i de un hexágono i se calculó como:

$$A_i = \sum_{j=1}^{n} \frac{1}{d_{ij} + 1}$$

donde:

- A_i: accesibilidad del hexágono i,
- d_{ij}: distancia entre el centroide del hexágono i y el centroide del núcleo poblado j,
- n: número total de núcleos poblados considerados,
- Se suma 1 al denominador para evitar divisiones por cero en caso de coincidencia exacta de ubicaciones.

3.4. Porcentaje de hexágonos adyacentes con presencia de lobo

Una de las hipótesis clave de este trabajo es que la distribución espacial del lobo ibérico puede presentar autocorrelación geográfica, es decir, que la presencia de la especie en un hexágono está influenciada por la presencia de la especie en los hexágonos adyacentes o vecinos. Este fenómeno es común en especies territoriales o sociales, donde la ocupación del espacio no es aleatoria.

Para tener en cuenta esta dependencia espacial, se construyó una nueva variable que representa el porcentaje de hexágonos vecinos con presencia de lobo, es decir, de los 6 vecinos que tiene (con los que comparte cara), cuántos presentan efectivamente presencia de lobo. Con esta variable se pretende entrenar un segundo modelo (Modelo 2), que tenga en cuenta la correlación espacial como otro factor explicativo.

Teniendo ya el centroide de todos los hexágonos, se construyó una matriz de distancias geodésicas (basada en la fórmula de Haversine) entre todos los centroides. Para cada hexágono, se identificaron los 6 más cercanos, excluyendo a sí mismo. Sobre esos vecinos, se calculó el porcentaje de ellos que tenían presencia de lobo.

La fórmula utilizada fue:

$$\text{Porcentaje}_i = \frac{1}{6} \sum_{j \in \text{vecinos}(i)} \text{presencia}_j \times 100$$

donde $\text{presencia}_j \in \{0, 1\}$

3.5. Feature engineering

El proceso de creación de nuevas variables a partir de las existentes, conocido como feature engineering, es una práctica habitual en ciencia de datos y modelización predictiva [32]. Este proceso permite capturar estructuras ocultas en los datos, reducir ruido y mejorar el rendimiento de los modelos.

Además de las variables originales derivadas de fuentes climáticas, topográficas y antrópicas, se construyeron dos nuevas variables. Por un lado, se creó una variable que resume de forma integrada tres factores asociados a la idoneidad del hábitat para el lobo ibérico: mayor proporción de bosque, menor densidad de población y menor accesibilidad (mayor distancia a núcleos urbanos). Todos estos componentes se estandarizaron mediante la siguiente fórmula:

$$\text{Índice}_{\text{hábitat}} = z(\text{porc\_bosque}) - z(\text{densidad}) - z(\text{accesibilidad})$$

Ha sido explicado ya el motivo de la inclusión de la variable velocidad del viento, sin embargo, nos surge otra cuestión. Puede que altas o bajas velocidades de viento puedan influir en la comunicación del lobo y, por ende, en su presencia, pero también podríamos plantear la hipótesis de que lo que realmente perjudicaría a su comunicación fueran cambios grandes en esta velocidad. Es posible que puedan adecuarse a vientos altos o vientos bajos, mientras estos sean estables. La variabilidad relativa mide la desviación en función del valor medio. Por este motivo, se ha creado la variable "variabilidad relativa del viento", que se ha usado en ecología como proxy de exposición, apertura del terreno y estrés ambiental.

$$\text{Viento}_{\text{relativo}} = \frac{\text{viento\_sd}}{\text{viento\_mean} + 0{,}1}$$

Por último, hemos creado la variable *presión humana*. La densidad por sí sola no siempre refleja la presión real: puede haber zonas poco pobladas pero muy accesibles, como por ejemplo parques o áreas rurales conectadas. Lo que ocurre con la accesibilidad es que, si una zona está conectada por caminos o cerca de varios núcleos urbanos, el riesgo de alteración es mayor. La combinación $\log(\text{población}+1) \times \text{accesibilidad}$ permite modelar el efecto sin alterarse excesivamente por valores extremos y dándole mayor importancia a zonas que, sin ser urbanas, pueden estar muy ocupadas o utilizadas.

Esta variable busca fusionar en una sola la cantidad de población humana total en un hexágono y cuán fácil es acceder a ese hexágono desde pueblos cercanos.

$$\text{Presión}_{\text{humana}} = z\left(\log(\text{población} + 1)\right) \times z(\text{accesibilidad})$$

Con esto cerramos la recolección completa de variables y procedemos en el apartado siguiente a la selección.

4. Selección de variables

Una vez hayamos recopilado todas las variables que a priori consideramos interesantes o buenas predictoras de la presencia del lobo, nos encontramos ante un conjunto de datos con muchas variables. Si todas ellas fueran incluidas en nuestro modelo final, podrían generarse modelos difíciles de interpretar y excesivamente complejos.

Uno de los principales retos al construir modelos predictivos es encontrar el equilibrio entre la complejidad del modelo y su capacidad de generalización. Este equilibrio se entiende y se explica en términos de sesgo y varianza (James et al., 2013).

El sesgo mide el error sistemático que se produce cuando un modelo es demasiado simple para captar la relación real entre las variables. Un modelo con alto sesgo tiende a hacer suposiciones fuertes sobre los datos, lo que puede dar lugar a predicciones alejadas de los valores reales.

Por otro lado, la varianza refleja la sensibilidad del modelo a los datos de entrenamiento. Es decir, un modelo con alta varianza se ajusta demasiado a los datos (incluyendo el ruido), y cambia drásticamente cuando cambian los datos. Esto da lugar a modelos con una mala capacidad de generalización.

Nuestro objetivo principal en este apartado es seleccionar conjuntos de variables que mejoren la interpretabilidad del modelo, reduzcan el sobreajuste (cuando un modelo tiene bajo sesgo pero alta varianza) y minimicen el error de predicción.

Esta selección de variables se llevará a cabo mediante la aplicación de distintas técnicas (siguiendo diferentes criterios y lógicas), las cuales compararemos más adelante, con el fin de identificar aquellas que permitan reducir el error de predicción.

Las técnicas o criterios de selección que utilizaremos serán: AIC, BIC, K, RFE, MMPC y un modelo inicial de Random Forest, los cuales explicaremos con más detalle más adelante.

Realizaremos primero una exploración descriptiva y un análisis de correlación de todas las variables disponibles. Cabe destacar que, dado que se entrenarán modelos con y sin información espacial, la selección de variables se realizará de forma paralela en ambos escenarios.

4.1. Análisis exploratorio

4.1.1. Estructura general del conjunto de datos

El primer paso del análisis exploratorio se basa en revisar la estructura y las características básicas de nuetro conjunto de datos. Esto nos permite conocer el número de observaciones y variables, así como los tipos de datos asociados a cada columna, lo cual es fundamental para detectar posibles errores, variables mal formateadas o valores raros o atípicos.

Obtuvimos la siguiente información:

- El dataset está compuesto por **11.605 observaciones** (hexágonos) y **14 variables**.
- La variable dependiente es `presencia`, un factor con dos niveles: “Yes” y “No”. Tenemos 9706 ausencias 1899 presencias.
- Las otras 13 variables son predictoras numéricas, y recogen información ambiental, antrópica y topográfica. Algunas han sido generadas a través de *feature engineering*.
- A través del resumen estadístico , se observaron escalas muy distintas entre variables (por ejemplo, `densidad` frente a `porc_bosque`), que en principio no deberían interferir, pero hay que tenerlas en cuenta por si fuera necesaria una estandarización para ciertos modelos.

- Pot último, se ha realizado una búsqueda de valores perdidos (NA) que ha dado como resultado la ausencia de este tipo de valores en todas las variables y observaciones. Por tanto, no hay que realizar ningún ajuste o eliminación en nuetsra base de datos.

Este paso nos permite garantizar que los datos estén correctamente preparados para el análisis posterior.

4.1.2. Histogramas y mapas de las variables predictoras

Con el objetivo de analizar la distribución de las variables predictoras, se han representado histogramas individuales. Esta visualización permite detectar posibles asimetrías, valores extremos o escalas muy distintas entre variables, aspectos clave antes de aplicar modelos estadísticos o de aprendizaje automático. A continuación, se presentan los histogramas, comenzando por la variable *densidad*.

4.1.3. Distribución de la variable *densidad*

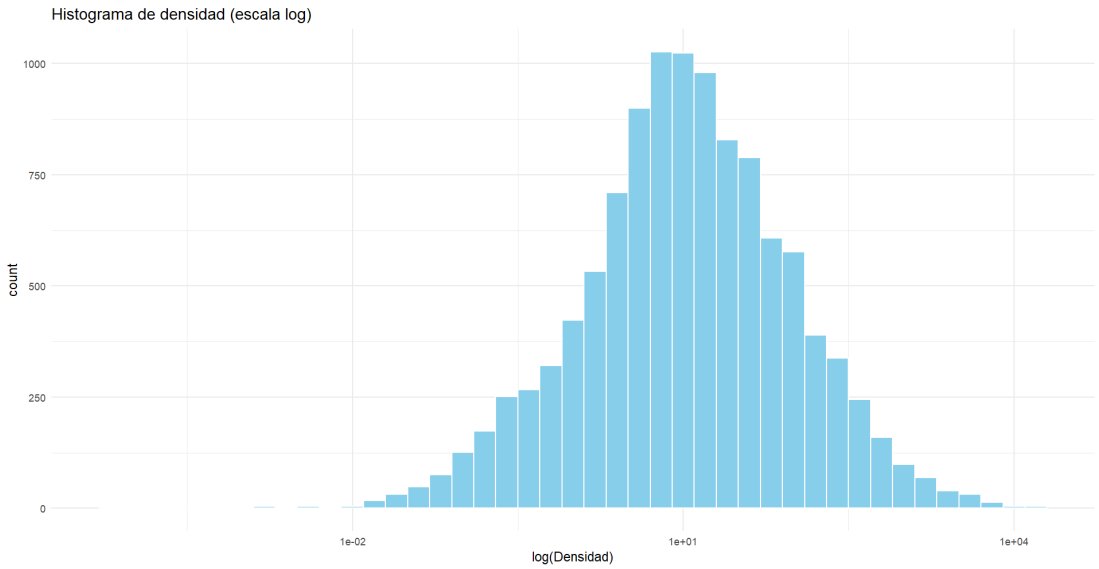

Figura 7: Histograma de la densidad de población por hexágono (escala logarítmica). Fuente: elaboración propia.

En su escala original, la variable *densidad* presentaba una distribución altamente sesgada a la derecha, con valores extremos en algunos hexágonos con gran concentración urbana. Esta distribución asimétrica dificultaba el análisis comparativo y podía introducir inestabilidad en ciertos modelos. Para corregir esto, se aplicó una transformación logarítmica (con $\log(\text{densidad} + 1)$), esto permitió representar los datos en una escala más manejable.

La mayor parte de los hexágonos tienen una densidad de población media de entre 4 y 20 (transformado desde escala logaritma para ser interpretado) habitantes por km^2.

Este resultado es coherente, ya que zonas con alta densidad urbana son escasas en la Península Ibérica y están concentradas en unos pocos hexágonos extremos, los cuales fueron precisamente la razón para aplicar la transformación logarítmica y así mejorar la representación gráfica y la simetría de la distribución.

4.1.4. Distribución de la variable *bio1* (temperatura media anual)

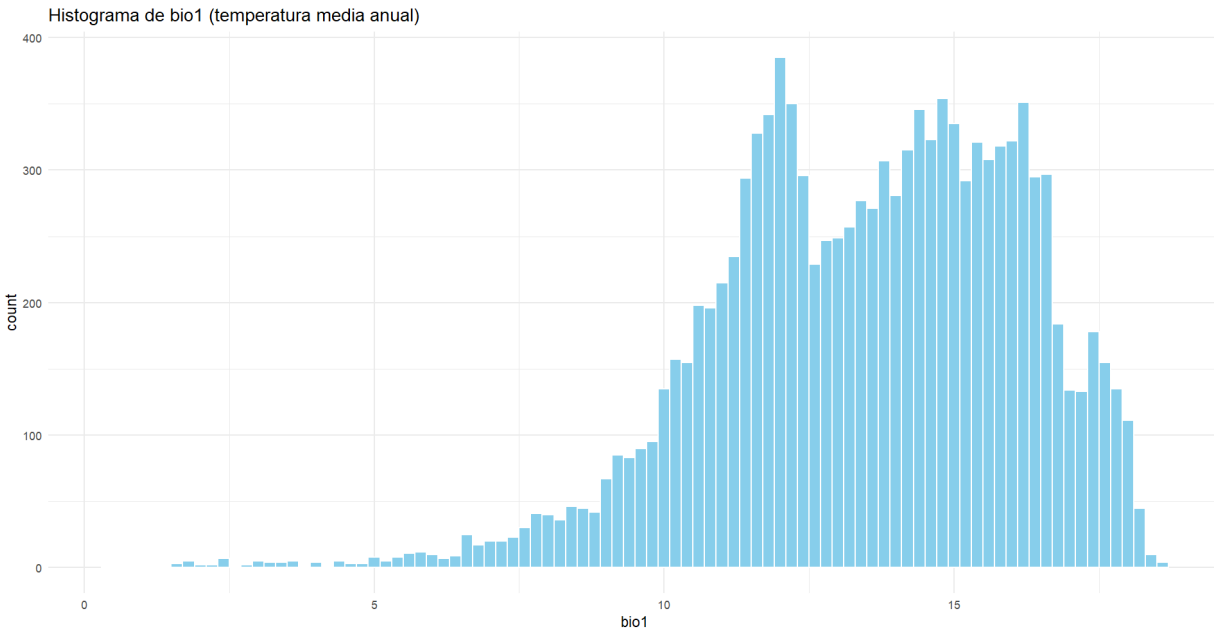

Figura 8: Histograma de la temperatura media anual por hexágono. Fuente: elaboración propia.

En este histograma observamos una distribución poco simétrica, con una notable acumulación hacia valores medio altos (temperaturas de entre 12°C y 16°C grados), es decir, gran parte del territorio presenta temperaturas templadas. Se observa un pico claro alrededor de los 13–14°C, lo que podría coincidir con zonas de clima mediterráneo predominante, donde la temperatura nunca se va a los extremos.

Hay pocos hexágonos con temperaturas medias anuales por debajo de 8°C, lo que probablemente corresponde a zonas del norte o de alta montaña. Las temperaturas más altas (mayores de 17°C) son menos frecuentes, posiblemente corresponden a zonas de la costa sur y sureste. Se puede intuir que la altitud podría ser un factor explicativo de esta dispersión de la temperatura, dado que las temperaturas más bajas están mas focalizadas.

El hecho de que haya una distribución más o menos continua nos indica que la cobertura de hexágonos es uniforme y que no hay grandes cantidades de datos perdidos.

A continuación, se muestra el mapa anterior de la malla de hexágonos coloreado según la temperatura anual media:

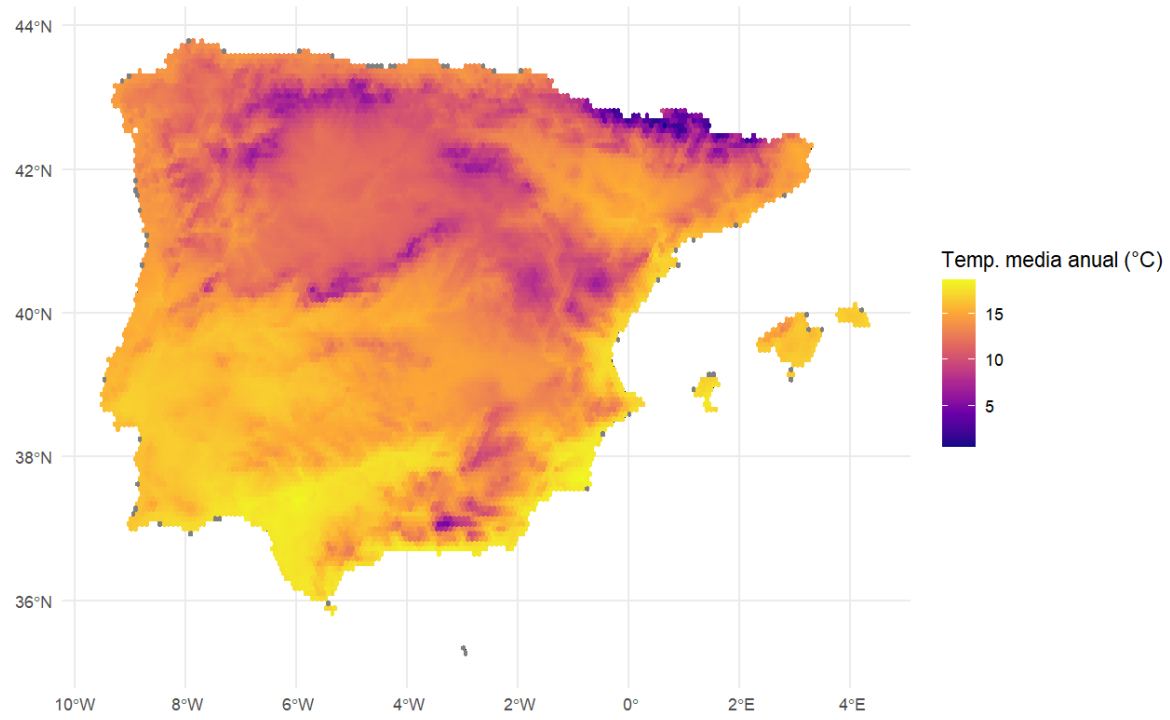

Figura 9: Mapa de la temperatura media anual por hexágono. Fuente: elaboración propia.

Con este mapa se comprueba lo observado en el histograma anterior. Vemos una clara diferencia entre noroeste y sureste, teniendo este último una temperatura media más elevada; y se comprueba tamién la influencia de la elevación, temperaturas más bajas en las zonas de Picos de Europa, Sierra Nevada y Pirineos. Al ser una península, la diferencia principal sin duda se observa en las diferencias entre zonas de costa y de interior.

4.1.5. Distribución de la variable *bio4* (Estacionalidad de la temperatura)

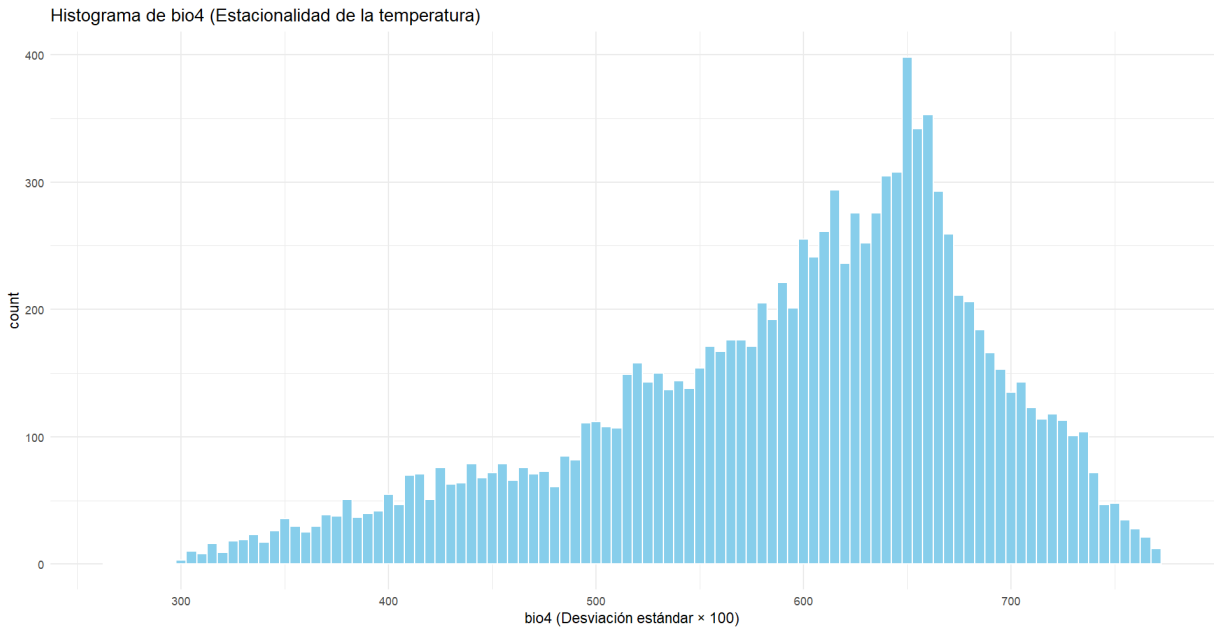

Figura 10: Histograma de la estacionalidad anual media por hexágono. Fuente: elaboración propia.

Observamos de nuevo una distribución asimétrica hacia la derecha, con mayor frecuencia de hexágonos con estacionalidad entre 500 y 600 (equivalente a 5–6.5°C de desviación estándar mensual). Observamos un pico muy marcado a la altura aproximada de 650, lo que indica que muchas áreas de hexágonos tienen una estacionalidad térmica al rededor de 6,5ºC. Aunque son menos frecuentes, existen hexágonos con valores inferiores a 400 y superiores a 700, indicando zonas con climas más suaves o más extremos respectivamente.

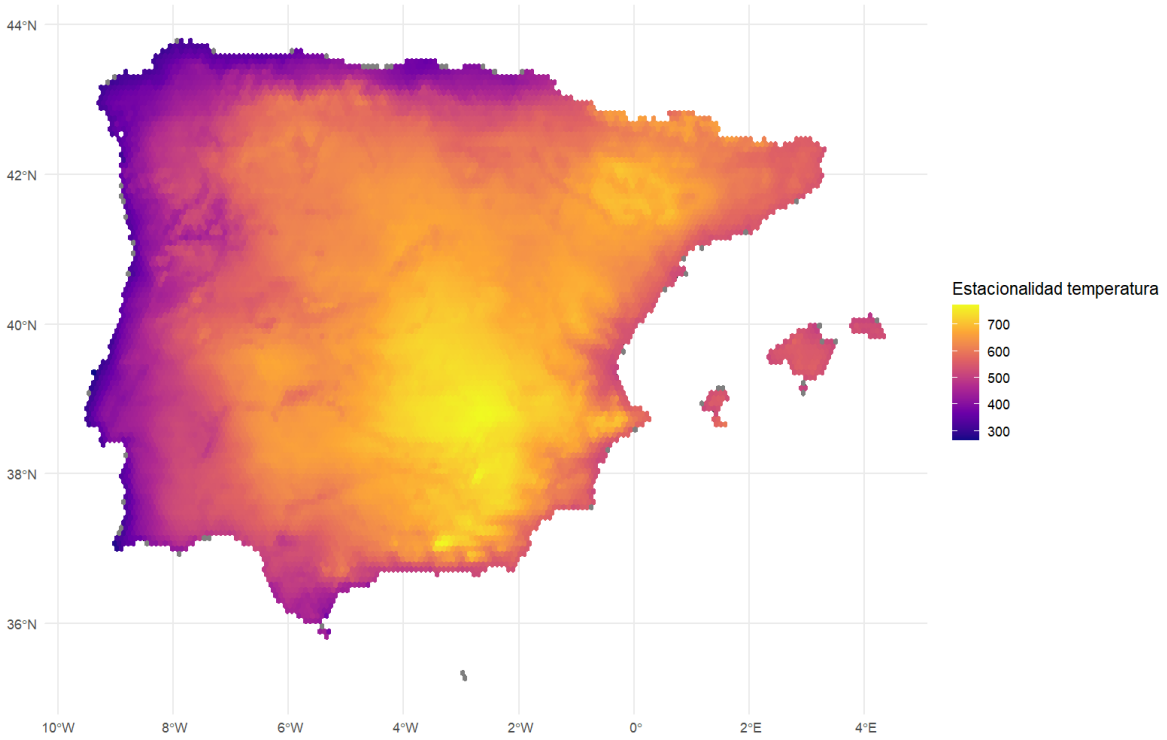

Figura 11: Mapa de la estacionalidad anual media por hexágono. Fuente: elaboración propia.

Con este mapa se comprueba la mayoria de hexágonos en torno a 6.5 y se observan otras apreciaciones. Vemos una clara diferencia entre la zona noroeste y la sureste, la costa atlántica tiene índices de estacionalidad muy bajos posiblemente por el efecto atenuador del mar(colores morados); y las zonas de interior, especialmente del sureste, muestran los valores más altos de estacionalidad (colores más amarillos), típicos de climas continentales.

4.1.6. Distribución de la variable *bio5* (Temperatura máxima del mes más calido)

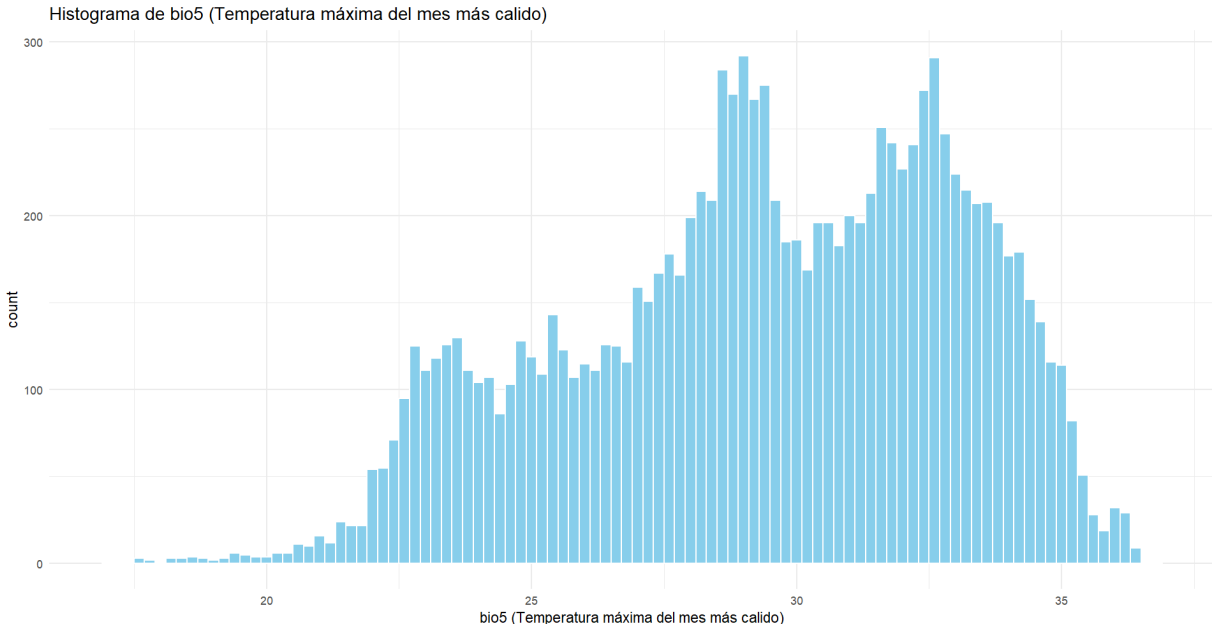

Figura 12: Histograma de la temperatura máxima del mes más calido. Fuente: elaboración propia.

De nuevo la misma observación que en los anteriores, asimetría hacia la derecha. Se observan dos picos muy claros, uno al rededor de los 28-29ºC y otro a los 33ºC. Estos dos picos representan de nuevo las dos grandes zonas representadas de España: el norte, y los lugares con gran elevación, cuya temperatura máxima del mes más calido no suele pasar de los 30 grados; y la meseta central, el sur y la costa mediterránea, cuyas temperaturas pueden llegar a grados muy altos en verano. Algunos hexágonos alcanzan temperaturas superiores a 40°C, lo que probablemente representa zonas del interior sur, como el valle del Guadalquivir o la meseta sur.

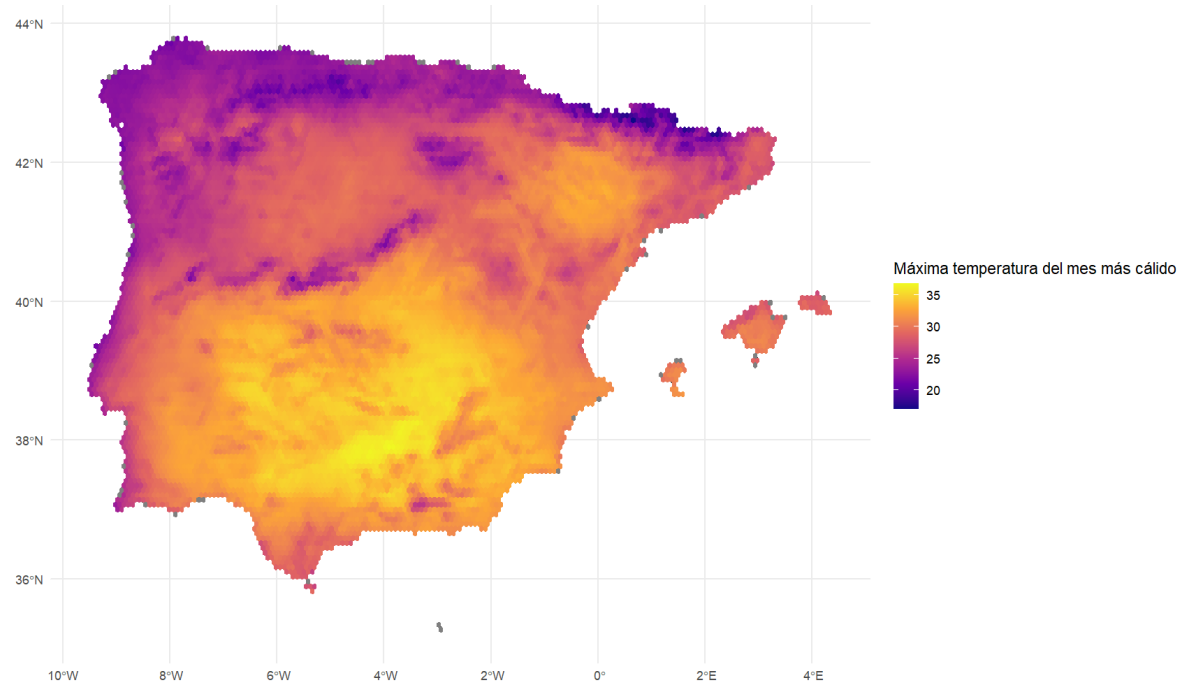

Figura 13: Mapa de la temperatura media máxima del mes más calido por hexágono. Fuente: elaboración propia.

Observamos en el mapa como el sur peninsular (Andalucía, Extremadura, Murcia) presenta los valores más elevados de temperatura máxima (colores amarillos intensos), superando los 40°C en algunas áreas. Las regiones ya antes mencionadas del Cantábrico, Galicia y zonas montañosas del norte muestran temperaturas máximas más suaves en verano (por debajo de 30°C), con tonos violetas y rosados. Se confirma también lo dicho de las zonas de mayor elevación (Pirineos, Sistema Central, Sierra Nevada); se observa una reducción clara de la temperatura máxima, debido a la influencia de la altitud.

4.1.7. Distribución de la variable *bio6* (Temperatura mínima del mes más frío)

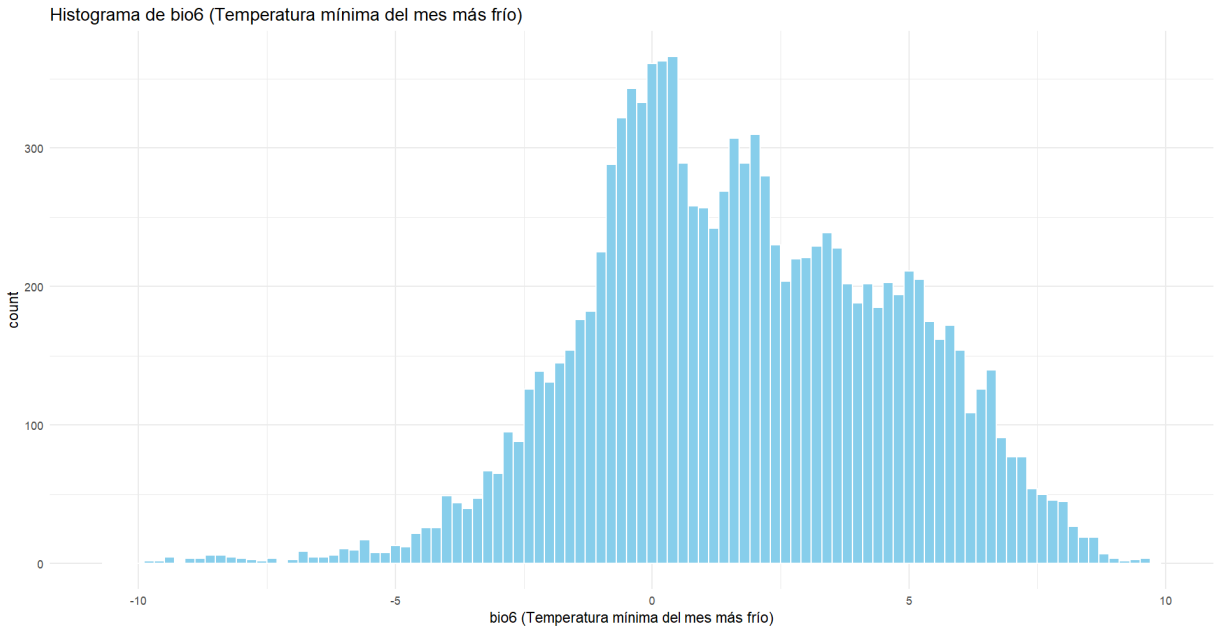

Figura 14: Histograma de la temperatura mínima del mes más frío. Fuente: elaboración propia.

En este histograma observamos un clara concentración entre los valores de 0º C y 5ºC, aunque también podemos observar una cantidad no despreciable de hexágonos con temperaturas por debajo de los 0ºC, probablemente correspondientes a zonas de alta montaña o zonas de interior. La mayor concentración se observa en los 0 ºC, lo que nos sugiere la presencia de inviernos bastante fríos, aunque no extremos, en buena parte de la península.

Mientras bio5 (máxima del mes más cálido) estaba más concentrado en valores altos, bio6 muestra una dispersión mayor hacia temperaturas más frías, lo cual resulta lógico.

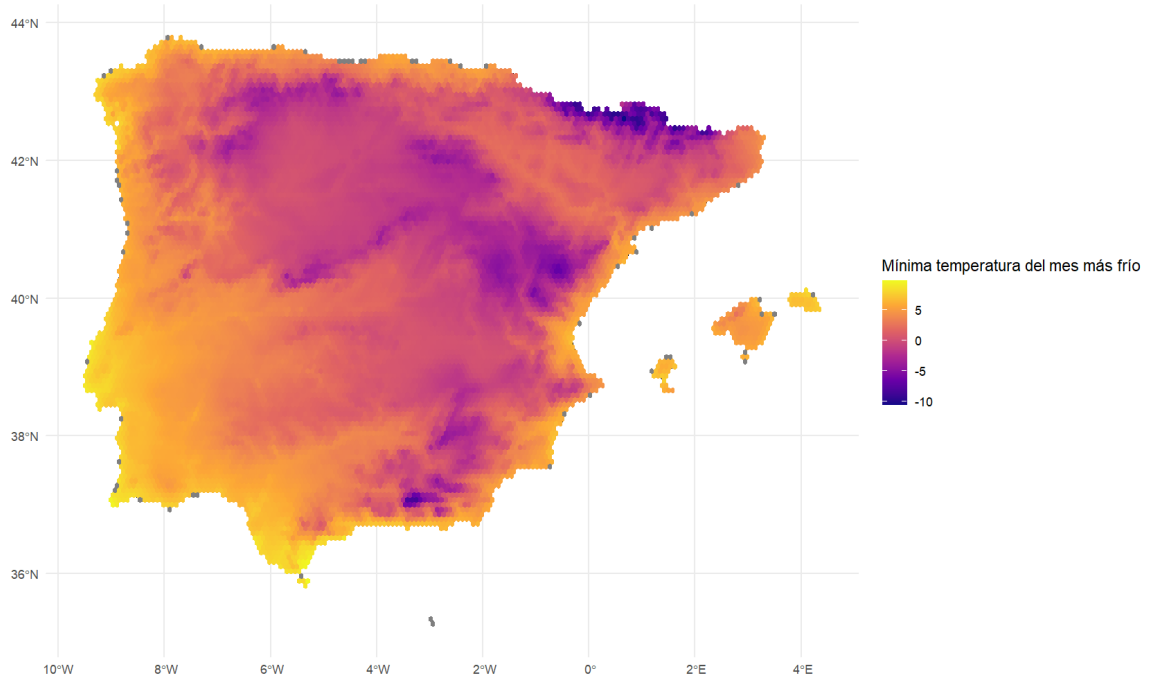

Figura 15: Mapa de la temperatura media mínima del mes más frío por hexágono. Fuente: elaboración propia.

Observamos en el mapa térmico como se confirma que son las zonas de interior y de alta montaña las que llegan a temperaturas más bajas, en contraste con las zonas costeras (especialmente la costa de Portugal), con temperaturas un poco más altas (entre los 4 y 6ºC). También se percibe el gradiente latitudinal, con el sur más templado que el norte. Cabe destacar que aquellas zonas donde vemos colores rosas oscuros o morados son aquellas donde la temperatura cae por debajo de los 0ºC, y por tanto podrían considerarse con riesgo climático de heladas invernales, lo que puede ser importante para especies sensibles o actividades agrícolas.

bio6 es una variable especialmente crítica en modelos de distribución de especies porque determina si una especie puede resistir las temperaturas invernales más extremas. Junto con bio5, define la amplitud térmica estacional y puede afectar la actividad y supervivencia de depredadores como el lobo ibérico.

4.1.8. Distribución de la variable *bio12* (Precipitación total anual)

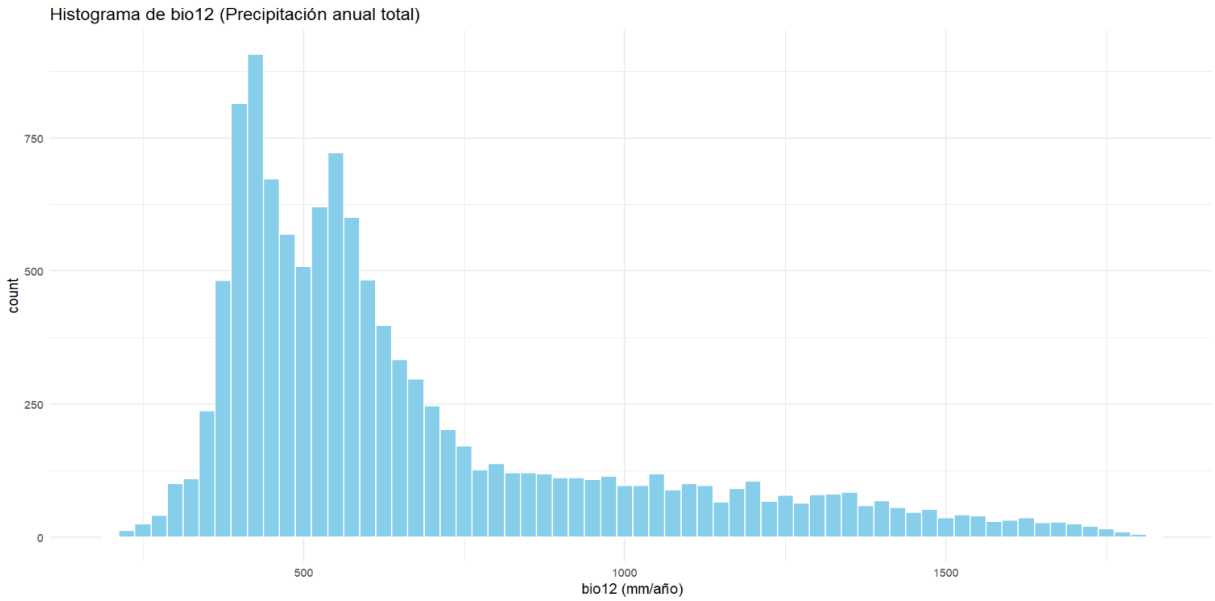

Figura 16: Histograma de la precipitación total anual. Fuente: elaboración propia.

La mayoría de los hexágonos tienen precipitaciones entre 400 y 800 mm anuales, con una caída progresiva hacia valores más altos. El valor más común ronda los 400 mm, lo que refleja que buena parte de la península tiene un régimen relativamente seco o moderado. Existen hexágonos con precipitaciones superiores a 1500 mm, aunque son poco frecuentes. Estos representan regiones muy concretas con alta pluviosidad.

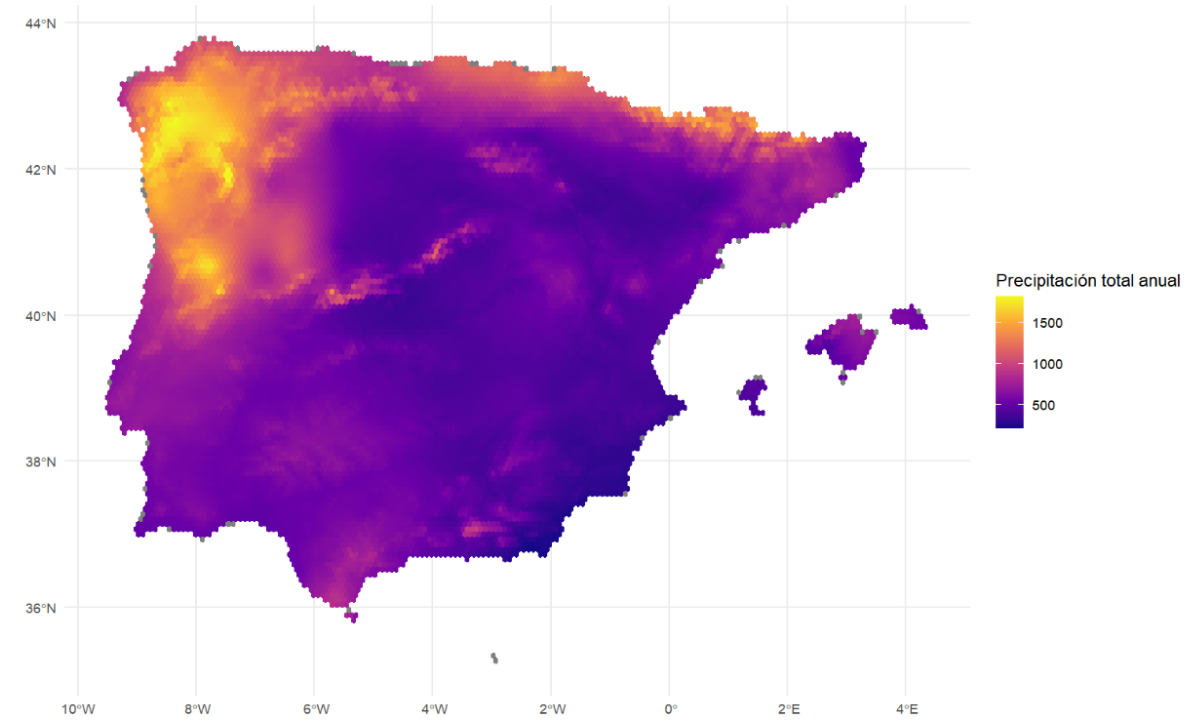

Figura 17: Mapa de la precipitación anual total. Fuente: elaboración propia.

Galicia y la Cordillera Cantábrica destacan por sus valores elevados de precipitación

anual, superando fácilmente los 1500 mm. Regiones como el sudeste peninsular (Murcia, Almería) y el valle del Ebro presentan las precipitaciones más bajas, algunas por debajo de 300–400 mm. El mapa refleja bien el patrón climático peninsular, con un claro descenso de precipitación de oeste a este. Además de la zona en especial de Galicia, también observamos manchas oscuras en las zonas de alta montaña.

4.1.9. Distribución de la variable *viento mean*(Velocidad media del viento)

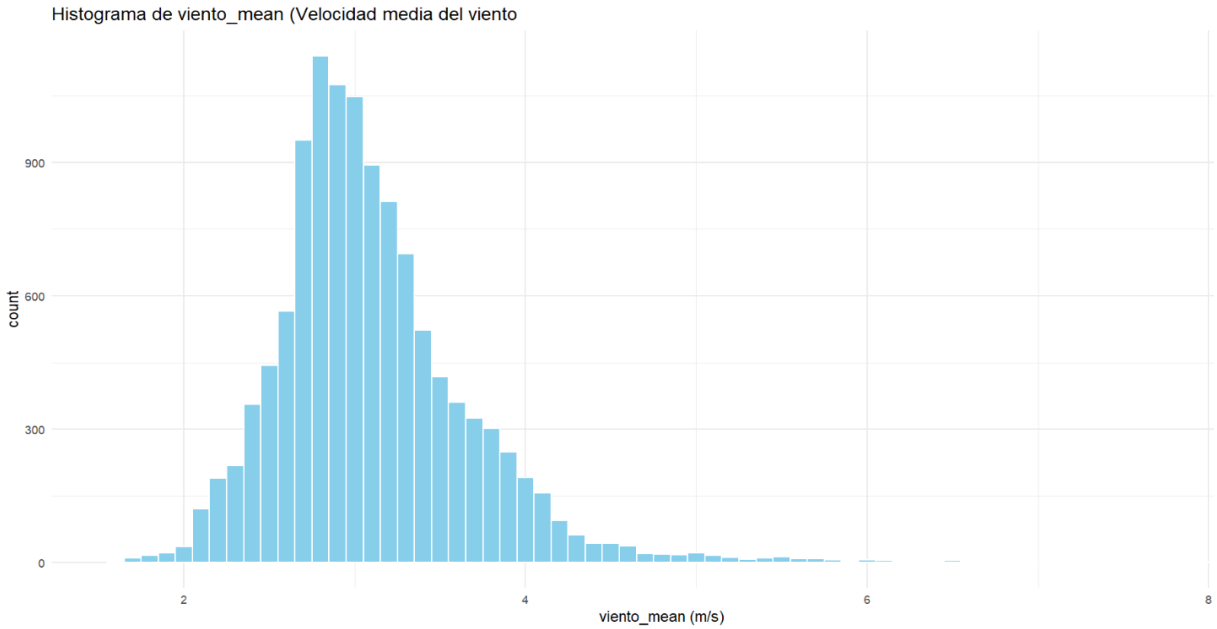

Figura 18: Histrograma de la velocidad media del viento. Fuente: elaboración propia.

En este histograma observamos como la mayoría de los hexágonos tienen velocidades de entre 2 y 4 m/s, con pocos valores por encima de 5 m/s. El valor más frecuente está alrededor de 3 m/s, lo que sugiere que en gran parte de la Península los vientos predominantes no son especialmente intensos. Las velocidades por encima de 5–6 m/s son poco comunes y están restringidas a zonas específicas.

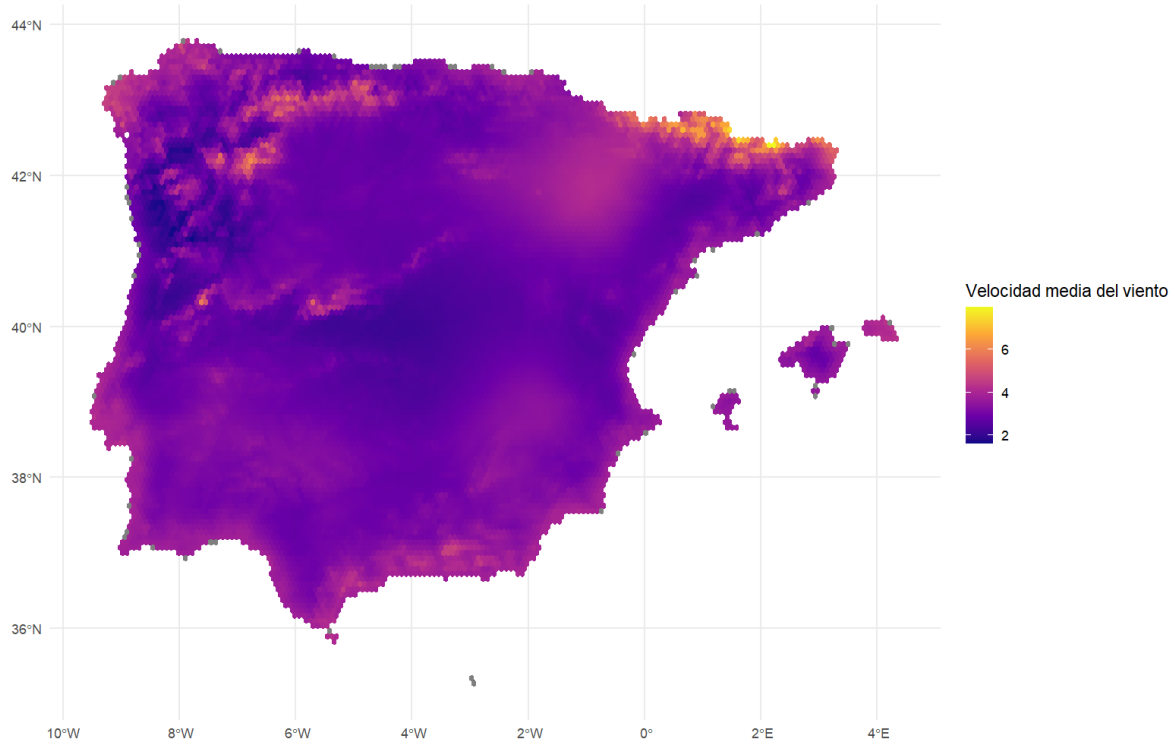

Figura 19: Mapa de la velocidad media del viento. Fuente: elaboración propia.

Se aprecian valores bajos de velocidad media del viento (tonos oscuros) especialmente en el noroeste, sur de Galicia y el norte de Portugal, así como en la zona central de España. Observamos que las zonas con la velocidad del viento más alta son en el valle del Ebro y en los Pirineos, es decir, las montañas actúan como aceleradores del viento en las crestas y barreras en las laderas, lo cual se refleja en los contrastes dentro de las mismas regiones.

4.1.10. Distribución de la variable *viento relativo*(Velocidad relativa del viento)

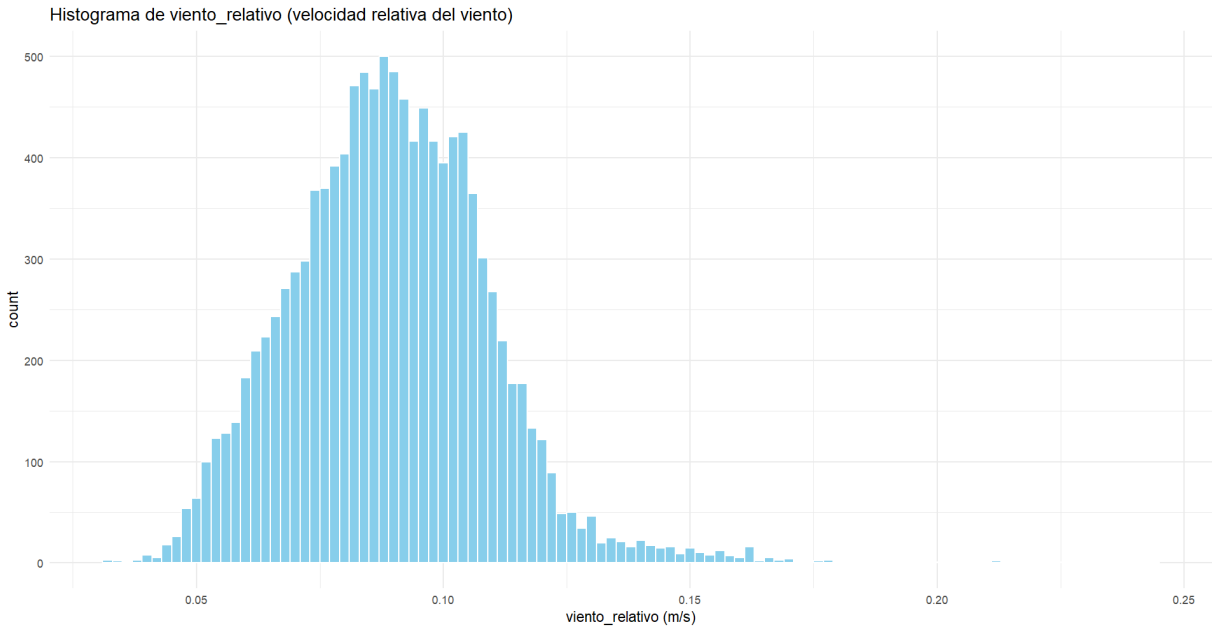

Figura 20: Histograma de la velocidad relativa del viento. Fuente: elaboración propia.

Valores altos indican que, el viento es constante y estable, mientras que valores bajos indican que el viento es variable o inestable. En este caso, hay una clara concentración de valores entre 0.07 y 0.11, la velocidad media del viento es baja en comparación con su variabilidad, lo que nos indica predominio de ráfagas, calmas y cambios irregulares.La mayoría de los valores se agrupan cerca de 0.08. En general vemos valores bastante bajos, un indicador fuerte de inestabilidad del viento. Algunos hexágonos presentan valores altos, indicando zonas con viento más sostenido y menos variable. Esto podría reflejar condiciones más constantes a lo largo del año.

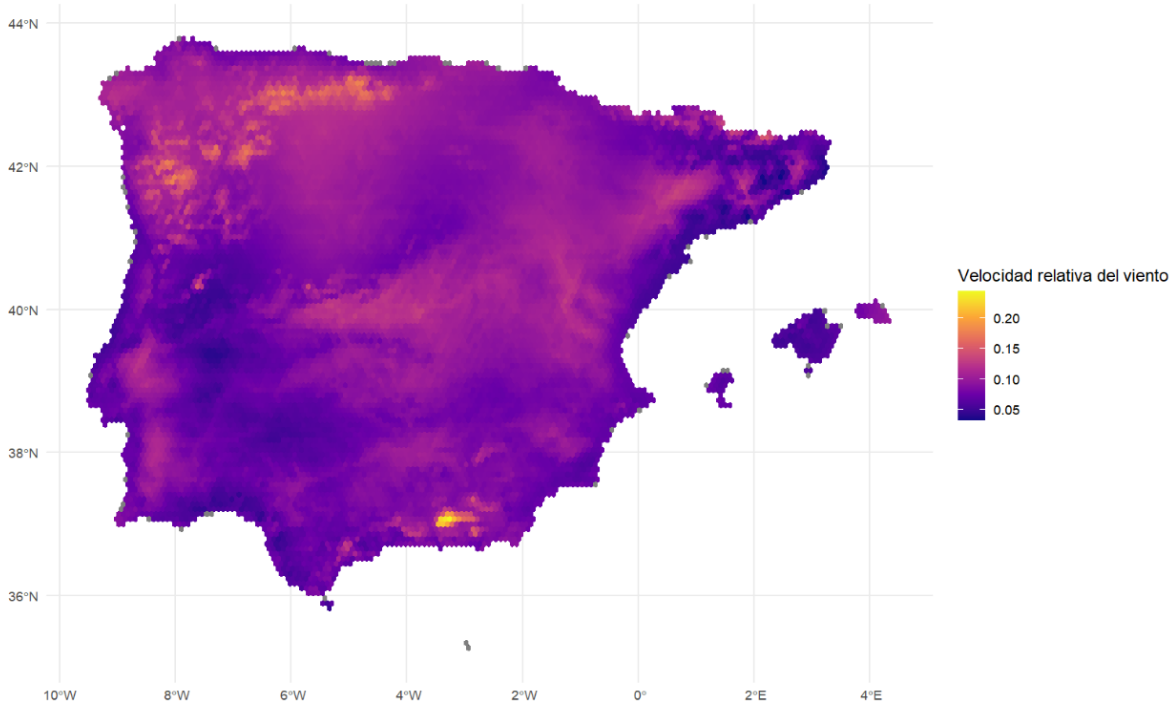

Figura 21: Mapa de la velocidad relativa del viento. Fuente: elaboración propia.

Vemos un mapa muy parecido al de el viento en media, las zonas con viento menos constante o irregular vuleven a ser zonas del noroeste peninsular.

4.1.11. Distribución de la variable *elevacion*

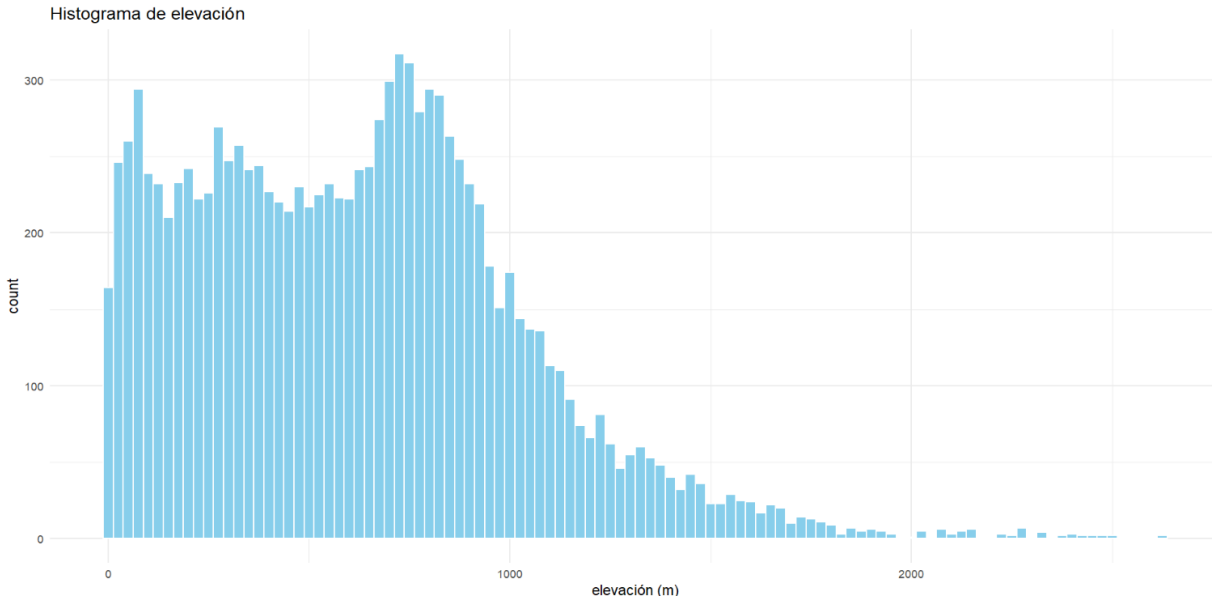

Figura 22: Histograma de la elevación. Fuente: elaboración propia.

La mayor parte del territorio se sitúa por debajo de los 1000 m, lo que indica una predominancia de áreas bajas y medias. Se observa un pico claro entre 600 y 800 m, el cual corresponde a las altitudes características de la Meseta Central, que cubre una gran extensión del interior peninsular.A medida que aumenta la altitud, disminuye rápidamente

el número de hexágonos. Las altitudes superiores a 2000 m son raras. Las zonas con elevación menor a 500 m corresponden a áreas litorales, depresiones fluviales (Guadalquivir, Ebro), y otras llanuras.

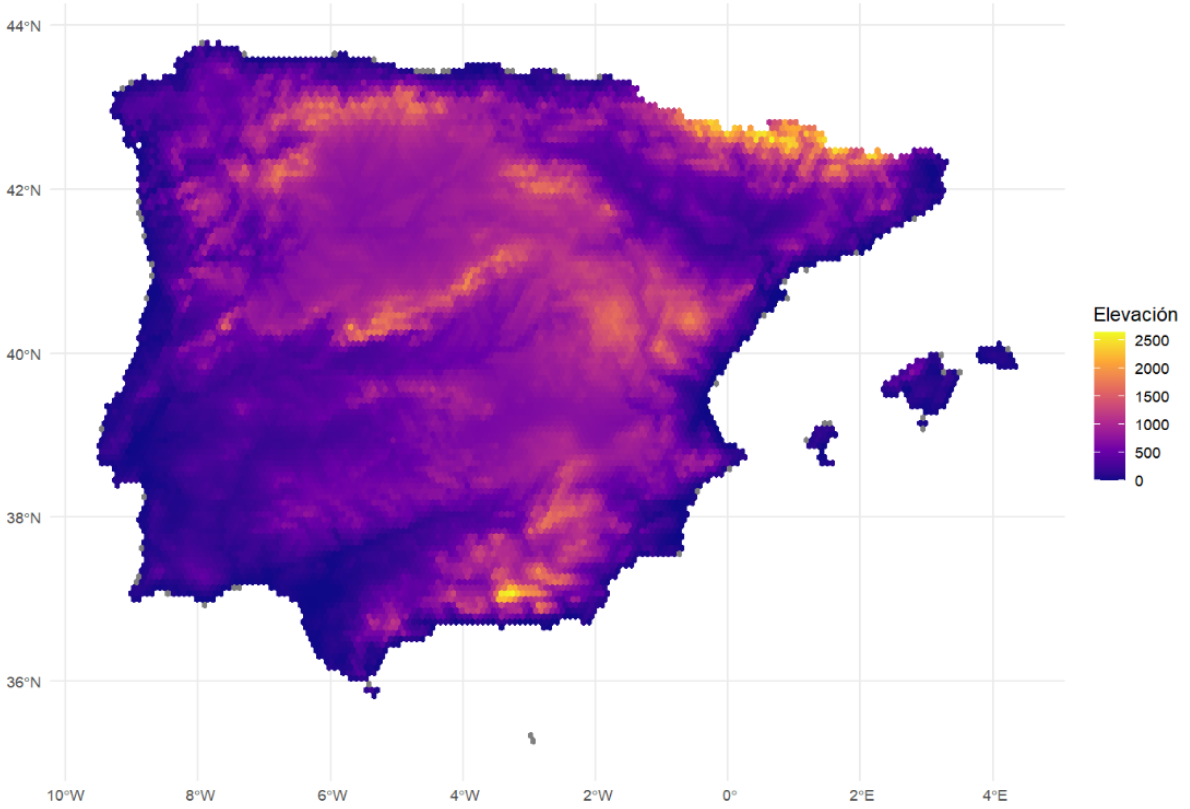

Figura 23: Mapa de la elevación. Fuente: elaboración propia.

Se confirma lo observado en el histograma, elevación alta en zonas de montaña, baja en zonas de costa y mantenida en la meseta. Zonas de mayor altitud tienden a presentar condiciones más frías y específicas, afectando a la distribución de especies tanto vegetales como animales.

4.1.12. Distribución de la variable *porc bosque* (Porcentaje de Bosque)

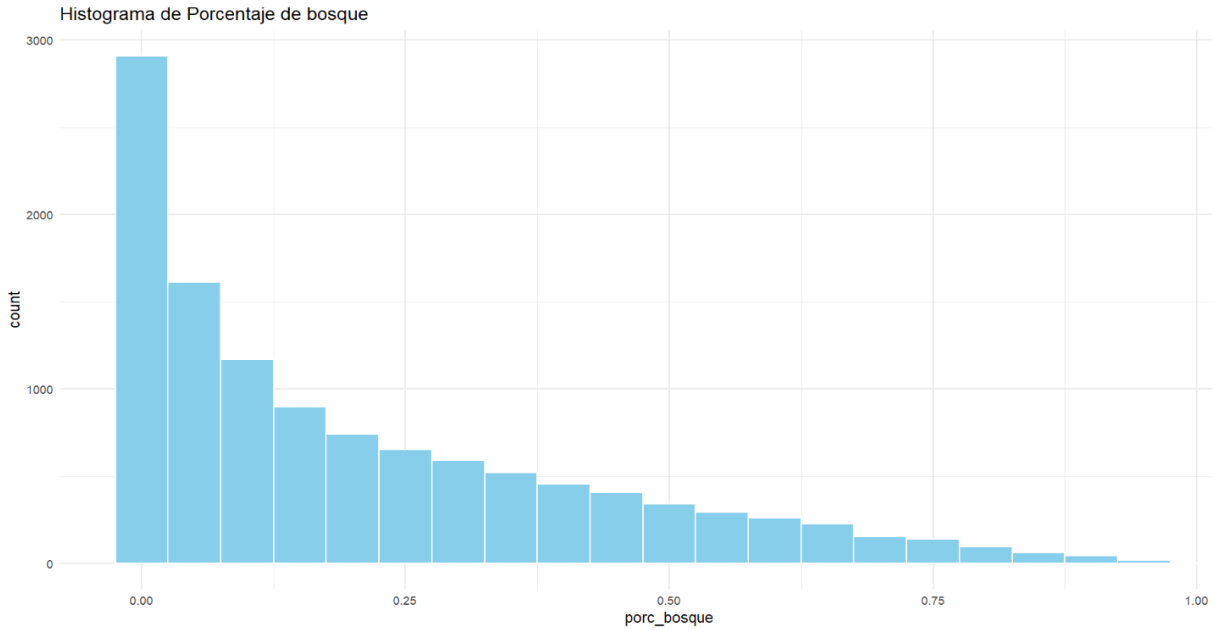

Figura 24: Mapa del porcentaje de bosque. Fuente: elaboración propia.

Distribución altamente asimétrica a la izquierda: con mayoría de hexágonos con porcentaje de bosque bajo, especialmente entre 0 y 0.25, lo que indica una baja cobertura forestal general en gran parte del territorio. Hay una gran acumulación de hexágonos sin bosque o con cobertura mínima (posiblemente áreas urbanas, agrícolas o secas).

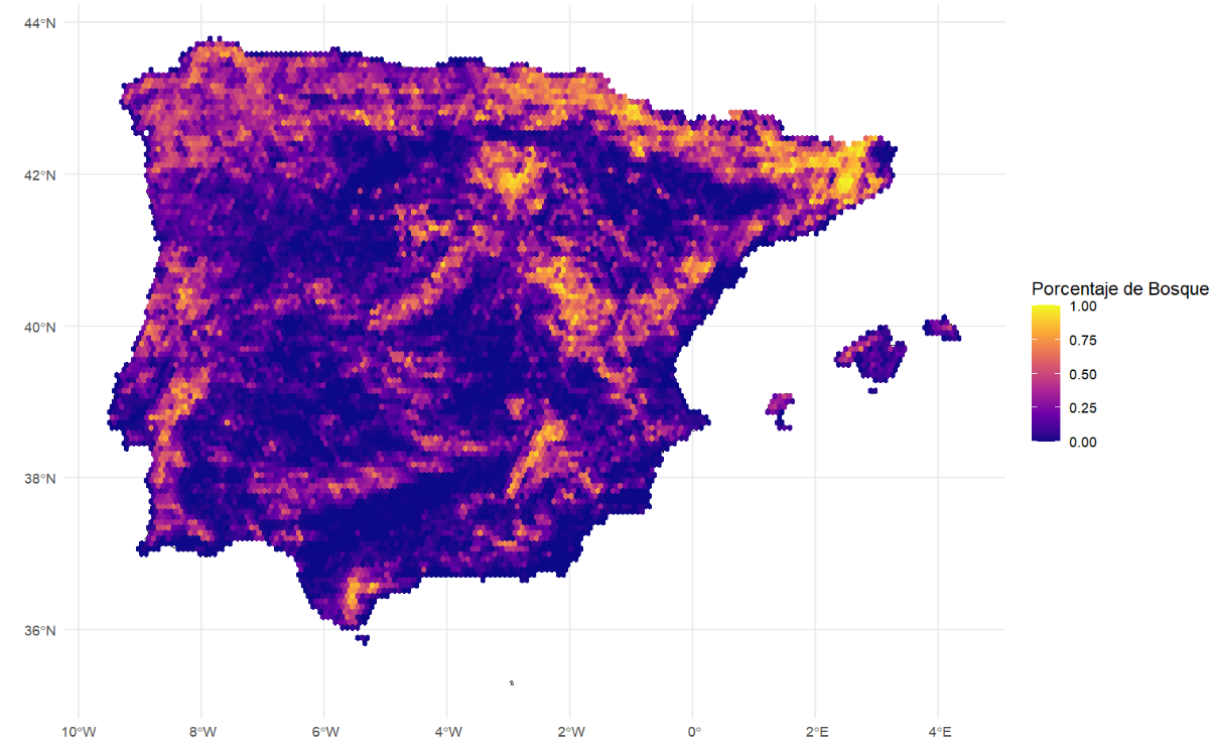

Figura 25: Mapa del porcentaje de bosque. Fuente: elaboración propia.

Observamos alta concentración forestal en zonas montañosas, coincidentes con el sistema central, los montes de León, la Cordillera Cantábrica y Pririneos. Observamos como la zona noroeste de Galicia tiene porcentaje de bosque moderadamente alto por toda la comunidad. Observamos escasa cobertura en las cuencas del Guadalquivir y del Ebro. Las zonas de mayor altitud y humedad coinciden con mayor presencia forestal, lo que refuerza el vínculo entre elevación y cobertura vegetal.

4.1.13. Distribución de la variable *accesibilidad*

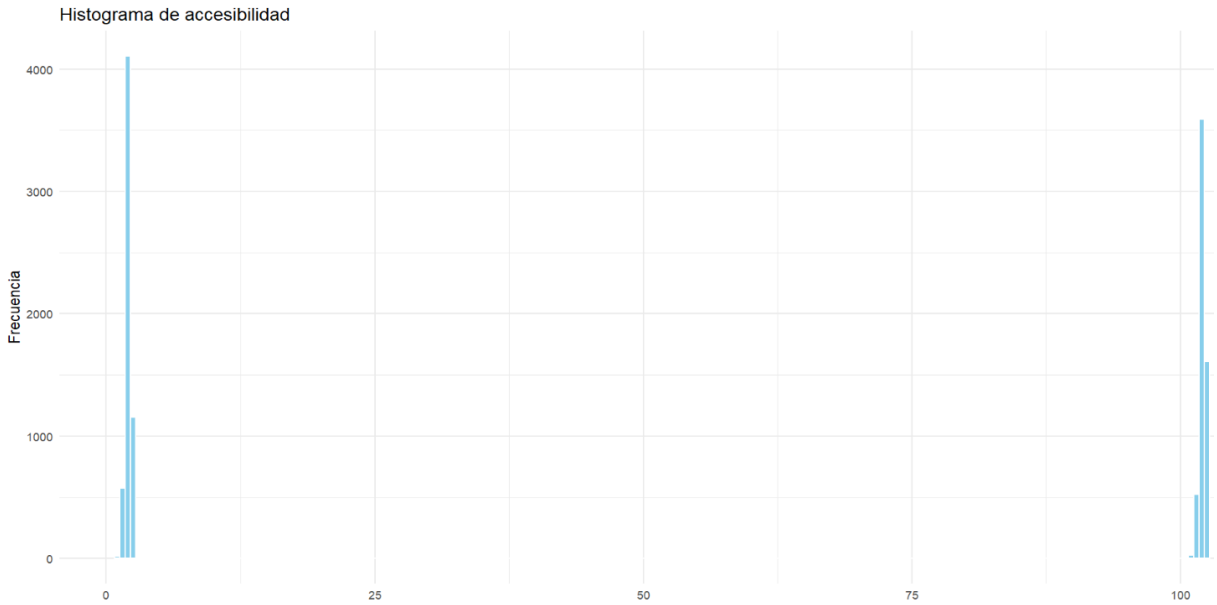

Figura 26: Histograma de Accesibilidad. Fuente: elaboración propia.

La formulación ya explicada anteriormente de accesibilidad genera una distribución altamente bimodal, como se observa en el histograma, donde hay una fuerte concentración de hexágonos con accesibilidad muy alta (zonas urbanas o periurbanas densamente pobladas) y otra con valores muy bajos (zonas montañosas o rurales aisladas). El vacío en el centro de la distribución confirma que hay pocas zonas con accesibilidad "intermedia". Por ello, en este caso el mapa de calor saldrá muy polarizado.

Figura 27: Mapa de accesibilidad. Fuente: elaboración propia

Observamos la clara concentración de gente en las ciudades grandes y el vacío en lugares intermedios o bajos.

4.1.14. Distribución de la variable *Indice Habitat*

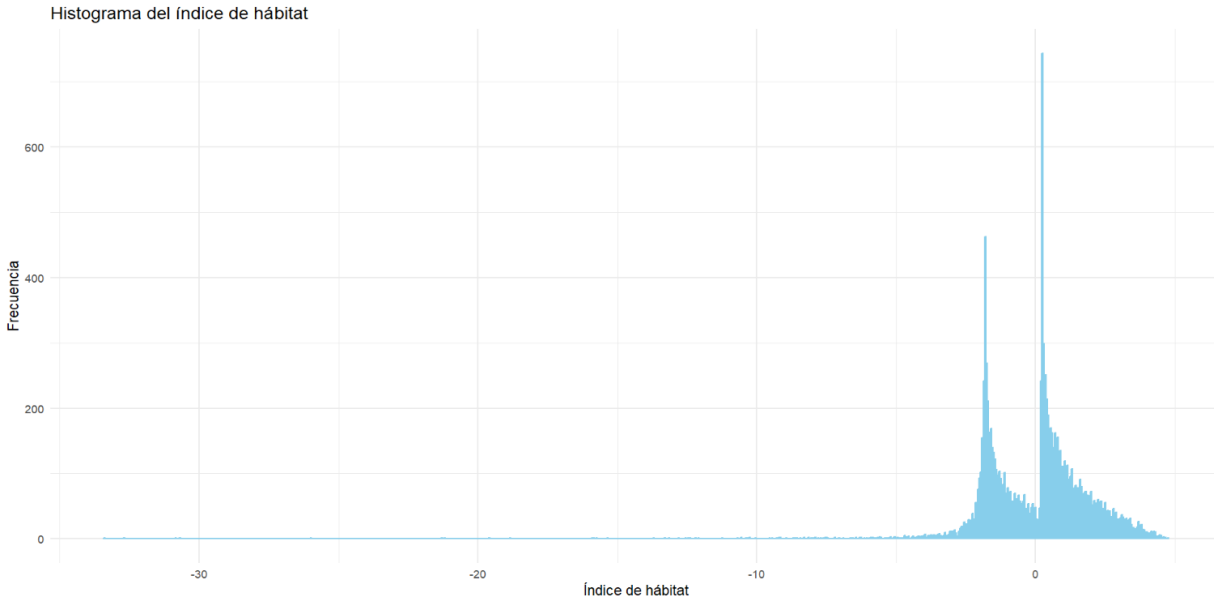

Figura 28: Histograma del Índice de habitat. Fuente: elaboración propia

Recordemos que el índice de habitat es una variable que hemos creado con feature engineering. Es una medida relativa de calidad de hábitat potencial, en la que valores positivos indican hexágonos más forestales, aislados, y con baja densidad local (potencialmente buenos hábitats). Valores cercanos a 0 indican condiciones medias, típicas del paisaje general. Mientras que valores negativos indican zonas más accesibles, densas o deforestadas, es decir, hábitats menos favorable.

El histograma muestra asimetría negativa, con valores acumulados cerca de 0, además de un pico al rededor de 2.5, probablemente relacionado con zonas de bosque o montaña.

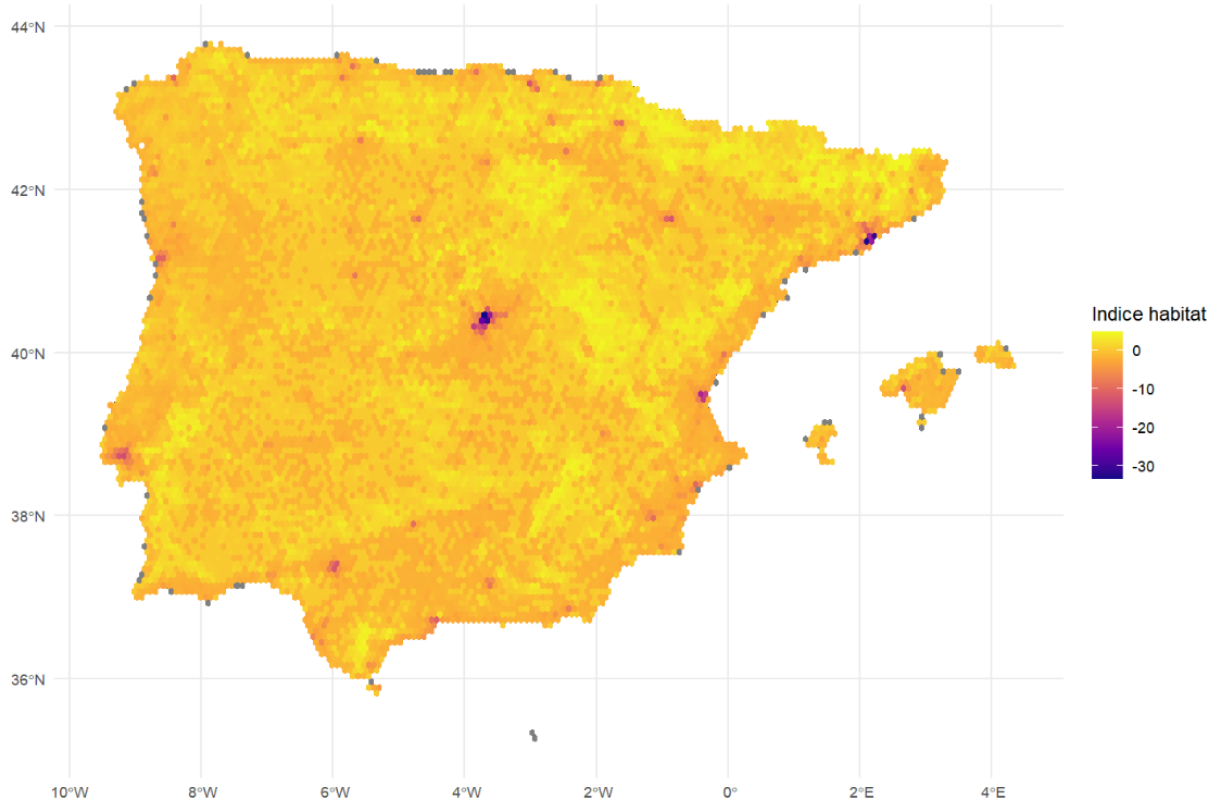

Figura 29: Mapa del Índice de habitat. Fuente: elaboración propia.

Tonos amarillos (cerca de 0 o positivos) cubren buena parte del territorio rural. Manchas moradas o azules (valores negativos extremos) coinciden con ciudades.

4.1.15. Distribución de la variable *Presion humana*

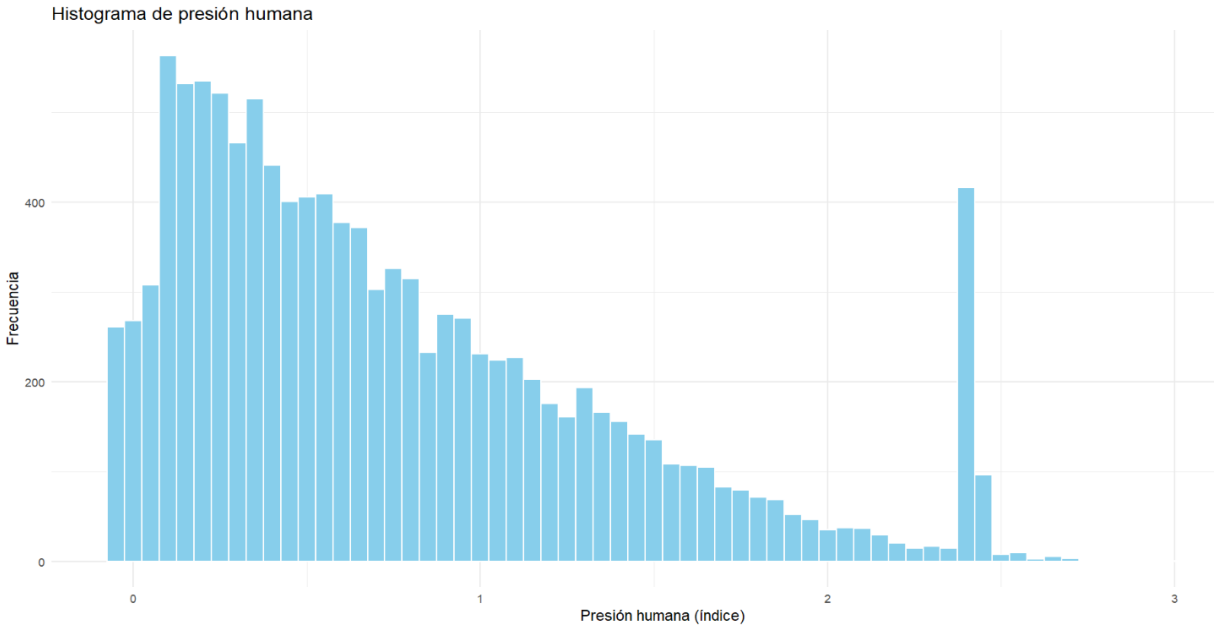

Figura 30: Mapa de presión humana. Fuente: elaboración propia.

Presión humana es una variable que hemos creado combinando presión y accesibilidad, por tanto al contrario que esta última, no es tan brusca. Observamos un índice alto alrededor de 0.2, lo que puede indicarnos muchas zonas con presión humana más bien baja; y otro pico al rededor de 2.5, lo cual podría hacer referencia a las grandes ciudades.

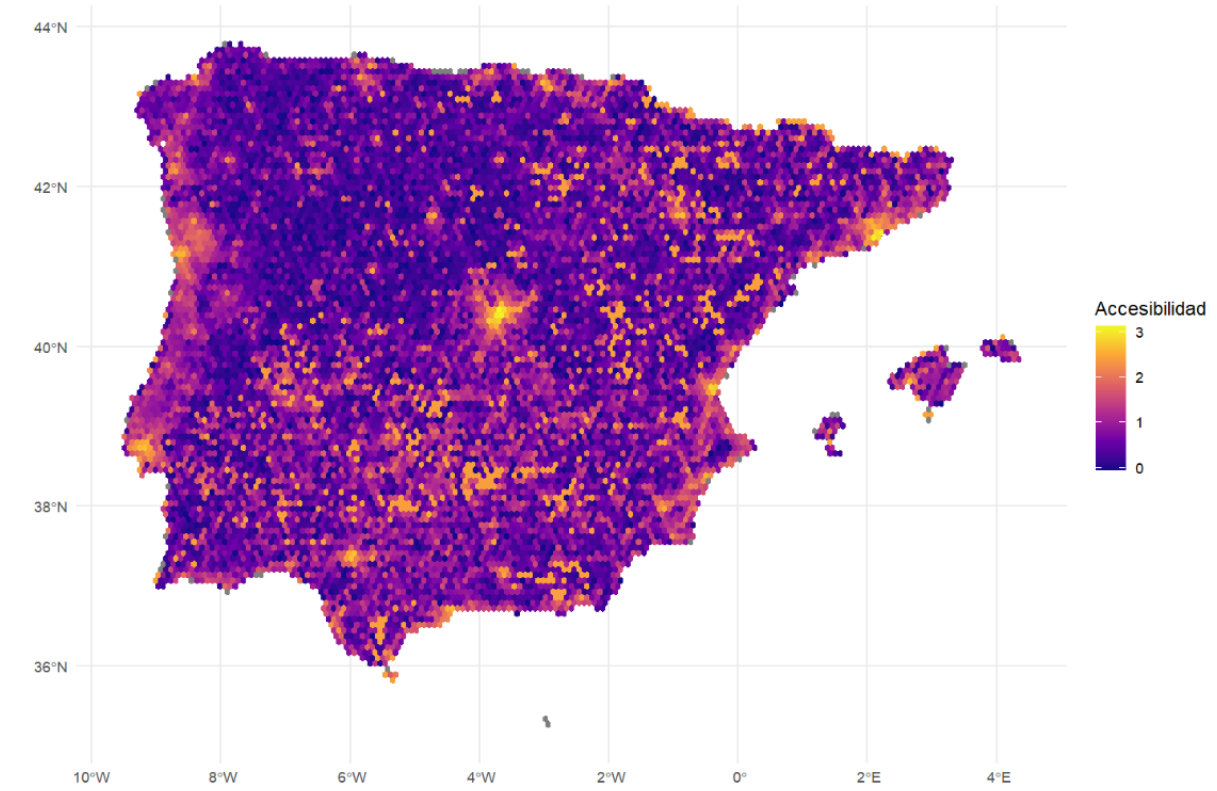

Figura 31: Mapa de presión humana. Fuente: elaboración propia.

Con este mapa se comprueba lo dicho con el histograma. La tónica general es la de una península con presión humana media baja, especialmente en el norte; pero con varias ciudades, tanto en Portugal como en España con presión humana alta (mucha concentración de habitantes y prueblos)

4.1.16. Gráfico de barras de la variable *Porc vecinos con lobo*

Por último, tenemos la variable *porc_vecinos_con_lobo*, una variable que utilizaremos para uno de nuestros dos modelos objetivo, aquel en el que sí conocemos la presencia de lobos en los hexágonos adyacentes. Por tanto sabemos que esta variable será como mínimo 0 (0 vecinos adyacentes) y como máximo 100 (6/6 vecinos adyacentes). Tomará los siguientes valores representados en el gráfico de barras:

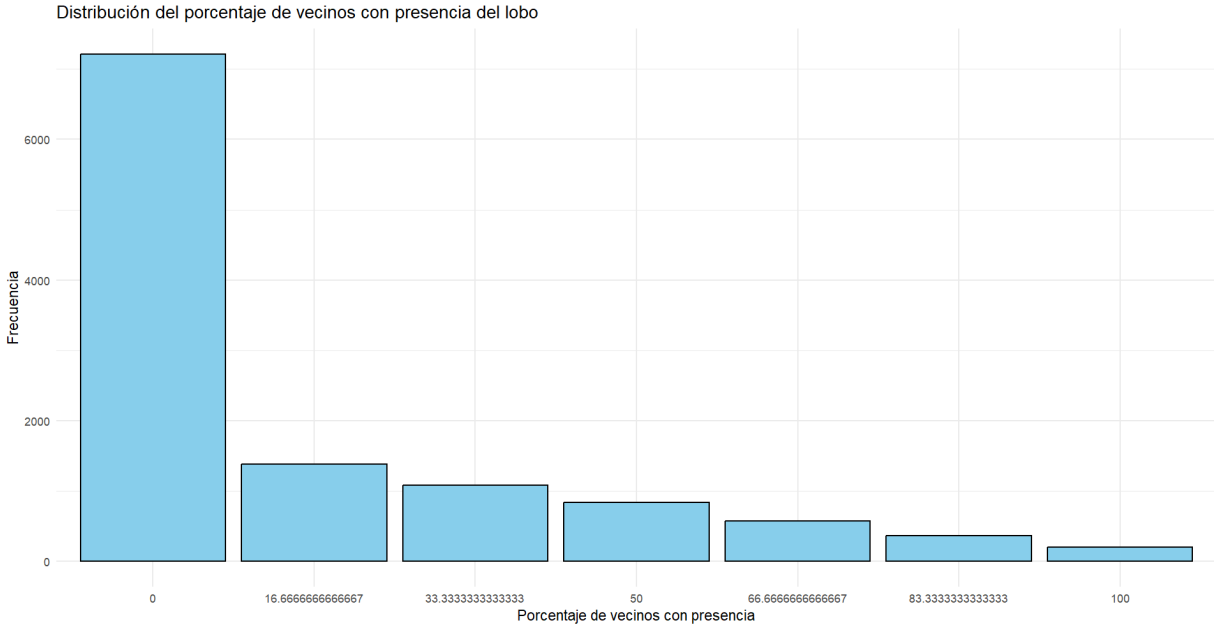

Figura 32: Gráfico de barras del porcentaje de vecinos adyacentes con presencia de lobo. Fuente: elaboración propia.

Observamos como la gran mayoría de valores son 0, esto tiene sentido ya que la mayoría de hexágonos no tienen presencia de lobo. Como es esperado también, la frecuencia baja con el porcentaje de presencia, siendo 100 el porcentaje que menos frecuencia tiene.

4.1.17. Matriz de correlación

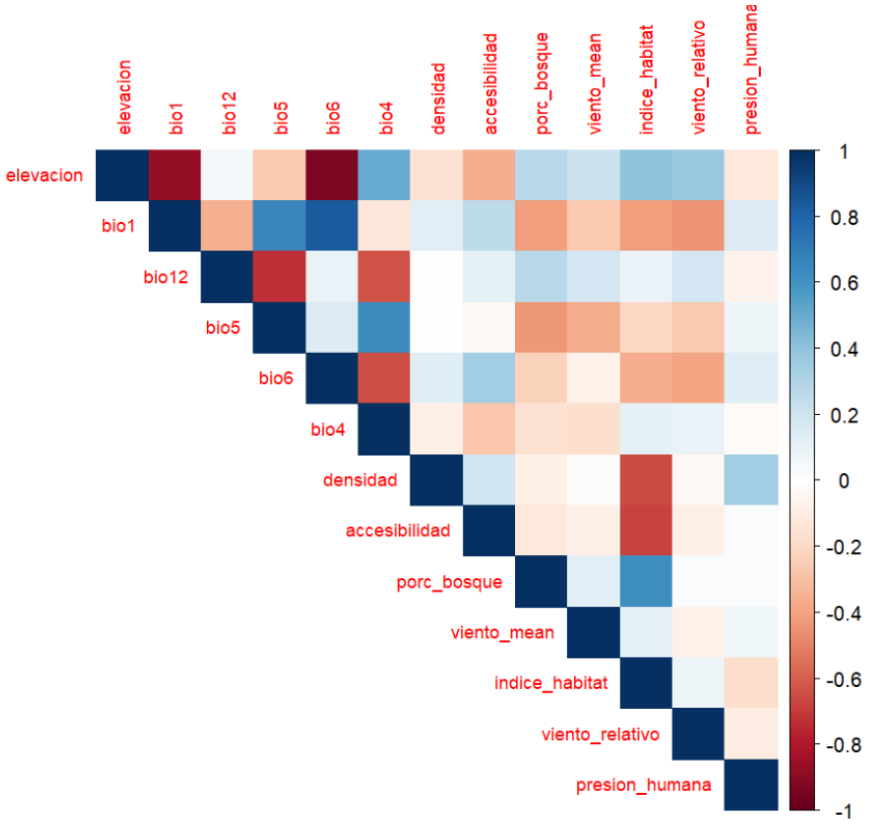

Figura 33: Matriz de correlaciones. Fuente: elaboración propia.

La matriz de correlaciones muestra relaciones coherentes entre las variables consideradas. Las variables relacionadas con el clima, especialmente las térmicas (como `bio1` y `bio6`), tienden a agruparse en torno a patrones asociados a la altitud. Por otro lado, la accesibilidad y la presión humana están claramente relacionadas de forma inversa: las zonas más accesibles suelen tener mayor presión humana. El índice de hábitat, como era esperable por su construcción, se correlaciona negativamente con estas variables de alteración humana, lo que respalda su interpretación como indicador de calidad ecológica. Además, algunas variables como la velocidad del viento, la densidad o la estacionalidad térmica presentan correlaciones bajas con el resto, lo que sugiere que cada predictor aporta información distinta y complementaria, algo positivo de cara a los modelos posteriores.

4.2. Criterios de selección: Stepwise AIC, Stepwise BIC, Stepwise k general, RFE, MMPC y Random Forest

Nuestro objetivo en esta sección es generar modelos predictivos utilizando distintos criterios de selección de variables. Estos métodos permiten reducir el número de variables predictoras y mejorar así la interpretabilidad y el rendimiento de los modelos. Analizaremos enfoques basados en información (como AIC y BIC), métodos de tipo wrapper (RFE), basados en independencia condicional (MMPC) y eligiendo número fijo de variables (k). Finalmente se compararán todos ellos en un Box-Plot para observar cuales dan mejores resultados, o lo que es lo mismo, mayor AUC (area bajo la curva, más tarde se explicará de forma detallada).

A continuación, se describe cada uno de ellos en detalle.

- AIC – Criterio de Información de Akaike

  El criterio de información de Akaike (AIC) fue propuesto en 1974 por Hirotogu Akaike como una estimación de la pérdida de información, la cual implica usar un modelo para representar el proceso que generó los datos. Es una medida ampliamente utilizada para comparar modelos estadísticos desde un enfoque basado en la teoría de la información (la cantidad de incertidumbre o "información" contenida en una fuente de datos).

  AIC tiene como objetivo minimizar la divergencia de Kullback-Leibler, la cual calcula la distancia entre la distribución real de los datos y la distribución aproximada generada por el modelo predictivo. En la práctica, AIC nos permite balancear dos aspectos: primero, conseguir un buen ajuste a los datos (mediante verosimilitud); y segundo, evitar el sobreajuste (penalizando la complejidad del modelo, cuantas más variables peor). Tiene la siguiente formula:

  $$\mathrm{AIC} = 2k - 2\ln(\hat{L}) \tag{1}$$

  donde:

  - k es el número de parámetros estimados en el modelo (incluyendo el intercepto),
  - $\hat{L}$ es el valor máximo de la función de verosimilitud del modelo ajustado.

  Guiada por el criterio AIC como medida de evaluación del rendimiento del modelo, el Stepwise AIC es una técnica se realizará una técnica automatizada que tiene por objetivo la selección de un subconjunto óptimo de variables predictoras.

  A diferencia del AIC normal, que compara modelos fijos, el Stepwise es una medida dinámica que utiliza el AIC en cada paso del proceso de selección. Esto permite crear un modelo de manera progresiva, buscando aquel que minimice el error.

  Hay tres formas de llevarlo a cabo, uno partiendo de un modelo completo y eliminando una a una, otro partiendo d eun modelo vacío y de por último de forma bidireccional. El que se usará en este trabajo es el bidireccional. El primer paso es considerar un modelo vacío, Se consideran todos los modelos adyacentes a este, añadiendo una nueva variable y eliminando una variable existente. Para cada modelo candidato se calcula el AIC, se elige el que mejor tenga, si el nuevo modelo mejora respecto al anterior, se actualiza y el proceso continúa, si no, nos se queda con el

anterior y el algoritmo termina (cuando ninguna adición ni eliminación mejora el AIC).

Para la selección automática de variables se utilizó la función `stepAIC()` del paquete `MASS` en R. En este caso, se parte de un modelo vacío y se define como `scope` el conjunto completo de variables disponibles. Se especifica la opción `direction = "both"` para permitir tanto la inclusión como la eliminación de variables en cada iteración del algoritmo, es decir, indicar que es bidireccional. El argumento `trace = FALSE` anula que se enseñe todo el proceso por pantalla.

Una vez que el proceso eliga el modelo óptimo, se extraen los nombres de las variables incluidas en este modelo mediante `names(coef(modelo_aic))[-1]`, eliminando el intercepto, que no nos hace falta. Finalmente, se consulta el número de variables seleccionadas (`length(vec)`) y su contenido (`dput(vec)`).

- BIC

Unos años más tarde, en 1978, Gideon Schwarz propone el criterio de información bayesiano (BIC), también conocido como criterio de Schwarz, como una alternativa con enfoque bayesiano del criterio AIC. Al igual que este último, tiene como objetivo equilibrar ajuste del modelo y complejidad, pero en este caso el BIC penaliza más fuertemente por número de parámetros, especialmente en muestras de mayor tamaño.

Este criterio evita el calculo directo de probabilidad de cada modelo, lo cual implica realizar operaciones complicadas y poco practicas, y lo hace aproximando la probabilidad con una formula simplificada. Esta aproximación asume una distribución a priori uniforme sobre los parámetros del modelo, para luego aplicar una transformación logarítmica para facilitar el cálculo. Gracias a esto, el criterio BIC se convierte en una herramienta práctica y eficaz, sobre todo cuando se trabaja con muestras grandes o con muchas variables predictoras.

El BIC se calcula con la siguiente fórmula:

$$\text{BIC} = k \ln(n) - 2 \ln(\hat{L}) \tag{2}$$

donde:

- k es el número de parámetros estimados en el modelo (incluyendo el intercepto),
- n es el número de observaciones (tamaño muestral),
- $\hat{L}$ es el valor máximo de la función de verosimilitud del modelo ajustado.

Buscamos que este valor sea el menor posible (cuanto más pequeño sea, mejor modelo se considera en comparación con otros). Podemos observar que a diferencia del AIC, el cual penaliza el número de variables incluidas con un valor constante ($2k$), el BIC penaliza en función del tamaño muestral ($k \ln(n)$), por lo que tiende a seleccionar modelos más simples a medida que n se hace más grande. Esta propiedad hace que BIC sea consistente.

Al igual que para el AIC, aplicamos el BIC en un Stepwise que comparará varios modelos y eligirá el óptimo según este criterio. El procedimiento `stepwise BIC` sigue el mismo esquema que el algoritmo `stepwise AIC`, pero utiliza el BIC como

métrica de selección. Esto implica que penaliza más por número de parámetros, lo que tiende a generar modelos más simples.

En cada paso, se evalúan posibles adiciones o eliminaciones de variables, y se conserva aquella combinación que minimice el BIC. Para su implementación en R, se adaptó la la misma función `stepAIC()` del paquete `MASS`, estableciendo el parámetro `k = log(n)` para reflejar la fórmula del BIC. En este caso nuestro $n = 11605$, por tanto ($ln(11605) = 9{,}3591$), redondeando, 9.

- K general

  En este trabajo se utilizó también un procedimiento stepwise guiado por un criterio de penalización general $k = 15$. Este valor no representa el número de variables seleccionadas, sino el peso asignado a cada parámetro del modelo en la penalización. La fórmula utilizada para evaluar cada modelo fue:

$$\text{Criterio}_k = k \cdot (\text{número de parámetros}) - 2\ln(\hat{L}) \tag{3}$$

  donde:

  - k es el valor de penalización establecido por el usuario ($k = 15$ en este caso),
  - $\hat{L}$ es el valor máximo de la función de verosimilitud del modelo ajustado.

  En teoría, no hay una recomendación exacta para usar en los K además de los valores que ya definen los dos criterior anteriores (AIC y BIC). Si se usa un valor distinto, se considera una decisión analítica basada en objetivos propios del estudio. Este enfoque permite un mayor control sobre la complejidad del modelo final, escogiendo un valor de k que se situe entre el $k = 2$ de AIC y el $ln(k)$ del BIC. Probar valores intermedios como 5, 7 o 10 permite ajustar la penalización según el objetivo del estudio, encontrando un equilibrio entre ajuste y simplicidad.

  En este caso se utilizó una variante del algoritmo `stepAIC()`, en la que se definió un valor personalizado de penalización (`k = 15`).

- RFE - Recursive Feature Elimination

  El método RFE (Recursive Feature Elimination) es una técnica de selección de variables de tipo wrapper, que se basa en entrenar un modelo repetidamente, eliminando en cada iteración las que menos aporten al rendimeinto final del modelo. Al contrario que con los modelos anteriores, los cuales penalizaban la complejidad de forma teorica, el RFE decide en función del comportamiento real del modelo al ajustar los datos. [25]

  Primero se ajusta un modelo predictivo (por ejemplo, regresión lineal, SVM o árbol). y se mide la importancia de cada variable según el modelo. La variable que salga menos importante se elimina y se repite el proceso en el subconjunto restante. El proceso se detiene cuando se alcanza el número predeterminado de variables (o el número máximo de variables disponibles) o si se alcanza una métrica óptima fijada también de antemano. [30]

  Si $\mathcal{V} = \{v_1, v_2, \ldots, v_p\}$ es el conjunto de predictores y $M(\mathcal{V})$ representa el modelo entrenado con ellos, entonces el algoritmo RFE produce una secuencia de subconjuntos de la forma:

$$\mathcal{V} \to \mathcal{V}_{-v_j} \to \mathcal{V}_{-v_j-v_k} \to \cdots \to \mathcal{V}_r$$

donde $\mathcal{V}_r$ es el conjunto final de r variables seleccionadas tras sucesivas eliminaciones basadas en su menor relevancia para el modelo.

Ventajas de RFE	**Limitaciones de RFE**
Permite usar modelos complejos como random forest o regresión logística.	Es más costoso computacionalmente que métodos como AIC o BIC.
Captura interacciones y no linealidades si el modelo base lo permite.	El resultado depende del modelo utilizado; no es universal.
Muy útil en problemas con muchas variables y relaciones complejas.	Puede seleccionar variables inestables si el modelo es sensible a pequeñas variaciones en los datos.

Cuadro 4: Ventajas y limitaciones del método RFE.

En este trabajo se aplicó RFE utilizando el paquete `caret` de R, con `random forest` como modelo base (`rfFuncs`) y validación cruzada de 4 particiones (`cv`). Se definieron diferentes tamaños posibles del subconjunto de variables, para evaluar qué combinación ofrecía el mejor rendimiento predictivo. Más adelante mostraremos los resultados. [16]

- MMPC - Max-Min Parents and Children

  MMPC (Max-Min Parents and Children) es un algoritmo de selección de variables que se basa en evaluar relaciones de independencia condicional entre las variables independientes y la variable dependiente. Tiene por objetivo hayar el conjunto mínimo de predictores que están directamente relacionados con la variable respuesta, sin que sea necesario construir modelos completos. Se aplican una serie de tests estadísticos (como Chi-cuadrado, G^2 o mutual information) que sirven para comprobar si una variable aporta información relevante sobre la dependiente, teniendo en cuenta las anteriormente seleccionadas. [47]

  MMPC se basa en la construcción de un grafo de dependencia estadística, en el cual cada nodo representa una variable, y las conexiones se determinan a través de los tests de independencia condicional anteriormente mencionados. El objetivo principal del algoritmo es identificar el conjunto mínimo de variables S tal que ninguna variable fuera de S proporciona información adicional sobre la variable objetivo una vez condicionada a S.

  Si T es la variable objetivo y X_i una variable candidata, MMPC busca el conjunto S tal que:

  $$X_i \not\perp\!\!\!\perp T \mid S \quad \text{(dependencia condicional)}$$

Esto se interpreta como: "X_i no es independiente de T dado S".

En primer lugar se inicializa un conjunto vacío, y en cada paso, para cada una de las variables candidatas se calcula el mínimo valor de dependencia condicional entre la variable predictora y la objetivo, condicionando sobre subconjuntos ya seleccionados. Se selecciona aquella cuyo valor de dependencia mínimo sea máximo y se repite el proceso hasta que ninguna de las variables traspase el p-valor.

Este enfoque resulta muy útil cuando lo que se quiere es reducir el número de variables quedándonos solo las que realmente aportan información nueva y no redundante. En este trabajo, el algoritmo se aplicó mediante el paquete `MXM` en R, utilizando el test G^2 como medida de dependencia, adecuado para variables discretas. [**?**]

Ventajas de MMPC	**Limitaciones de MMPC**
Selecciona variables realmente relevantes para la predicción.	Requiere elegir un test estadístico apropiado.
Basado en fundamentos estadísticos sólidos (independencia condicional).	Puede ser sensible al tamaño de muestra y al umbral de significación.
Apto para datos de alta dimensión.	No se basa en el rendimiento de un modelo, sino en relaciones estadísticas.

Cuadro 5: Ventajas y limitaciones del método MMPC.

La función utilizada para los mmpc en R tiene varios parámetros de entrada, pero hay uno que podemos tunear o probar varios valores, ya que según el que introduzcamos podrán salir resultados diferentes. Se trata del theshold:

- `threshold`: valor umbral de significación (p-valor) utilizado para decidir si una variable es considerada dependiente de la respuesta. Por ejemplo, si se establece en 0,05, implica que solo se incluyen variables cuya relación con la variable objetivo es estadísticamente significativa al 95 % de confianza.

- Random Forest Aunque el método Random Forest será tratado con mayor detalle más adelante, junto con otros algoritmos de aprendizaje automático, en este apartado también se empleará a partir de un conjunto con todas las variables. El objetivo es observar qué variables selecciona a partir de un modelo completo y compararlo con los resultados obtenidos por los otros métodos presentados. Por ello, explicaremos vagamente en este apartado de qué trata, y en profundidad en apartados posteriores.

  Random Forest es un algoritmo de aprendizaje automático del tipo ensamble, que combina múltiples árboles de decisión para mejorar la precisión y reducir el sobreajuste. Fue propuesto por Leo Breiman (2001). Es útil tanto para problemas de clasificación como de regresión

4.2.1. Sin información espacial

A continuación, en este apartado mostraremos todos los resultados obtenidos en R para la ejecución de todos los procedimientos anteriormente mencionados. En este caso se realizará sin la información de la variable que recoge información de correlación espacial.

- AIC

  El número óptimo de variables seleccionadas de nuestro modelo mediente el proceso AIC ha sido 10:

  ```
  c("bio1", "viento_mean", "bio12", "presion_humana", "porc_bosque",
  "bio6", "bio4", "viento_relativo", .elevacion", "densidad")
  ```

- BIC

  El número óptimo de variables seleccionadas de nuestro modelo mediente el proceso BIC ha sido 9:

  ```
  c("bio1", "viento_mean", "bio12", "presion_humana", "porc_bosque",
  "bio6", "bio4", "viento_relativo", .elevacion")
  ```

- K general

  El número óptimo de variables seleccionadas de nuestro modelo mediente el proceso K general ha sido 9 (las mismas que para BIC):

  ```
  c("bio1", "viento_mean", "bio12", "presion_humana", "porc_bosque",
  "bio6", "bio4", "viento_relativo", .elevacion")
  ```

- RFE

  En el caso de RFE la primera salida que tenemos es la de un gráfico, el cual nos muestra en el eje X el número de variables seleccionadas y en el eje Y el error de predicción que se genera con cada variable añadida (en nuestro caso se medirá con el MAE: error absoluto medio). Este gráfico nos permite visualizar como varía el rendimiento del modelo a medida que se incrementa el número de variables predictoras, ayudandonos a saber cuántas variables nos conviene conservar para lograr un equilibrio entre simplicidad y precision. Según la literatura especializada (Guyon et al., 2002; Kuhn & Johnson, 2013), se suele elegir el número óptimo de variables en RFE en el punto donde el error alcanza su mínimo. No obstante, también es aceptable considerar el punto donde el error se estabiliza como criterio de selección, tal como proponen autores como Elith et al. (2010), con el fin de evitar sobreajuste (a veces un ínfimo descenso en error no justifica la adición de variables). En nuestro caso tenemos el siguiente gráfico:

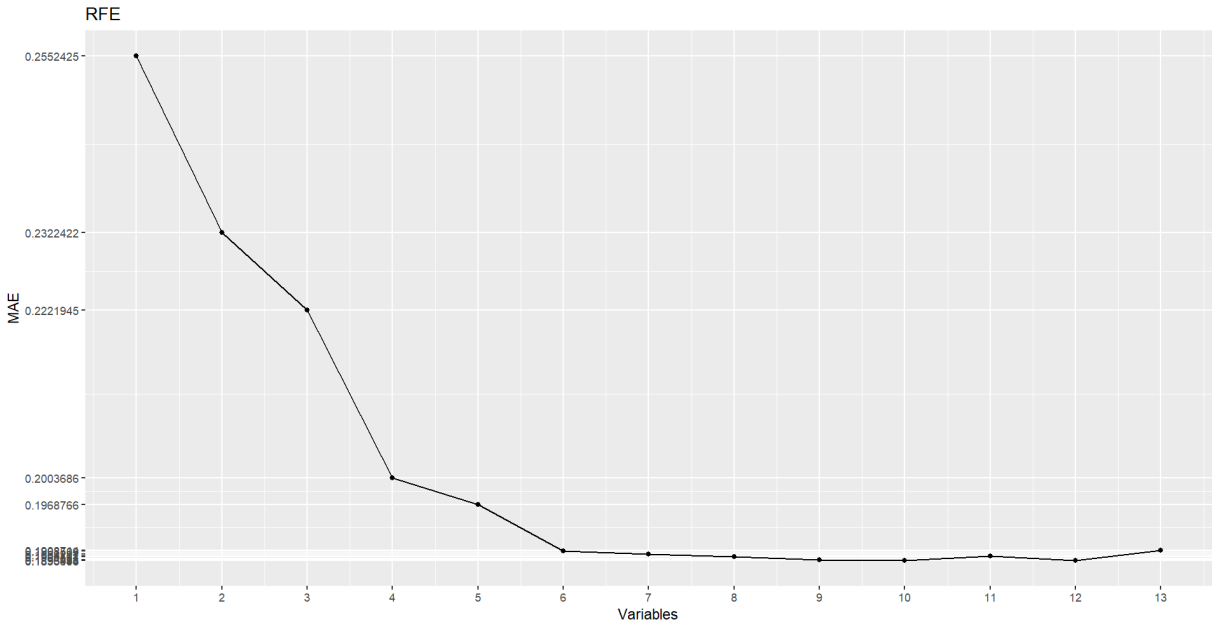

Figura 34: Gráfico RFE. Fuente: elaboración propia.

Observamos en el gráfico como el mínimo se alcanza en 9 variables, y como a partir de ahí el error no baja más. Sin embargo, a partir de 6 variables el error se estabiliza. Según la literatura explicada anteriormente ambas serían opciones de eleción, en este caso se ha decidido escoger ambas para su porterior comparación.

- Modelo con 6 variables

  `c("viento_relativo", .ªccesibilidad", "viento_mean", "bio5", "porc_bosque", "bio12")`

- Modelo con 9 variables

  `c("viento_relativo", .ªccesibilidad", "viento_mean", "bio5", "porc_bosque", "bio12", "bio4", .ªlevacion", "bio1")`

- MMPC

  Para MMPC se han probado varios thresholds entre 0.01 y 0.1 y todos han dado la misma combinación de 6 variables:

  `c(.ªlevacion", "bio1", "bio12", "viento_mean", "viento_relativo", "presion_humana")`

4.2.2. Con información espacial

En el siguiente apartado mostraremos los resultados para el modelo con información espacial, es decir, añadimos la variable *porc_vecinos_con_lobo* al conjunto de preselección de variables, conocemos el porcentaje de hexágonos adyacentes que tienen presente el lobo.

- AIC

El número óptimo de variables seleccionadas de nuestro modelo mediente el proceso AIC ha sido 11:

```
c("porc_vecinos_con_lobo", "bio1", "viento_mean", "bio12",
"presion_humana", indice_habitat", "bio6", "bio4", "viento_relativo",
.elevacion", "porc_bosque")
```

- BIC

  El número óptimo de variables seleccionadas de nuestro modelo mediente el proceso BIC ha sido 10:

  ```
  c("porc_vecinos_con_lobo", "bio1", "viento_mean", "bio12",
  "presion_humana", ïndice_habitat", "bio6", "bio4", "viento_relativo",
  .elevacion")
  ```

- K general

  El número óptimo de variables seleccionadas de nuestro modelo mediente el proceso K general ha sido 9 :

  ```
  c("porc_vecinos_con_lobo", "bio1", "viento_mean", "bio12",
  "presion_humana", "bio6", "bio4", "viento_relativo", .elevacion")
  ```

- RFE

  Para el RFE observamos el siguiente gráfico de importancia de variables:

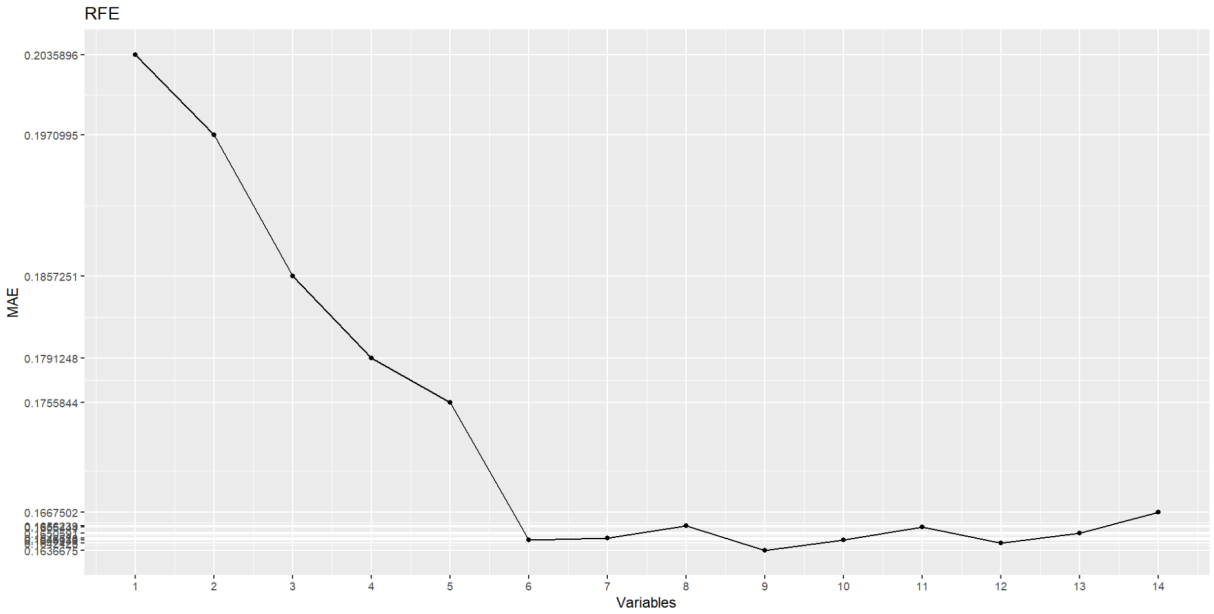

Figura 35: Gráfico RFE. Fuente: elaboración propia.

Observamos en el gráfico como el mínimo se alcanza en 9 variables, y cómo a partir de ahí el error no baja más. Sin embargo, a partir de 6 variables el error se estabiliza. De nuevo, ocurre lo mismo que para el modelo sin la información adyacente.

- Modelo con 6 variables:

  `c("porc_vecinos_con_lobo", "viento_relativo", .ªccesibilidad", "bio4", "viento_mean", "bio5")`

- Modelo con 9 variables:

  `c("porc_vecinos_con_lobo", "viento_relativo", .ªccesibilidad", "bio4", "viento_mean", "bio5", "bio12", .ªlevacion", "porc_bosque")`

- MMPC

  Para MMPC se han probado varios thresholds entre 0.01 y 0.1, en este caso para 0.1 y para 0.01 han salido dos modelos diferentes:

  - Modelo al 0.01 con 6 variables

    `c(.ªlevacion", "bio1", "bio12", "viento_relativo", "presion_humana", "porc_vecinos_con_lobo")`

  - Modelo al 0.1 con 7 variables

    `c(.ªlevacion", "bio1", "bio12", "viento_mean", "viento_relativo", "presion_humana", "porc_vecinos_con_lobo")`

4.3. Comparación de métodos y resultados

Una vez seleccionados distintos subconjuntos de variables mediante métodos como AIC, BIC, RFE, MMPC o Random Forest, el siguiente paso consiste en comparar el rendimiento de los diferentes modelos que han generado cada uno de los subconjuntos. Para ello, se emplea una métrica llamada AUC (Área Bajo la Curva ROC), para entenderla explicaremos primero qué es la curva ROC.

La curva ROC (Receiver Operating Characteristic) es una representación gráfica de la relación entre (la tasa de verdaderos positivos, en este caso sería tasa de presencias clasificadas correctamente) y (la tasa de falsos positivos, en este caso sería la tasa de ausencias clasificadas como presencia) para todos los posibles umbrales de clasificación, siendo el eje de las X $1 - especificidad$ y el eje de las Y $sensibilidad$ (Bradley, 1997). El umbral de clasificación es el punto de corte que indica a partir de qué probabilidad se clasifica como presencia o ausencia una predicción probabilística; por ejemplo, si el umbral es 0.5, eso quiere decir que todas las observaciones con una probabilidad estimada igual o superior al 50 % se clasificarán como presencia.

El objetivo del AUC es cuantificar el área bajo esta curva ROC, representando una medida del poder discriminativo del modelo, es decir, su capacidad para distinguir correctamente entre las dos clases (Fawcett, 2006). A diferencia del MSE, que se aplica a variables continuas, el AUC es más apropiado en este tipo de problemas donde la variable respuesta es categórica.

Un modelo con clasificación perfecta tendrá una curva ROC que pasa por el punto (0,1) y tendrá un AUC = 1, lo cual indica lo cual indica una tasa de verdaderos positivos del 100 % y una tasa de falsos positivos del 0 %. Por el contrario, un modelo de predicción aleatoria, uno que no discrimina en absoluto entre presencias y ausencias, generará una diagonal que pasa por los puntos (0,0) y (1,1); con un AUC = 0.5, sería igual de útil tirar una moneda al aire. Valores intermedios entre 0.5 y 1 reflejan diferentes niveles de calidad del modelo de clsificación, buscando que sea lo más alto posible.En términos prácticos, el AUC es la probabilidad de que, elegidas una presencia y una ausencia al azar, el modelo asigne una mayor probabilidad de presencia a la verdadera que a la ausencia. Por ejemplo, un AUC de 0.85 indica que existe un 85 % de probabilidad de que el modelo clasifique correctamente una presencia frente a una ausencia aleatoria (Hanley & McNeil, 1982).

Esta métrica tiene la ventaja de no depender de un umbral de corte fijo, lo cual es especialmente útil cuando no se desea tomar decisiones binarias inmediatas, sino comparar modelos en términos de rendimiento global.

A pesar de su utilidad, el AUC tiene ciertas limitaciones. Diversos autores han señalado que este índice puede no ser representativo en escenarios donde las clases están muy desbalanceadas (Lobo, Jiménez-Valverde & Real, 2008), como suele ocurrir en modelización de distribución de especies. En tales casos (como el nuestro), el AUC puede mostrar un valor elevado incluso cuando el modelo predice de forma pobre la clase minoritaria, que en este caso es la presencia. Sin embargo, nuestro objetivo con AUC no es utilizar su valor de forma directa si no usarlo como método comparativo para elegir el mejor modelo. Una vez este sea elegido se complementará la evaluación con otros indicadores como la sensibilidad, la especificidad, el índice Kappa o el poder predictivo positivo (PPP).

4.3.1. Sin información espacial

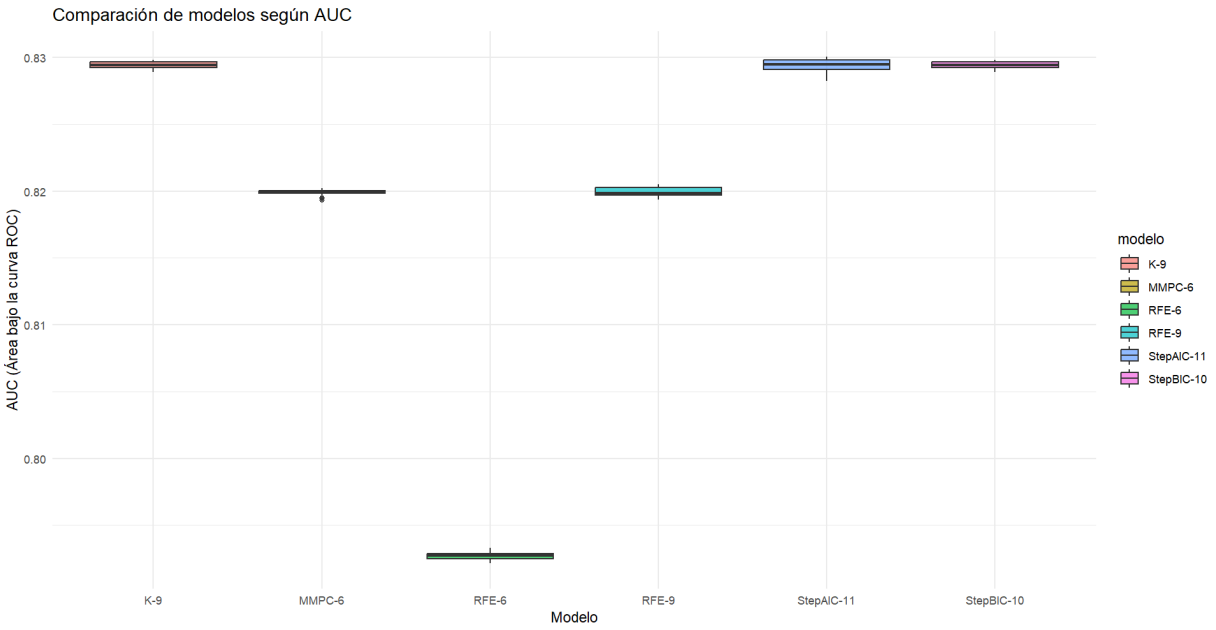

Figura 36: Gráfico Box-Plot, comparación de modelos. Fuente: elaboración propia.

En la figura anterior se presenta un gráfico tipo Box-Plot que compara el rendimiento de los diferentes modelos predictivos basándose en su área bajo la curva cuando no se incluye la información espacial como variable predictora. Antes de nada, recordar lo dicho anteriormente, cuanto mayor sea el área bajo la curva, mejor modelo será. En este tipo de representaciones se pueden observar tres cosas por modelo: la mediana del AUC obtenido (linea central de cada caja), su variabilidad (ancho de la caja) y posibles valores atípicos (puntos). Cada modelo ha sido nombrado por el tipo de técnica utilizada para la selección de sus variables y el número de variables que ha seleccionado. Debemos puntualizar que tanto el modelo StepBIC-9 como K-9 han elegido el mismo conjunto de variables, por tanto se trata realmente del mismo modelo; a partir de ahora obviaremos el segundo.

En general, se observa que los modelos etiquetados como StepBIC-9 y StepAIC-10 muestran los valores más altos de AUC, situándose alrededor de 0.83. Estos valores nos indican una muy buena capacidad discriminativa, lo que implica que estos modelos tienen una alta probabilidad de clasificar correctamente entre observaciones de presencia y ausencia. El modelo StepBIC-9, en particular, destaca por su escasa variabilidad, lo cual refuerza su estabilidad como método de selección de variables. Por tanto, si tuviéramos que elegir entre el AIC y el BIC, teniendo ambos una AUC practicamente idéntica, premiaríamos el BIC por su variabilidad ligeramente menor (su caja es menos ancha).

En el extremo opuesto, el modelo RFE-6 presenta los valores más bajos de AUC, situándose por debajo de 0.8. Este resultado sugiere que, el subconjunto de variables seleccionado usando este método y habiendo cortado en 6 variables, reduce de notablemente el AUC y por tanto, su capacidad predictiva. A pesar de que el algoritmo RFE (Recursive Feature Elimination) es comúnmente utilizado en selección de variables, sus resultados aquí indican que el subconjunto de 6 variables tiene una calidad inferior al resto para capturar adecuadamente el patrón de presencia y ausencia en los datos utilizados.

Recordamos que cuando implementamos el RFE en apartados anteriores escogimos dos subconjuntos, uno con 6 variables (a partir de 6 variables se estabilizaba la curva de

error del RFE) y otro con 9 variables (donde se alcanzaba el mínimo error en la gráfica). Observamos como el subconjunto RFE-9 tiene significativamente mejor AUC que el RFE-6 (sube de 0.795 a 0.82), esto nos indica que para este método de selección, el número óptimo de variables se encontraba donde el error alcanzaba el mínimo. A pesar de esto, sigue siendo peor modelo que los AIC o BIC, por una distancia de 0.01.

Por su parte, los modelos RFE-9 y MMPC-6 se sitúan en un rango intermedio/alto, alrededor de 0.82. Si bien su desempeño no alcanza al de los modelos seleccionados por Stepwise (AIC o BIC), ambos mantienen una capacidad aceptable de clasificación, lo cual podría justificar su uso en escenarios donde se busque priorizar interpretabilidad o reducir complejidad computacional.

En términos generales, podríamos concluir gracias a este gráfico que los métodos basados en criterios de información, como el AIC o el BIC, ofrecen mejores resultados de AUC en ausencia de información espacial, lo cual sugiere que estas estrategias logran identificar subconjuntos de variables altamente informativos para la predicción de la ocurrencia del lobo.

Por último, debemos elegir el modelo óptimo, aquel que haya obtenido mejores métricas, para su posterior modelización. En este caso tanto el StepAIC como el StepBIC tienen una mediana del AUC prácticamente igual, con lo cual es difícil determinar cuál es mejor; sin embargo, observamos como el BIC tiene ligeramente menos variabilidad y una unidad menos de variables, por tanto, por precisión y simplicidad, elegiremos este como modelo óptimo.

4.3.2. Con información espacial

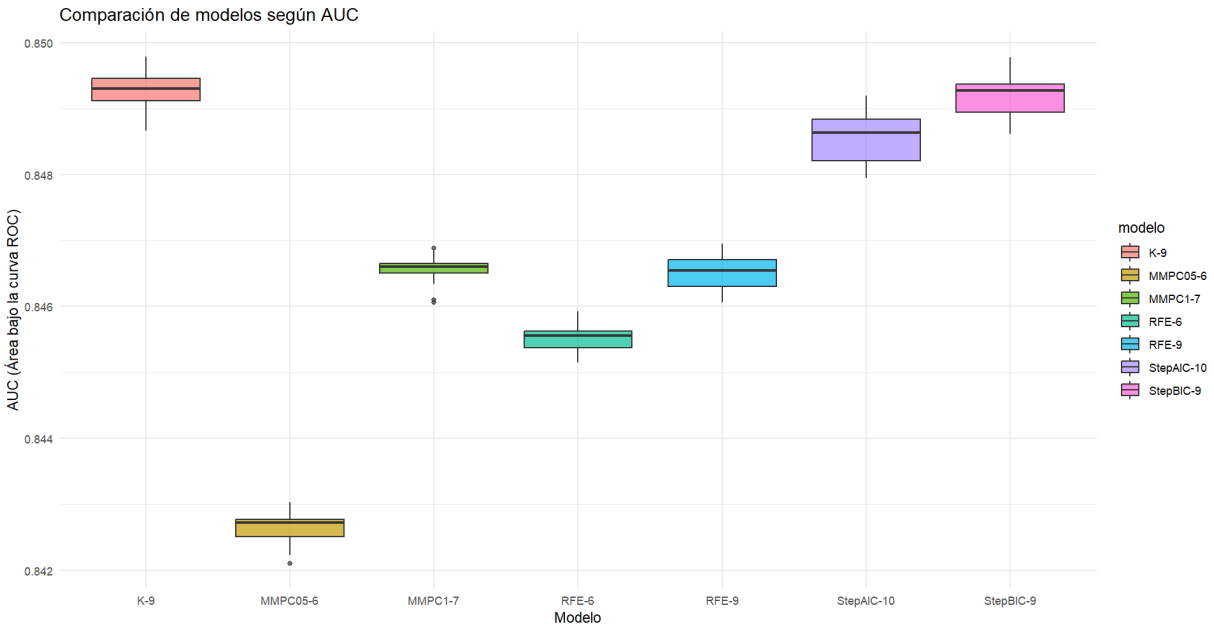

Figura 37: Gráfico Box-Plot, comparación de modelos. Fuente: elaboración propia.

Se muestra la comparación del rendimiento predictivo de distintos modelos de clasificación binaria incorporando información espacial como variable predictora. Como en el caso anterior, se ha utilizado el AUC (Área Bajo la Curva ROC) como métrica de evaluación principal y se representa mediante diagramas de caja para reflejar tanto la mediana como la variabilidad en los resultados.

En esta ocasión, se observa un ligero aumento generalizado del AUC respecto a los modelos que no incluían variables espaciales (de 0.03 puntos), lo que sugiere que la incorporación de dicha información ha contribuido a una mejor discriminación entre presencia y ausencia de las especies modeladas. Esta mejora es consistente con numerosos estudios que subrayan la importancia de variables espaciales o autocorrelación espacial en el modelado de distribución de especies (Dormann et al., 2007).

En este caso los modelos StepBIC-9 y K-9 son los más altos (en este caso no son coincidentes), seguidos muy de cerca por StepAIC-10 (siendo este el que presenta mayor variabilidad, es decir, caja más ancha), con medianas por encima de 0.849 y de 0.848 para el AIC. Los tres están basados en procedimientos de selección de variables stepwise guiados por criterios de información, lo cual refuerza la eficacia de estos métodos a la hora de capturar la complejidad del fenómeno ecológico modelado. Su rendimiento constante tanto con como sin variables espaciales subraya su fiabilidad como herramienta robusta y explicativa.

En un rango intermedio se encuentran los modelos RFE-9, 6 y MMPC1-7, con valores de AUC alrededor de 0.846–0.847. Aunque sean resultados inferiores, siguen considerándose buenos métodos de predicción. Estos resultados, aunque algo inferiores, siguen considerándose buenos, aunque tal vez no tan ajustados como los modelos stepwise o el K-9.

Por otro lado, el modelo MMPC05-6 muestra el peor desempeño, con un AUC cercano a 0.843 y una mayor dispersión de los resultados.

En resumen, la incorporación de variables espaciales ha tenido un efecto positivo generalizado sobre el rendimiento de los modelos, y los enfoques basados en BIC y AIC siguen liderando tanto en capacidad predictiva como en estabilidad. Esta mejora refuerza la hipótesis de que los patrones de distribución del lobo no solo dependen de condiciones ambientales directas, sino también de componentes espaciales asociados al hábitat o a procesos ecológicos subyacentes.

Por último, debemos elegir el modelo óptimo, aquel que haya obtenido mejores métricas, para su posterior modelización. En este caso tanto el StepK como el StepBIC tienen una mediana del AUC prácticamente igual, con lo cual es difícil determinar cuál es mejor; sin embargo, observamos como el StepK tiene ligeramente menos variabilidad una mediana más centrada, por tanto, elegiremos este como modelo óptimo.

Como comparación final, hemos incluido un gráfico que contiene en un BoxPlot todos los modelos anteriormente mencionados; para comparar así la contribución de la información espacial al AUC en modelos de regresión.

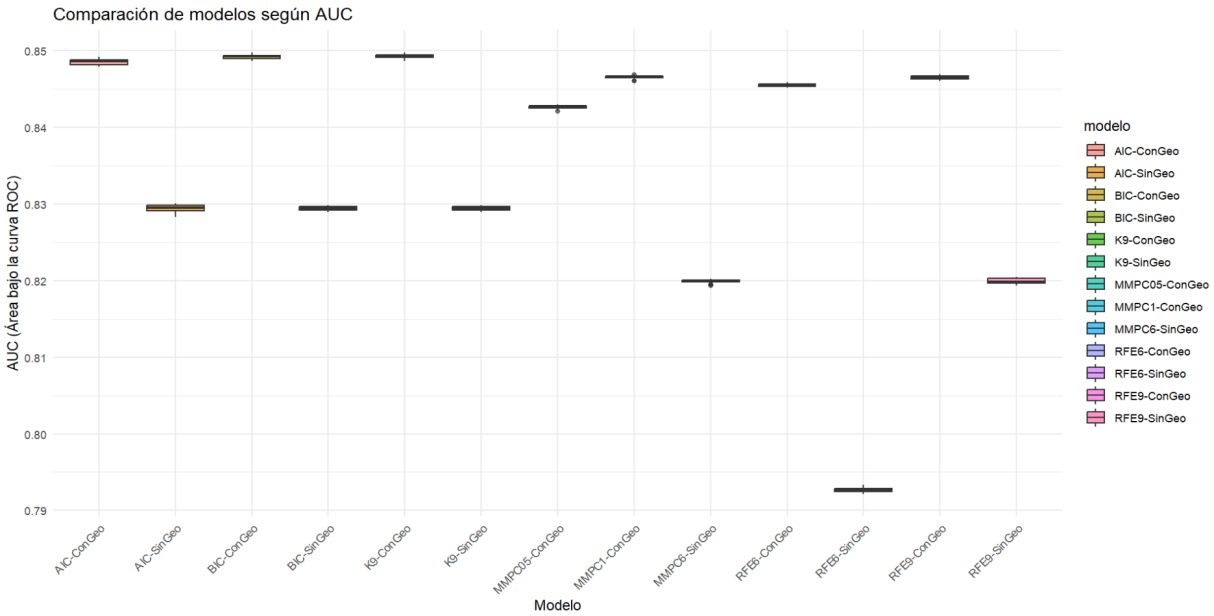

Figura 38: Gráfico Box-Plot, comparación de modelos. Fuente: elaboración propia.

Los modelos con información espacial (ConGeo) tienden a tener AUC ligeramente más altos que sus versiones sin información espacial (SinGeo), lo que sugiere que incluir variables espaciales mejora la capacidad predictiva.

Dentro de los modelos evaluados, AIC-ConGeo, BIC-ConGeo, y K9-ConGeo presentan los AUC más altos, situándose cerca de 0.85. Esto indica un buen rendimiento. En contraste, algunos modelos como RFE6-SinGeo, RFE9-SinGeo o MMPC6-SinGeo muestran AUC más bajos (cerca de 0.79 el primero y de 0.82 los restantes).

Aunque el gráfico incluye técnicas de selección de variables y criterios como AIC, BIC, RFE y MMPC, no hay una diferencia muy grande entre los métodos más sólidos (especialmente los que conservan variables geográficas), lo que refuerza la robustez del modelo en presencia de estas variables.

5. Modelización

En esta sección se describen los distintos algoritmos de modelización utilizados para estimar la probabilidad de presencia de la especie. A partir de los subconjuntos de variables seleccionados mediante diferentes criterios anteriormente, se construyeron modelos utilizando algoritmos de aprendizaje automático, entre los que se incluyen Random Forest, Gradient Boosting XgBoost y CatBoost.

Estas técnicas permiten capturar relaciones complejas y no lineales entre las variables predictoras y la variable respuesta, ofreciendo una elevada capacidad predictiva incluso en contextos con multicolinealidad o efectos de interacción. Además, algunos de estos algoritmos permiten obtener medidas de importancia de variables, lo cual resulta útil para la interpretación ecológica del modelo.

Las siguientes secciones detallan la configuración general, el proceso de entrenamiento y las particularidades de cada algoritmo empleado, así como las métricas de evaluación utilizadas para su validación. No obstante, solo se desarrollará en profundidad aquel modelo que obtenga el mejor rendimiento en la comparación final en un último Box-Plot, con el objetivo de centrar el análisis en la aproximación más robusta y eficaz.

5.1. Random Forest

El algoritmo Random Forest (Bosque Aleatorio), propuesto por Leo Breiman en 2001, es una técnica de aprendizaje automático de tipo ensamblado basada en árboles de decisión. Su objetivo principal es mejorar la precisión y estabilidad de un árbol de decisión individual; esto lo consigue combinando múltiples árboles, los cuales han sido creados a partir de subconjuntos aleatorios de nuestros datos y de nuestras variables predictoras. Esta estrategia lo convierte en un modelo robusto, no paramétrico y altamente eficaz tanto en tareas de clasificación como de regresión (Breiman, 2001). Por tanto, antes de explicarlo debemos entender qué es un árbol de decisión.

Un arbol de decisión es un algoritmo de aprendizaje supervisado cuya principal característica es que divide progresivamente el espacio de los datos en subgrupos más homogéneos mediante una estructura jerárquica en forma de árbol.

Se parte de un nodo raíz tiene todo nuestro conjunto de datos y se van realizando divisiones o ramificaciones a partir de preguntas o condiciones hechas sobre nuestras variables; en cada nodo, una vez elegida la variable óptima y el punto de corte óptimo, los datos se dividen en dos ramas (por ejemplo, ¿La temperatura media anual es mayor o menor a 15 grados? En caso de que sí, se ramifica hacia un lado, en caso de que no, hacia otro). Estas condiciones generan nodos internos, que continúan ramificándose hasta que se cumplan algunas de las condiciones preestablecidas (tamaño mínimo de observaciones por nodo, profundidad máxima de árbol...) y se llega a los nodos terminales u hojas, donde se toma una decisión final.

El resultado será una predicción final donde en modelos de clasificación cada hoja vota por la clase mayoritaria de los datos que contiene y en regresión se predice el promedio de la variable objetivo en la hoja.

Los árboles tienen muchas ventajas, son fáciles de entender e interpretar (se asemeja a una secuencia lógica de preguntas), no requiere normalización de los datos, puede manejar tanto variables numéricas como categóricas entre otras. Sin embargo, tiende a sobreajustarse fácilmente si no se limita su crecimiento (overfitting), tienen poca estabilidad (por ejmplo, cambios leves en los datos pueden generar árboles muy distintos), etc.

Por estas razones, en la práctica, para solucionar o mejorar esta limitaciones de los árboles se prefiere recurrir a métodos que combinan múltiples árboles de decisión, como es el caso del algoritmo Random Forest. Esta técnica entrena muchos árboles de decisión juntos y combina sus predicciones, mejorando la estabilidad y la precisión que tendian los árboles por individual. Así, Random Forest reduce la variabilidad y el riesgo de sobreajuste, conservando las principales ventajas de los arboles, como la capacidad de interpretar la importancia relativa de las variables predictoras. Esta técnica, utilizada en el este trabajo, ofrece una solución eficaz y robusta para problemas de clasificación binaria como la predicción de presencia o ausencia de especies, proporcionando resultados más estables y fiables que los obtenidos mediante árboles aislados.

Este conjunto de múltiples árboles, también llamado bosque, se construye siguiendo las siguientes premisas:

- Bootstrap: para cada árbol, se selecciona una muestra aleatoria con reemplazo (bootstrapping) del mismo tamaño que el dataset original. Algunos datos se repiten y otros quedan fuera (estos datos se llaman out-of-bag o OOB). Esto introduce variedad entre los árboles y ayuda a reducir el sobreajuste.

- Construcción de árboles de decisión: Para cada subconjunto seleccionado en el paso anterior se crea un arbol de decisión. En cada nodo del árbol, en lugar de evaluar todas las variables disponibles, se selecciona de forma aleatoria un subconjunto de variables candidatas. El mejor punto de corte se elige únicamente dentro de ese subconjunto aleatorio. Esta doble aleatorización (tanto de los datos como de las variables) es lo que le da el nombre al algoritmo y contribuye a su diversidad interna, evitando que todos los árboles sean parecidos.

- Crecimiento completo de los árboles: Este procedimiento se repite tantas veces como árboles se hayan especificado (ntree). El número de árboles es un parámetro clave del modelo, cuanto más grande sea este mejor será la estabilidad del bosque hasta alcanzar el punto de saturación.

- Agregación por mayoría o promedio: en clasificación, la predicción final es la clase más frecuente entre los árboles (votación mayoritaria); en regresión, se promedia la salida de todos los árboles.

- Cálculo de la importancia de las variables predictoras: El modelo permite obtener dos principales métricas que estiman el aporte de cada variable a la calidad de la predicción: %IncMSE (ya explicado) e IncNodePurity, el cual indica cuánto mejora la pureza de los nodos al utilizar una determinada variable como criterio de división. Estas medidas permiten interpretar indirectamente la relevancia de las variables, incluso si el modelo en sí no es directamente interpretable.

Para programar el Random Forest debemos ajustar diversos parámetros, los cuales, con un ajuste adecuado pueden mejorar el rendimiento del modelo. Los tres principales son: ntree (número de árboles, a mayor número de árboles, mayor estabilidad y menor varianza del modelo, aunque con mayor coste computacional), mtry (número de variables consideradas en cada división, si es pequeño, los árboles serán más distintos entre sí, y si es grande, los árboles tienden a parecerse más y el modelo puede sobreajustar) y nodesize (tamaño mínimo del nodo final, si es pequeño permite árboles más profundos y complejos, mientras que uno mayor genera árboles más simples).

En el caso particular de este trabajo, se optó por evaluar tres modelos diferentes con Random Forest con el objetivo de analizar su comportamiento bajo distintos enfoques de selección de variables y evaluar su robustez y capacidad predictiva en distintos contextos.

En primer lugar, se entrenó un modelo Random Forest utilizando el subconjunto de variables que produjo el mejor valor de AUC durante el proceso de selección. Esta configuración representa el escenario donde el modelo ha sido ajustado específicamente para maximizar su capacidad de discriminación entre clases.

En segundo lugar, se aplicó Random Forest utilizando el subconjunto con máximo número de variables obtenido en el apartado de criterios de selección. Si este conjunto coincide con el óptimo, entonces elegiremos otro con muchas variables.

En segundo lugar, se aplicó el Random Forest sobre un subconjunto creado a partir de la estimación de un modelo Random Forest con todas las variables disponibles. Esta configuración permite explotar una de las principales ventajas del algoritmo: su robustez frente al sobreajuste y su capacidad para manejar conjuntos de datos con muchas variables, incluso cuando algunas de ellas pueden ser irrelevantes o colineales. Además, este modelo resulta especialmente útil para calcular la importancia relativa de las variables (%IncMSE, IncNodePurity), información que puede ser clave para la interpretación ecológica del modelo.

La comparación entre estas tres aproximaciones permite valorar no solo qué modelo obtiene mejores métricas de rendimiento, sino también cuál ofrece mayor estabilidad, interpretabilidad o relevancia ecológica. Este enfoque multidimensional en el análisis de Random Forest sigue recomendaciones presentes en la literatura especializada (Breiman, 2001; Kuhn & Johnson, 2013), donde se enfatiza la necesidad de considerar simultáneamente la precisión, la complejidad del modelo y la utilidad interpretativa en contextos aplicados como el de la modelización de la distribución de especies.

Los pasos a seguir a la hora de programar serán los siguientes:

- Paso 1: Obtener el número óptimo de arboles que debemos utilizar (ntree). Se visualiza el error OBB conforme aumenta el número de árboles y se elige el número de árboles óptimo a partir del cual se estabiliza este error.

- Paso 2: A continuación, para cada uno de los tres subconjuntos de variables que hemos definido, se realiza un entrenamiento de Random Forest utilizando la función train() de caret, donde se evalúan distintas configuraciones del parámetro mtry. Se explora un rango de valores para determinar cuál proporciona el mejor rendimiento, medido por AUC, y se devuelve por pantalla qué valor de mtry produce el mayor valor de AUC. Este valor se almacena como el óptimo para ese conjunto de variables y lo usaremos en los pasos posteriores para validar el modelo.

- Paso 3: Se define una función para repetir el entrenamiento de cada modelo bajo validación cruzada, con múltiples repeticiones (a elección del programador, en este caso se han realizado 10 repeticiones de 4 particiones). Esta función aplica exactamente el mismo algoritmo Random Forest a cada subconjunto de variables, utilizando el valor de mtry óptimo para cada uno de los modelos (calculado previamente), así como el número de árboles obtenido en el paso 1 y el tamaño mínimo de nodo definido por defecto (en este caso 10).

- Paso 4: Para cada subconjunto de variables, se calcula la media de los AUCs obtenidos en las repeticionesde cada grupo, para su posterior comparación en un Box-Plot.

5.1.1. Sin información espacial

Los tres modelos sobre los que aplicaremos el Random forest sin información espacial son:

El modelo con mejores resultados de AUC durante el proceso de selección, en este caso el BIC:

`c("bio1", "viento_mean", "bio12", "presion_humana", "porc_bosque", "bio6", "bio4", "viento_relativo", .elevacion")`

El modelo con mayor número de variables, en este caso el AIC:

`c("bio1", "viento_mean", "bio12", "presion_humana", "porc_bosque", "bio6", "bio4", "viento_relativo", .elevacion", "densidad")`

Un modelo creado aplicando un random forest sobre todo el conjunto de variables:

Cuadro 6: Importancia de variables según Random Forest

Variable	**%IncMSE**	**IncNodePurity**
bio1	0.0564	224.93
bio4	0.0335	132.19
bio5	0.0268	150.68
accesibilidad	0.0245	126.99
bio12	0.0234	117.69
bio6	0.0229	81.34
elevacion	0.0199	90.98
viento_mean	0.0183	122.13
indice_habitat	0.0180	77.05
viento_relativo	0.0172	143.65
porc_bosque	0.0158	85.41
densidad	0.0133	72.56
presion_humana	0.0096	83.99

Y el siguiente grágico:

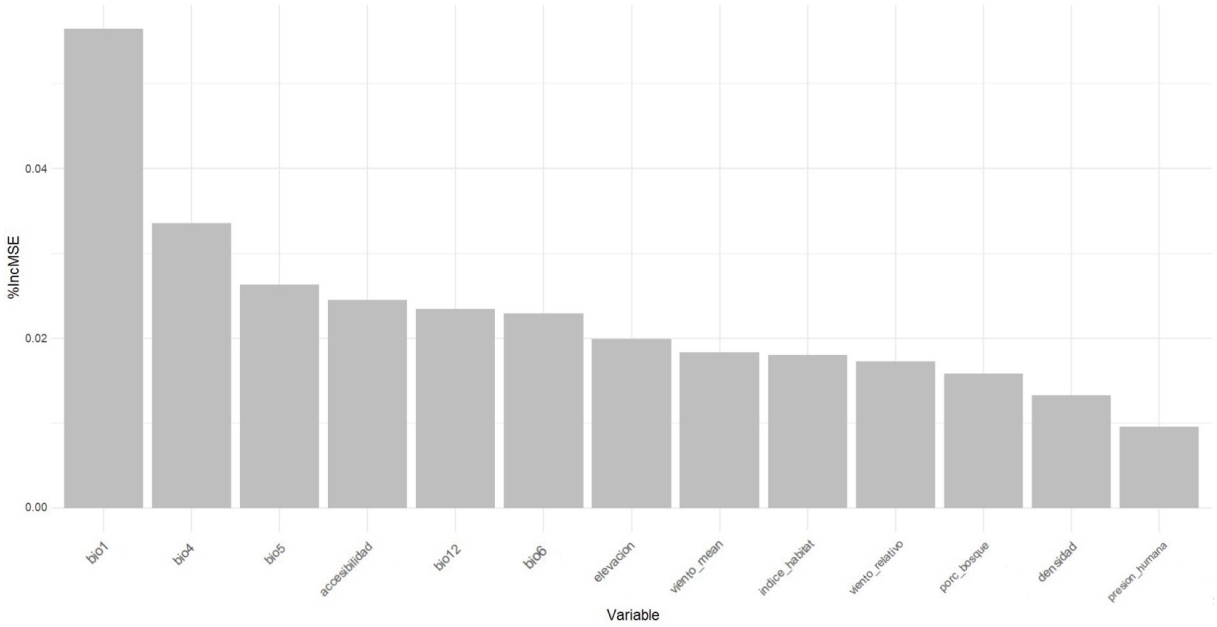

Figura 39: Gráfico de barras del Incremento del error en un Random Forest. Fuente: elaboración propia.

Se observa que la variable más importante con diferencia es `bio1`. A partir de esta, la importancia de las variables disminuye progresivamente. Dado que el modelo Random Forest se ha ajustado utilizando todas las variables disponibles, la selección del subconjunto final se ha realizado mediante la inspección de la tabla y el gráfico de importancias, identificando visualmente un punto de inflexión.

Existen diversas estrategias para determinar este corte; en este caso, se ha decidido seleccionar las variables hasta el punto en el que la curva de importancia comienza a estabilizarse, lo cual ocurre aproximadamente en la variable `viento_relativo`. En total, se han seleccionado 10 variables.

```
   c("bio1", "bio4", "bio5", .ªccesibilidad", "bio12", "bio6"
.elevacion", "viento_mean", ïndice_habitat", "viento_relativo")
```

El primer paso definidos ya los modelos es calcular el número óptimo de árboles. Para ello, entrenamos un modelo preliminar con todas las variables predictoras y visualizamos un gráfico que muestra cómo se comporta el error OOB a medida que se agregan más árboles al modelo. Suele observare una curva que disminuye rapidamente y luego tiende a estabilizarse. El punto o zona donde la curva se estabiliza indica a partir de cuántos árboles añadir más ya no mejora el rendimiento significativamente. Si se eligen pocos árboles, esto puede resultar en un modelo inestable, sin embargo, si se usan demasiados, el coste computacional aumenta sin sentido, ya que no se ganaría en precisión.

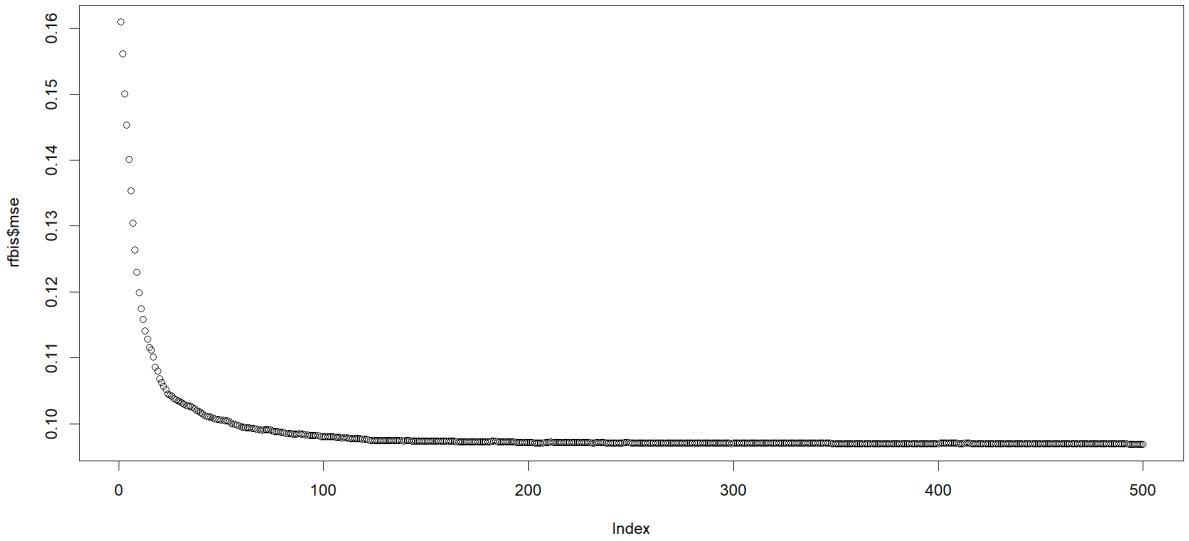

Figura 40: Gráfico del error OBB. Fuente: elaboración propia.

Se comprueba lo dicho anteriormente, el error baja drásticamente con el aumento del número de árboles para luego estabilizarse hasta 500. Observamos cómo de 0 a 100 árboles el error se reduce a grandes pasos, de 100 a 200 árboles continúa bajando, aunque a mucha menor velocidad y a partir de los 200 árboles el error se estabiliza, sin bajar de forma significativa y manteniendose en una linea practicamente recta hasta los 500.

Por tanto, concluiremos que el número óptimo de árboles para utilizar en nuestro random forest y a partir de el cual el error OBB se estabiliza es 200.

El segundo parámetro que debemos tunear es el mtry. Como ya se ha explicado, entrenaremos de nuevo random forest, usando validación cruzada y evaluando el rendimiento para cada valor de mtry (probaremos para cada modelo desde 2 variables hasta el número de variables del modelo).

Para el modelo BIC el número óptimo del mtry ha sido 3, para el AIC ha sido 2 y para el random forest completo ha sido 6. Se entiende que para este número de mtry se maximiza Accuracy y Kappa.

Una vez hemos definido el número óptimo tanto de árboles como de variables, procedemos a comparar el rendimiento de los tres modelos con un Box-Plot:

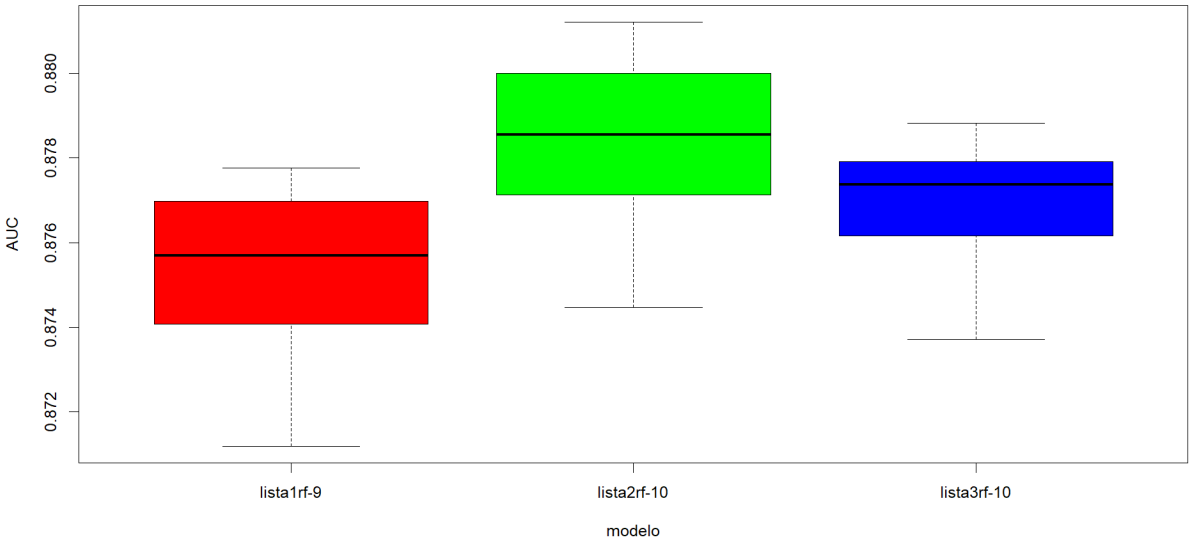

Figura 41: Gráfico Box-Plot comparativo de los tres modelos. Fuente: elaboración propia.

Observamos tres cajas (una por modelo), en un gráfico que acotado por un rango de error de entre 0.872 y 0.88. Tenemos en rojo el modelo BIC, en verde el AIC y en azul el random forest completo. A pesar de tener los tres modelos un AUC muy cercano, nuestra misión es quedarnos con el mejor; en este caso, el modelo con mayor AUC es sin duda el modelo AIC (el verde), con una mediana cercana a 0.878. Como observación, decir que el modelo con menor variabilidad es el random forest, con una caja ligeramente menos ancha.

Por tanto, el modelo óptimo que eligiríamos sería el modelo AIC.

5.1.2. Con información espacial

Los tres modelos sobre los que aplicaremos el Random forest con información espacial son:

El modelo con mejores resultados de AUC durante el proceso de selección, en este caso el StepK:

```
c("porc_vecinos_con_lobo", "bio1", "viento_mean", "bio12",
"presion_humana", "bio6", "bio4", "viento_relativo", .elevacion")
```

El modelo con mayor número de variables, en este caso el AIC:

```
c("porc_vecinos_con_lobo", "bio1", "viento_mean", "bio12",
"presion_humana", indice_habitat", "bio6", "bio4", "viento_relativo",
.elevacion", "porc_bosque")
```

Un modelo creado aplicando un random forest sobre todo el conjunto de variables:

Cuadro 7: Importancia de variables según Random Forest

Variable	**%IncMSE**	**IncNodePurity**
porc_vecinos_con_lobo	0.0657	399.42
bio1	0.0497	118.66
bio5	0.0346	113.93
bio12	0.0298	110.17
bio6	0.0297	67.85
bio4	0.0271	94.06
elevacion	0.0233	72.75
accesibilidad	0.0220	89.40
viento_relativo	0.0193	111.48
viento_mean	0.0146	74.94
indice_habitat	0.0140	62.96
densidad	0.0127	61.93
porc_bosque	0.0123	75.88
presion_humana	0.0093	69.48

Acompañada del siguiente gráfico:

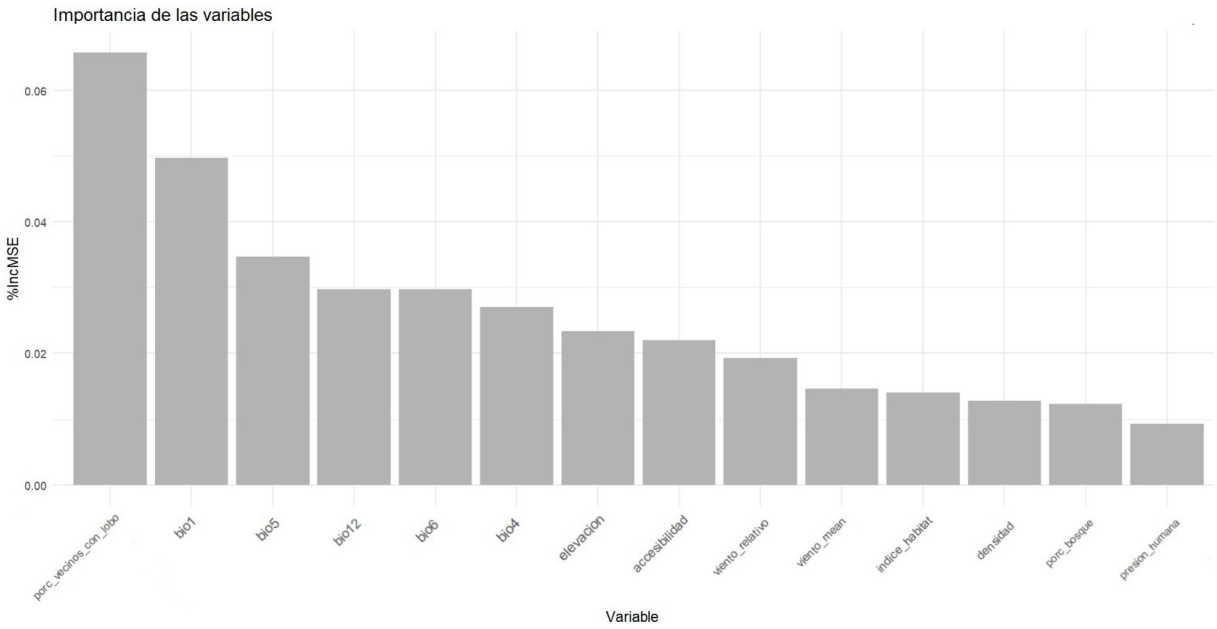

Figura 42: Gráfico de importancia de variables en un Random Forest. Fuente: elaboración propia.

Se observa que la variable más importante con diferencia es `porc_vecinos_con_lobo`. A partir de esta, la importancia de las variables disminuye progresivamente. Dado que el modelo Random Forest se ha ajustado utilizando todas las variables disponibles, la selección del subconjunto final se ha realizado mediante la inspección de la tabla y el gráfico de importancias, identificando visualmente un punto de inflexión.

Existen diversas estrategias para determinar este corte; en este caso, se ha decidido seleccionar las variables hasta el punto en el que la curva de importancia comienza a estabilizarse, lo cual ocurre aproximadamente a partir de la variable `viento_mean`, inclusive. En total, se han seleccionado 9 variables.

`c("porc_vecinos_con_lobo", "bio1", "bio5", "bio12", "bio6", "bio4", .elevacion", .accesibilidad", "viento_relativo")`

De nuevo, lo primero que se hará es buscar el número óptimo de árboles en el gráfico OBB:

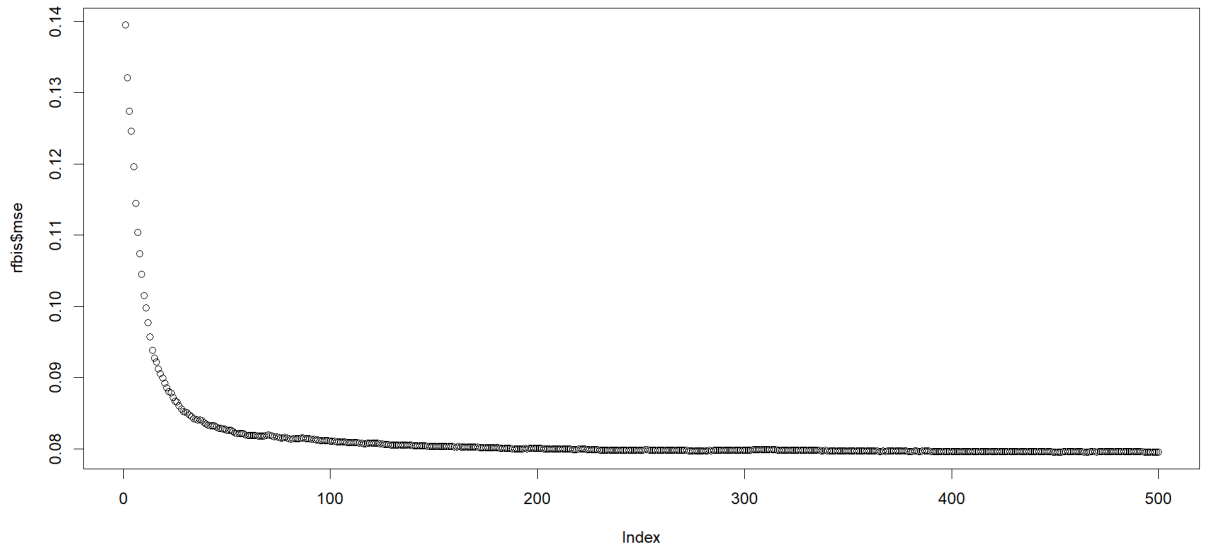

Figura 43: Gráfico del error OBB. Fuente: elaboración propia.

De nuevo, el error se estabiliza a partir de los 200 árboles.

El segundo parámetro que debemos tunear es el mtry. Para el modelo K-9, el número óptimo del mtry ha sido 2, para el AIC ha sido 5 y para el random forest completo ha sido 3. Se entiende que para este número de mtry se maximiza Accuracy y Kappa.

Una vez hemos definido el número óptimo tanto de árboles como de variables, procedemos a comparar el rendimiento de los tres modelos con un Box-Plot:

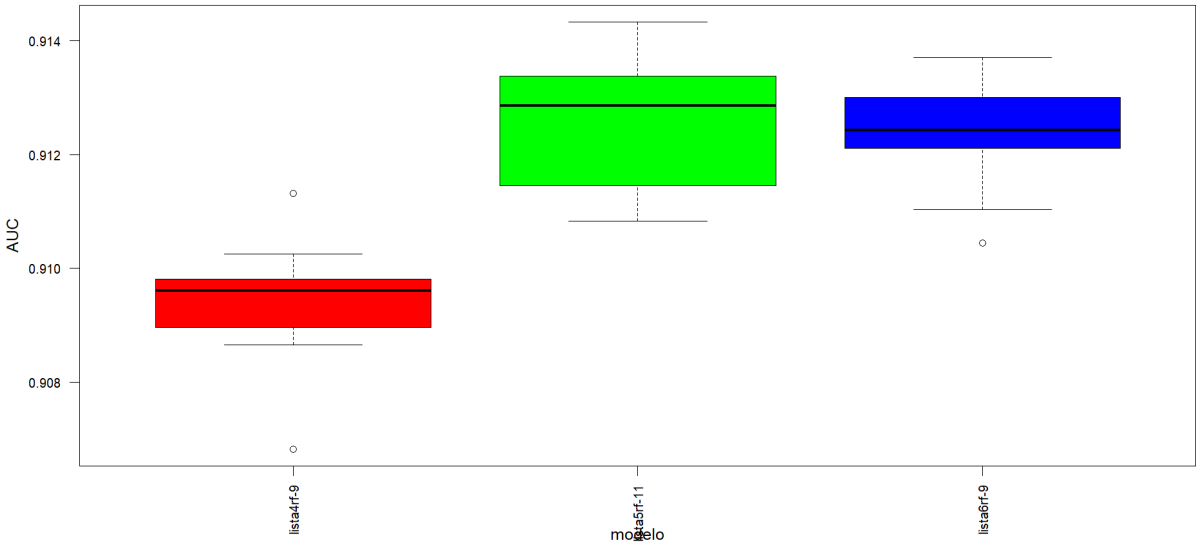

Figura 44: Gráfico Box-Plot comparativo de los tres modelos. Fuente: elaboración propia.

Para este segundo Box-Plot de los modelos con información espacial observamos de nuevo 3 cajas. La caja roja representa el modelo k-9, el cual presenta el menor rendimiento (AUC más bajo); la caja verde representa el modelo AIC y tiene el AUC más alto (linea media), pero la varianza más grande con diferencia (tiene la caja más ancha); y por último la caja azul, que representa el modelo creado a partir de un random forest, tiene AUC muy cercano al AIC .

Para la elección del mejor modelo, si nos fijáramos sólo en la media del AUC, la elección sería el modelo AIC; sin embargo, el modelo random forest tiene un AUC muy cercano y una varianza o ancho de caja mucho menor. Por todo esto, y con el objetivo de mantener un equilibro entre estabilidad y complejidad, el modelo elegido como óptimo ha sido el Random Forest.

5.2. Gradient Boosting

Aunque el algoritmo Random Forest ha demostrado ser una herramienta robusta y eficaz para problemas de clasificación y regresión, presenta ciertas limitaciones. Una de las principales debilidades de Random Forest es que todos los árboles del conjunto son generados de forma independiente, sin considerar los errores que generan los árboles anteriores. Esto implica que el modelo trata con el mismo nivel de importancia tanto los aciertos como los fallos, lo que puede limitar su capacidad para refinar las predicciones en casos más complejos o con patrones sutiles (Kuhn & Johnson, 2013).

Además, aunque Random Forest tiende a reducir la varianza del modelo mediante la utilización de múltiples árboles, puede no ser suficientemente flexible para capturar relaciones no lineales finas o interacciones complejas entre variables, a no ser que aumentaramos enormemente el número de árboles utilizados, en cuyo caso aumentaría el coste computacional. Asimismo, el algoritmo no incluye un mecanismo explícito para centrarse en las observaciones más difíciles de predecir, lo que puede ser relevante en contextos con clases desbalanceadas, como en nuestro caso.

Frente a estas limitaciones, el algoritmo Gradient Boosting surge como una alternativa potente. Su principal característica es que los árboles se construyen de forma secuencial

uno detrás de otro, y cada uno intenta predecir y corregir los errores cometidos por el árbol anterior, lo que permite un mejor aprendizaje. Cada árbol aprende de los errores del anterior reduciendo los del actual (Friedman, 2001). Esta estrategia convierte a Gradient Boosting en una técnica altamente eficaz en tareas donde se requiere precisión elevada, siendo necesaria antes una calibración de los hiperparámetros.

Para el algoritmo Grading Boosting, debemos tener en cuenta principalmente 2 hiperparámetros:

- *interaction.depth* Define la profundidad máxima que puede tener cada árbol de decisión, profundidades bajas generan modelos más simples y menos propensos al sobreajuste, pero con menor capacidad de capturar relaciones complejas. Por el contrario, profundidades altas permiten crear modelos más complejos, pero aumenta el riesgo de sobreajuste.

- *shrinkage* También llamado learning rate, controla cuánto contribuye cada nuevo árbol al modelo final. Se trata de un factor de amortiguación que regula la magnitud del ajuste que realiza cada iteración sobre los errores del modelo previo. Cuando el modelo entrena un nuevo árbol para corregir los errores, su aporte se multiplica por el valor del shrinkage antes de sumarlo al modelo acumulado. Por ejemplo, si shrinkage equivale a 1, cada árbol corrige el error completamente (se trataría de un modelo muy agresivo). Un shrinkage alto (0.3 – 1.0) equivale a un aprendizaje rápido, pero aumenta mucho riesgo de sobreajuste. Un shrinkage bajo (0.01 – 0.1) tiene un aprendizaje más lento y controlado, pero mejor generalización si se entrena con muchos árboles.

En términos de código, primero se define una rejilla de valores (grid) con distintas combinaciones de los parámetros más relevantes del algoritmo: la tasa de aprendizaje (shrinkage), el número de árboles (n.trees), el número mínimo de observaciones en cada nodo final (n.minobsinnode), y la profundidad de los árboles (interaction.depth). Estos valores se combinan, permitiendo comparar el rendimiento del modelo en múltiples configuraciones.

El modelo es evaluado mediante un proceso de validación cruzada con cuatro particiones, lo que significa que los datos se dividen en cuatro bloques: en cada iteración, tres se usan para entrenar y uno para validar, rotando hasta cubrir todos. Al finalizar el proceso, se identifican los valores óptimos de los parámetros que maximizan el AUC. Además, el código sacará por pantalla el valor óptimo de cada parámetro, lo que haremos será quedarnos con el valor óptimo de n.minobsinnode y recodificar, fijando este valor y modificando el resto de parámetros con nuevas combinaciones.

5.2.1. Modelo sin información espacial

Tanto el Grading Boosting como los algoritmos de aprendizaje automático que utilizaremos serán realizados sobre el modelo que hemos determinado como óptimo en el conjunto sin información espacial:

```
   c("bio1", "viento_mean", "bio12", "presion_humana", "porc_bosque",
"bio6", "bio4", "viento_relativo", .elevacion", "densidad")
```

Gradient Boosting se ha aplicado sobre este y el resultado es la siguiente gráfica:

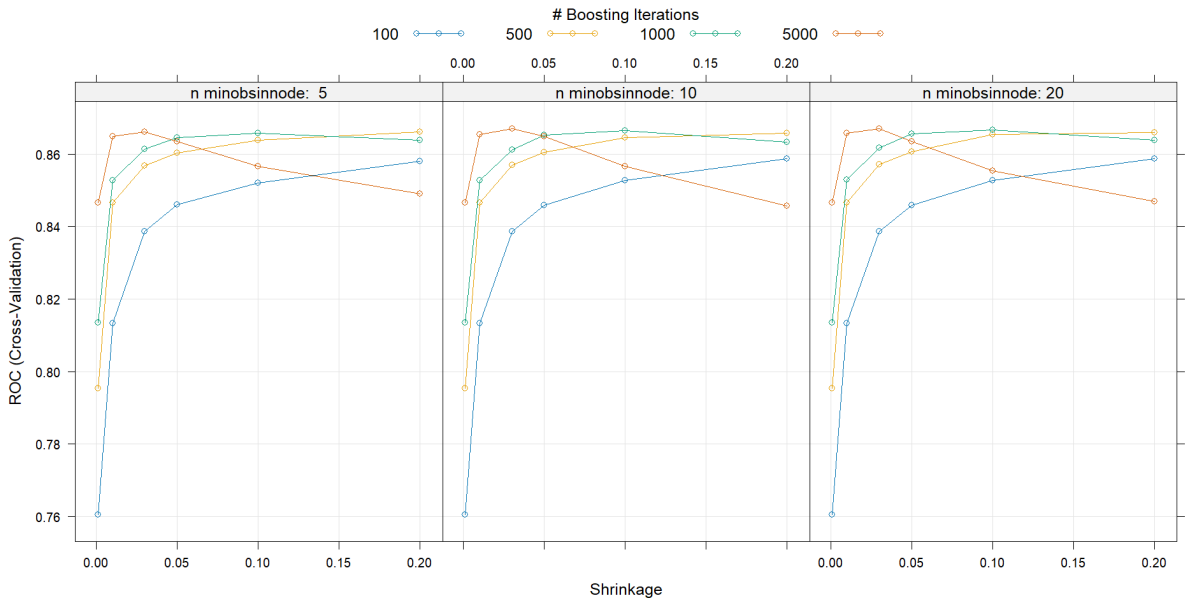

Figura 45: Gráfico Gradient Boosting. Fuente: elaboración propia.

Observamos tres gráficos, para 3 valores diferentes de número mínimo de observaciones en cada nodo final. La linea azul representa el modelo entrenado con 100 árboles, la linea amarilla 500 árboles, la verde 1000 árboles y por último la naranja, de 5000 árboles. Por otro lado, el eje de abscisas representa los diferentes niveles de shrinkage, mientras que el de ordenadas representa la curva ROC. Podríamos intentar sacar conclusiones de los tres gráficos para decidir cual es mejor pero la realidad es que los tres se asemejan mucho, es el propio código el que nos saca por pantalla donde se maximiza.

El código ha determinado que los parámetros que maximizan la curva ROC y por tanto el área bajo la curva son $n.trees = 5000$, $interaction.depth = 2$, $shrinkage = 0{,}03$ y $n.minobsinnode = 10$. Lo que se hará a continuación será, como hemos dicho, fijar el valor de $n.minobsinnode$ en 10, de forma que el resultado será solamente una gráfica, no tres. Respecto a los otros parámetros, incluiremos los óptimos pero también otros diferentes, para un mayor contraste.

El resultado ha sido la siguiente gráfica:

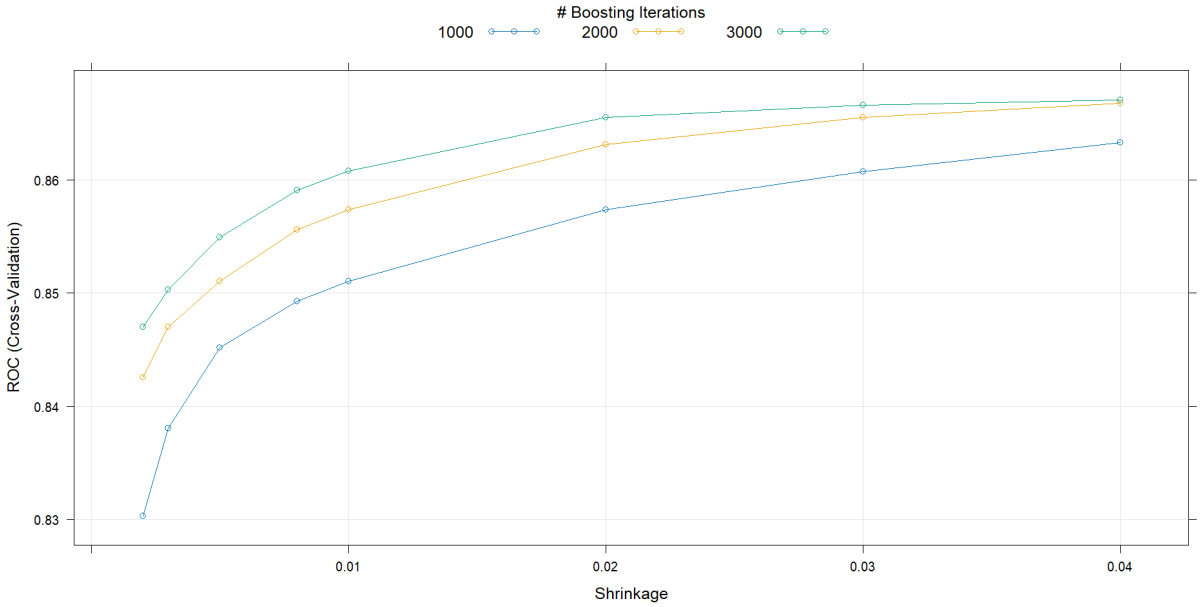

Figura 46: Gráfico Gradient Boosting. Fuente: elaboración propia.

En este gráfico es más sencillo sacar conclusiones. Observamos como claramente el número de árboles que genera valores más altos de ROC son 3000 árboles, mientras que 1000 y 2000 se quedan ligeramente por debajo. El código ha determinado que los parámetros que maximizan la curva ROC y por tanto su área bajo la curva son: $n.trees = 3000$, $interaction.depth = 2$, $shrinkage = 0{,}04$ y $n.minobsinnode = 10$. Estos valores los guardaremos como configuración final del modelo. Posteriormente, se utilizarán para entrenar un modelo definitivo sobre todo el conjunto de datos, que será evaluado y comparado directamente con otros modelos avanzados que explicaremos más adelante. De este modo, se garantiza que la comparación entre algoritmos se realice sobre modelos que ya hemos previamente optimizado.

5.2.2. Modelo con información espacial

Tanto el Grading Boosting como los algoritmos de aprendizaje automático que utilizaremos serán realizados sobre el modelo que hemos determinado como óptimo en el conjunto con información espacial:

```
c("porc_vecinos_con_lobo", "bio1", "bio5", "bio12", "bio6",
"bio4", ."elevacion", ."accesibilidad", "viento_relativo")
```

Gradient Boosting se ha aplicado sobre este y el resultado es la siguiente gráfica:

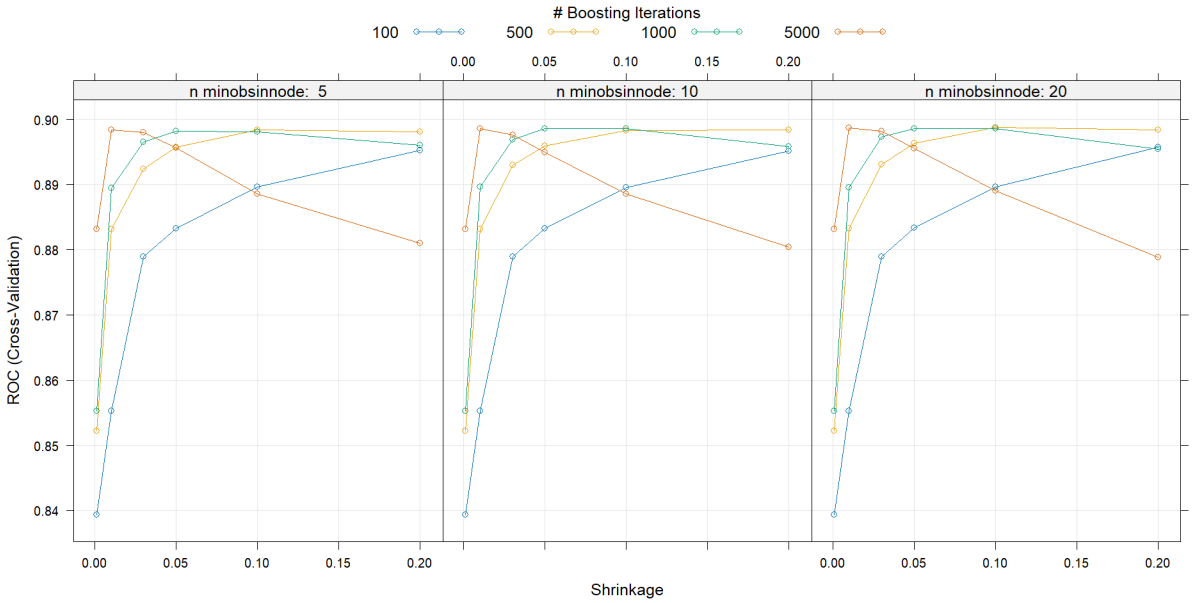

Figura 47: Gráfico Gradient Boosting. Fuente: elaboración propia.

Observamos tres gráficos, para 3 valores diferentes de número mínimo de observaciones en cada nodo final. La línea azul representa el modelo entrenado con 100 árboles, la línea amarilla 500 árboles, la verde 1000 árboles y por último la naranja, de 5000 árboles. Por otro lado, el eje de abscisas representa los diferentes niveles de shrinkage, mientras que el de ordenadas representa la curva ROC. Podríamos, de nuevo, intentar sacar conclusiones de los tres gráficos para decidir cuál es mejor, pero la realidad es que los tres se asemejan mucho. Es el propio código el que nos saca por pantalla donde se maximiza.

El código ha determinado que los parámetros que maximizan la curva ROC y por tanto el área bajo la curva son $n.trees = 500$, $interaction.depth = 2$, $shrinkage = 0{,}1$ y $n.minobsinnode = 20$. Lo que se hará a continuación será, de nuevo, fijar el valor de $n.minobsinnode$ en 20, de forma que el resultado será solamente una gráfica, no tres. Respecto a los otros parámetros, incluiremos los óptimos pero también otros diferentes, para un mayor contraste.

El resultado ha sido la siguiente gráfica:

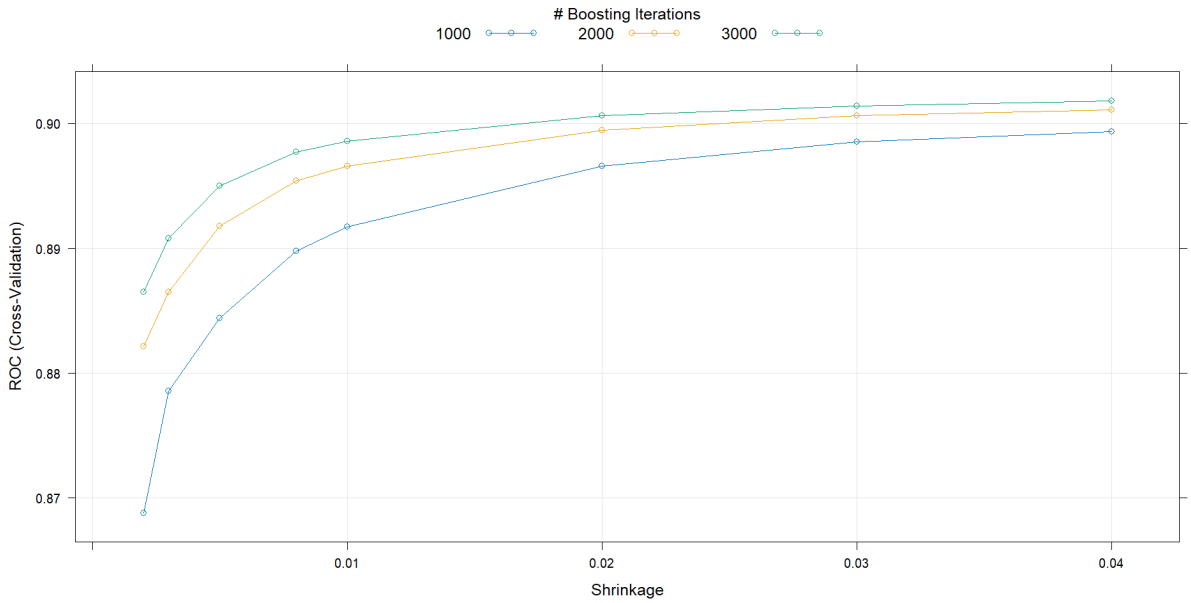

Figura 48: Gráfico Gradient Boosting. Fuente: elaboración propia.

Observamos como claramente el número de árboles que genera valores más altos de ROC son 3000 árboles, mientras que 1000 y 2000 se quedan ligeramente por debajo. El código ha determinado que los parámetros que maximizan la curva ROC y por tanto su área bajo la curva son: $n.trees = 3000$, $interaction.depth = 2$, $shrinkage = 0{,}04$ y $n.minobsinnode = 20$. Estos valores los guardaremos como configuración final del modelo. Posteriormente, se utilizarán para entrenar un modelo definitivo sobre todo el conjunto de datos, que será evaluado y comparado directamente con otros modelos avanzados que explicaremos más adelante. De este modo, se garantiza que la comparación entre algoritmos se realice sobre modelos que ya hemos previamente optimizado.

5.3. XgBoost

Una vez explicado Gradint Boosting y sus mejoras respecto al random forest, resulta oportuno incorporar a la comparación una de sus versiones más avanzadas y optimizadas de este: XGBoost (eXtreme Gradient Boosting). Este algoritmo fue propuesto por Chen y Guestrin (2016) y rápidamente se ha convertido en uno de los métodos de referencia en tareas de clasificación y regresión, principalmente por su alto rendimiento, sueficacia computacional y su gran flexibilidad a la hora de ajustar hiperparámetros.

XGBoost se basa en la misma idea general que el Gradient Boosting tradicional: construye árboles de forma secuencial, donde cada árbol intenta corregir los errores del anterior; sin embargo, introduce varias mejoras. Entre ellas, destacan el uso de una función de pérdida regularizada, la cual controla mejor el sobreajuste, y una implementación altamente optimizada que utiliza diferentes técnicas como el procesamiento paralelo, la poda eficiente y la gestión de valores perdidos de forma automática.

Una de las ventajas más destacadas de XGBoost es que ofrece un muy buen rendimiento predictivo incluso cuando se trabaja con conjuntos de datos pequeños. Esto se debe a que está diseñado para ser rápido y eficiente, aprovechando bien los recursos del ordenador y evitando procesos innecesarios. A diferencia de otros modelos, XGBoost permite trabajar con un amplio conjunto de parámetros ajustables, como la tasa de aprendizaje (eta), que controla cuán rápido aprende el modelo; el número de árboles a construir

(nrounds); la profundidad máxima de cada árbol (max_depth); el número mínimo de datos por nodo (min_child_weight); o el porcentaje de variables usado en cada árbol (colsample_bytree), entre otros.

A la hora de programar, para calibrarlo correctamente, se definió una rejilla de combinaciones posibles de estos parámetros, y se aplicó un proceso de validación cruzada para evaluar el rendimiento de cada configuración, utilizando el AUC como criterio principal. Una vez identificado el conjunto de parámetros óptimos, se entrenará un modelo final con esa configuración óptima. Este modelo se incluirá después en la comparación con los otros algoritmos evaluados (Gradient Boosting, CatBoost entre otros).

5.3.1. Sin información espacial

El gráfico resultate para el modelo sin información espacial es el siguiente:

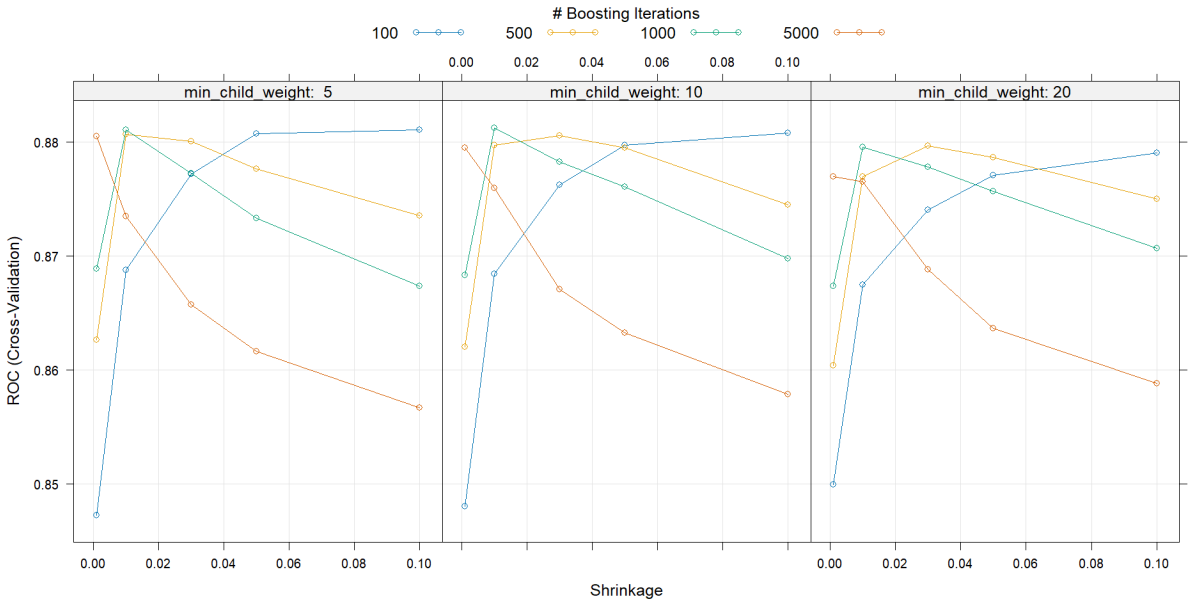

Figura 49: Gráfico XGBM. Fuente: elaboración propia.

En la grñáficase presenta el resultado del ajuste de hiperparámetros del modelo XGBoost mediante validación cruzada. Se analizan diferentes combinaciones de los parámetros *min_child_weight* y *shrinkage* (también conocido como *eta*). Cada panel corresponde a un valor distinto de *min_child_weight* (5, 10 y 20), mientras que el eje X representa el valor de *shrinkage* y el eje Y el AUC promedio obtenido.

Se puede observar que para valores bajos de *shrinkage* (especialmente menores a 0.05), se alcanzan los mejores valores de AUC, destacando el caso en el que *min_child_weight* es igual a 10. En general, al aumentar el valor de `shrinkage` se reduce el rendimiento del modelo, lo que podría deberse a un sobreajuste por aprendizaje demasiado agresivo.

El conjunto óptimo de hiperparámetros determinado es el siguiente: $nrounds = 1000$, $max\_depth = 6$, $eta = 0{,}01$, $gamma = 0$, $colsample\_bytree = 1$, $min\_child\_weight = 10$ y $subsample = 1$

Estos valores los guardaremos como configuración final del modelo para su posterior comparación.

5.3.2. Con información espacial

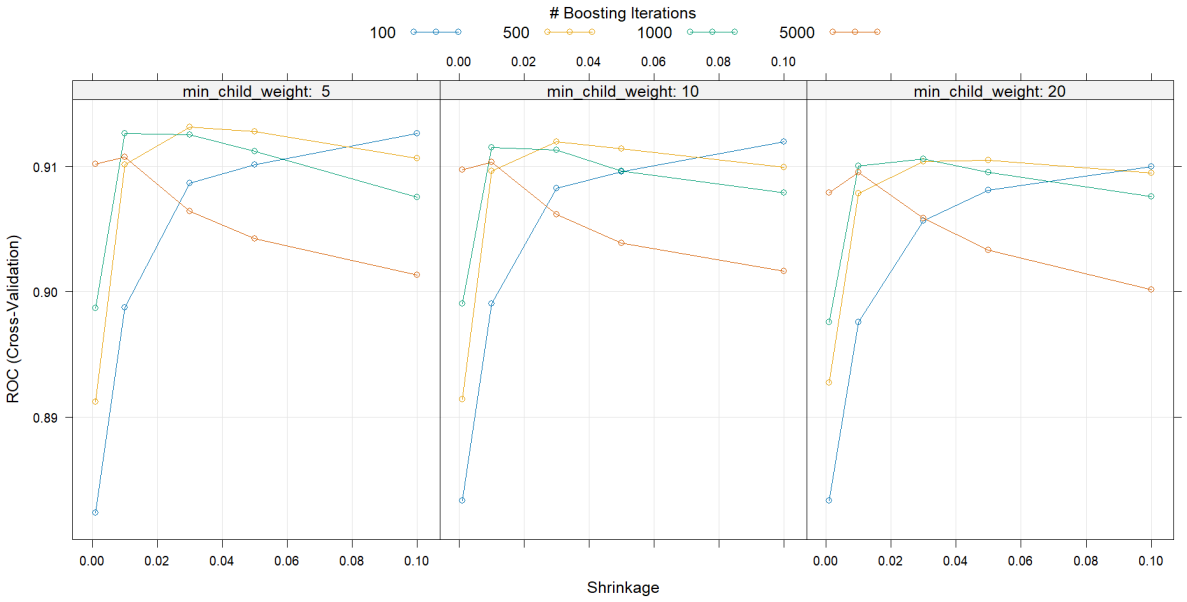

Figura 50: Gráfico XGBM. Fuente: elaboración propia.

Para el caso del modelo con información espacial, el conjunto óptimo de hiperparámetros es el siguiente: $nrounds = 500$, $max\_depth = 6$, $eta = 0{,}03$, $gamma = 0$, $colsample\_bytree = 1$, $min\_child\_weight = 5$ y $subsample = 1$

5.4. CatBoost

Otra de las versiones optimizadas derivada de Grading Boosting es el algoritmo CatBoost (Categorical Boosting). Este modelo, propuesto por el equipo de investigación de Yandex (Dorogush et al., 2018), pertenece a la familia de los algoritmos de Gradient Boosting, pero introduce mejoras específicas que lo diferencian de otros métodos como XGBoost o el Gradient Boosting clásico.

La principal característica de CatBoost es su capacidad para tratar automáticamente variables categóricas, sin necesidad de tener que codificarlas a mano mediante técnicas como por ejemplo el one-hot encoding. En situaciones reales, donde muchas de las variables no son numéricas, esto es una gran ventaja, ya que mantiene la información estructural de los datos reduciendo a su vez el riesgo de sobreajuste. Además, CatBoost emplea una técnica conocida como ordenamiento por permutaciones (ordered boosting), gracias a la cual se mitiga el problema del target leakage, es decir, el mal uso de información del futuro en el proceso de entrenamiento.

Desde el punto de vista práctico, CatBoost se ha mostrado especialmente eficaz en problemas de clasificación con conjuntos de datos con tamaños pequeños o medianos. El modelo permite ajustar diversos hiperparámetros como la tasa de aprendizaje (learning_rate), el número de iteraciones (iterations), la profundidad de los árboles (depth) y los términos de regularización (l2_leaf_reg).

En este trabajo, al igual que con los demás algoritmos, se llevó a cabo una búsqueda de hiperparámetros mediante validación cruzada, utilizando el AUC como métrica principal para seleccionar la mejor configuración. Posteriormente, el modelo final calibrado será

comparado con los de Random Forest, Gradient Boosting y XGBoost bajo los mismos criterios de evaluación.

5.4.1. Sin información espacial

Se aplicó el algoritmo CatBoost sobre el modelo óptimo para el conjunto sin información espacial y se obtuvo la siguiente gráfica:

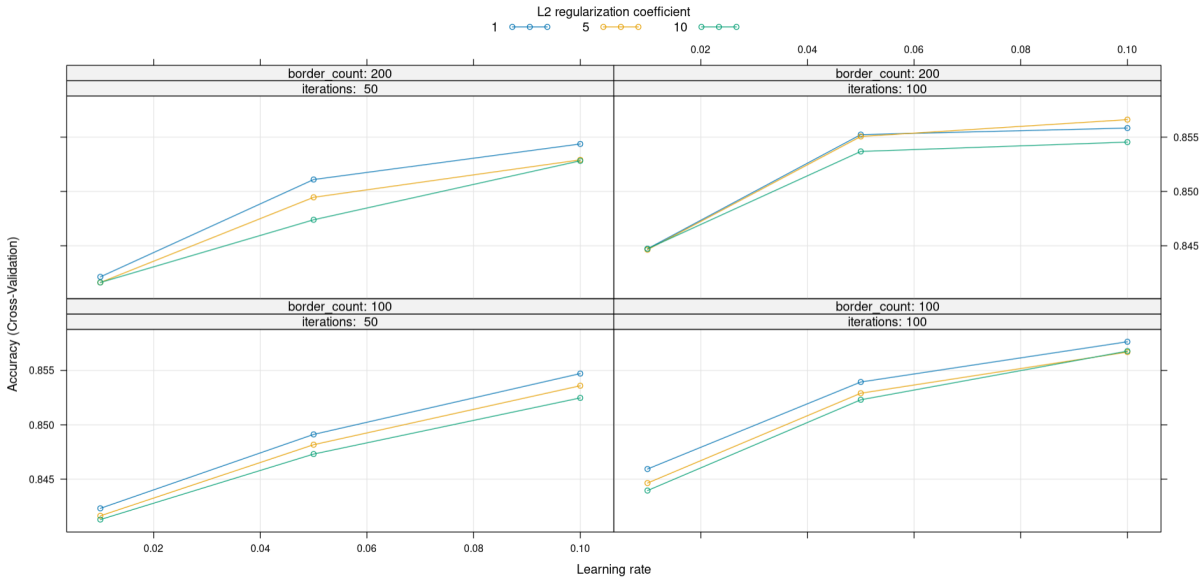

Figura 51: Gráfico Catboost. Fuente: elaboración propia.

El gráfico nos muestra el proceso de ajuste de hiperparámetros de CatBoost para el conjunto sin información espacial. Se observa mejora en precisión cuanto más aumenta la tasa de aprendizaje y también cuando se aplica una regularización L2 baja (l2_leaf_reg = 1). Se seleccionó como modelo óptimo el que combina un $learningrate = 0{,}1$, $iterations = 100$ y un $border|\_count = 100$, manteniendo $depth = 6$ y $rsm = 0{,}8$ constantes.

5.4.2. Con información espacial

Se aplicó el algoritmo CatBoost sobre el modelo óptimo para el conjunto con información espacial y se obtuvo la siguiente gráfica:

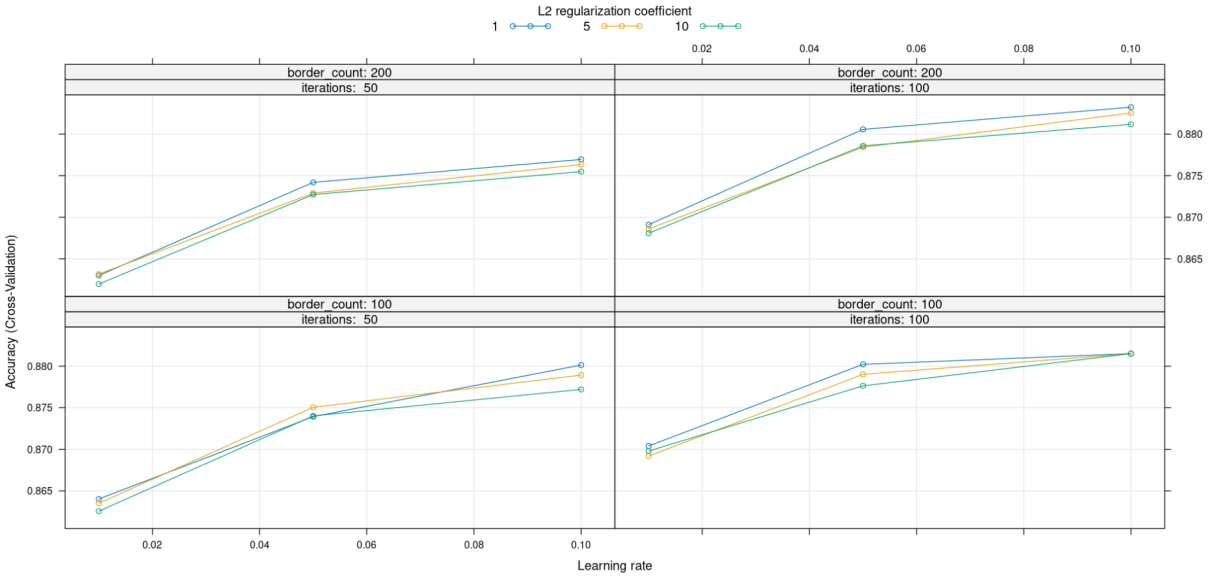

Figura 52: Gráfico Catboost. Fuente: elaboración propia.

En esta configuración de CatBoost, se exploraron también diferentes combinaciones de learning_rate, l2_leaf_reg, iterations y border_count, manteniendo fija la profundidad (depth = 6) y el subsample de columnas (rsm = 0.8). El rendimiento del modelo mejora cuanto mayor es la tasa y mejora también cuanto mayor es el número de iteraciones. El mejor modelo se obtuvo con $learning\_rate = 0{,}1$, $iterations = 100$ y $border\_count = 200$, una regularización L2 baja, $l2\_leaf\_reg = 1$.

5.5. Comparación de rendimiento entre modelos y elección del óptimo

A lo largo del trabajo hemos hecho selección de variables con diferentes procedimientos de regresión (AIC, BIC, K-general, RFE y MMPC), estos han sido comparados en cuanto a rendimiento en un Box-Plot y se ha escogido el mejor (el que ha dado mayor AUC) y el que tuviera más variables. A estos dos hemos añadido un tercero, creado con un Random Forest sobre todas las variables, para luego aplicar de nuevo el procedimiento Random Forest sobre los tres. Estos tres se han vuelto a comparar en un Box-Plot, para finalmente escoger aquel con mejores métricas o rendimiento y entrenarlo con diferentes técnicas de aprendizaje automático (Grading Boosting, Xgboost y Catboost).

Nuestro objetivo es utilizar el mejor modelo de todos ellos para predecir, con las variables de este modelo, la probabilidad de presencia de lobo en un hexágono. Para hacer esto, hemos creado un último Box-Plot en el que hemos incluido: el modelo con mejores resultados en la primera comparación (entre los métodos de regresión de elección de variables), aquel con mejores resultados en la segunda comparación (entre los Random Forest) y este último entrenado con las tres técnicas de aprendizaje automático ya mencionadas. En total, cinco modelos.

Para las técnicas de aprendizaje automático, se ha entrenado este último modelo seleccionando las métricas óptimas obtenidas en apartados anteriores. Finalmente hemos obtenido dos gráficos, uno parael modelo sin información espacial y otro con ella.

Modelo sin información espacial:

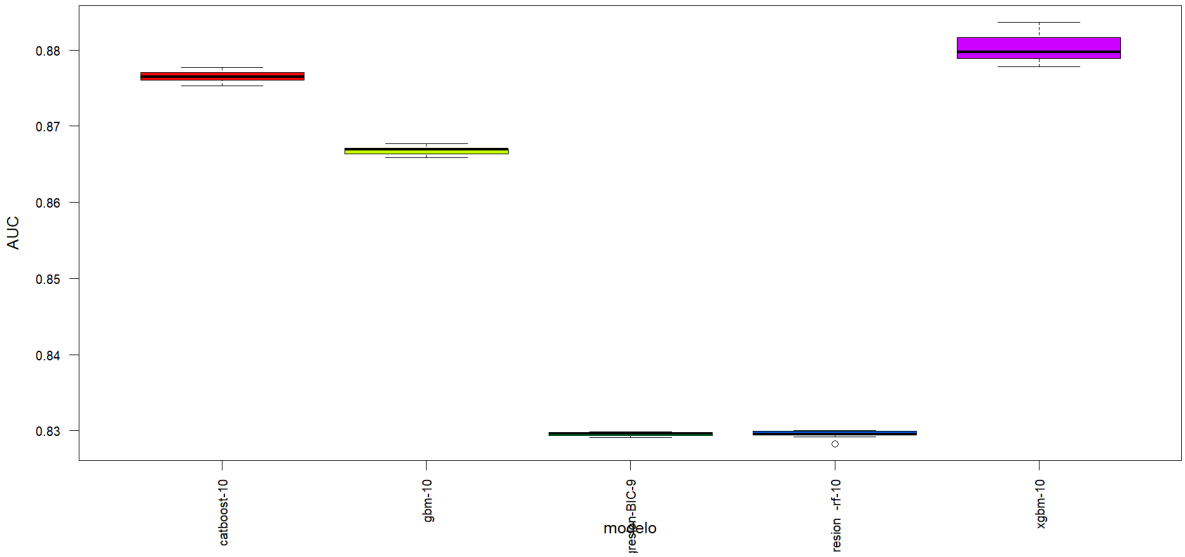

Figura 53: Gráfico comparativo final. Fuente: elaboración propia.

En este gráfico comparativo final se observan varias cuestiones. Primero, los dos modelos con peor rendimiento son aquellos sobre los que no se han aplicado técnicas de aprendizaje automático, lo cual respalda la mejora en rendimiento derivada de estas técnicas respecto a un modelo de regresión simple. Con un rendimiento medio se encuentra el modelo de gradient boosting básico, aunque con bastante más varianza que los anteriormente mencionados.

Por último, se observa un rendimiento muy bueno (con una AUC cercana a 0.88) tanto en el modelo CatBoost como en XGBoost, teniendo este último una caja mucho más ancha, es decir, una varianza considerablemente mayor.

Por todo ello, a la hora de decidir cuál es el mejor modelo, y dado que ambos presentan un rendimiento muy similar, se opta por CatBoost, premiando su menor varianza. Por tanto, el modelo elegido será el siguiente, entrenado con CatBoost:

```
c("bio1", "viento_mean", "bio12", "presion_humana", "porc_bosque",
"bio6", "bio4", "viento_relativo", ."elevacion", "densidad")
```

Modelo con información espacial

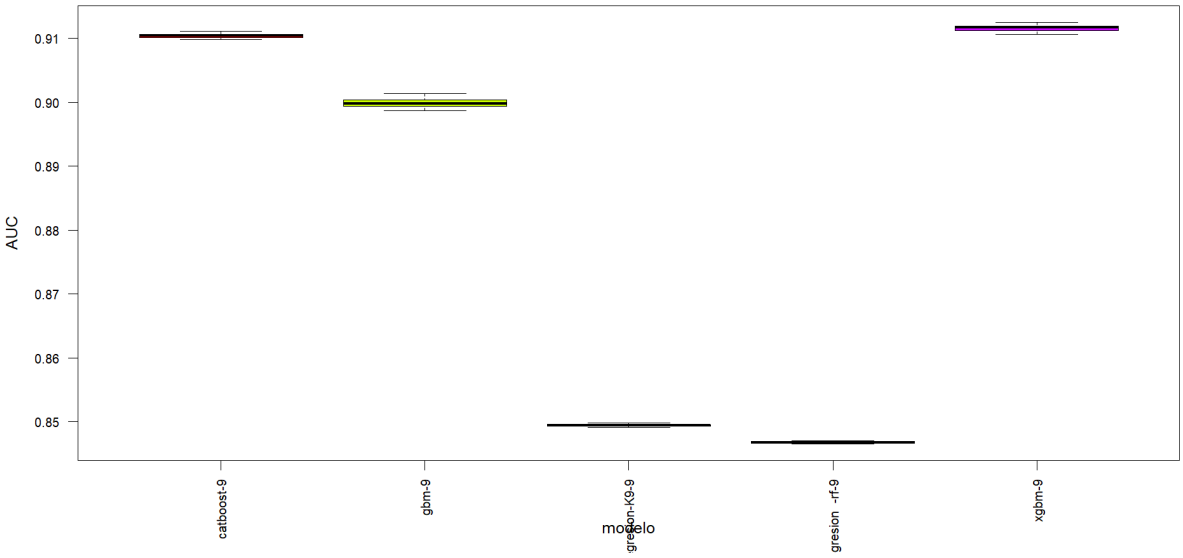

Figura 54: Gráfico comparativo final. Fuente: elaboración propia.

En este gráfico comparativo final se observan varias cosas. En primer lugar, los modelos con menor rendimiento vuelven a ser aquellos que no incorporan técnicas de aprendizaje automático, lo que refuerza la utilidad de estos métodoss. En un nivel intermedio de rendimiento se sitúa de nuevo el modelo de gradient boosting básico, aunque con más varianza que los modelos principales.

Finalmente, los mejores resultados los presentan tanto el modelo CatBoost como XGBoost, con valores de AUC claramente superiores al resto (cercana a 0.91). En este caso, la Accuracy medio más alto corresponde a XGBoost, que además mantiene una varianza baja.

Por tanto, dado que XGBoost no solo presenta el valor de AUC medio más alto, sino también una baja varianza, se decide elegir este modelo como el mejor en este caso. Por tanto, el modelo elegido será el siguiente, entrenado con XgBoost:

```
c("porc_vecinos_con_lobo", "bio1", "bio5", "bio12", "bio6",
"bio4", .elevacion", .accesibilidad", "viento_relativo")
```

5.5.1. Matriz de confusión, curva ROC, AUC, sensibilidad, especificidad

Para evaluar el rendimiento de los modelos de clasificación binaria, se han utilizado distintos indicadores complementarios. La matriz de confusión permite visualizar la distribución de predicciones correctas e incorrectas, separando los verdaderos positivos (VP), falsos positivos (FP), verdaderos negativos (VN) y falsos negativos (FN). A partir de esta matriz se pueden derivar medidas más interpretables como la sensibilidad, que indica la proporción de casos positivos correctamente identificados (VP/(VP+FN)), y la especificidad, que mide la proporción de negativos correctamente clasificados (VN/(VN+FP)).

Por otro lado, visualizaremos la curva ROC para cada uno de los modelos, anteriormente explicada en otros apartados, así como el AUC

Modelo sin información espacial

Cuadro 8: Matriz de confusión del modelo CatBoost

Predicción / Real	**No**	**Yes**
No	9380	1191
Yes	326	708

Cuadro 9: Métricas de evaluación del modelo CatBoost

Métrica	**Valor**	**Interpretación**
Accuracy	0.8693	Precisión general del modelo
Kappa	0.4153	Acuerdo con respecto al azar
Sensitivity	0.9664	Detección de la clase No
Specificity	0.3728	Detección de la clase Yes
Balanced Accuracy	0.6696	Media de Sensibilidad y Especificidad
Pos Pred Value	0.8873	Precisión al predecir No
Neg Pred Value	0.6847	Precisión al predecir Yes

El modelo CatBoost muestra una alta precisión general (86.93 %) y una muy buena sensibilidad (96.64 %), lo que significa que identifica correctamente la mayoría de los casos de la clase mayoritaria, en este caso las ausencias. Sin embargo, su especificidad es más baja (37.28 %), lo que indica que tiene más dificultad para detectar correctamente los casos positivos reales, en este caso las presencias.

Esto es coherente con el desequilibrio de clases en los datos (prevalencia del 83.64 % para la clase ausencia). En este contexto, la Accuracy puede sobrestimar el rendimiento real, y por eso se reporta también la Balanced Accuracy (66.96 %) como una medida más justa.

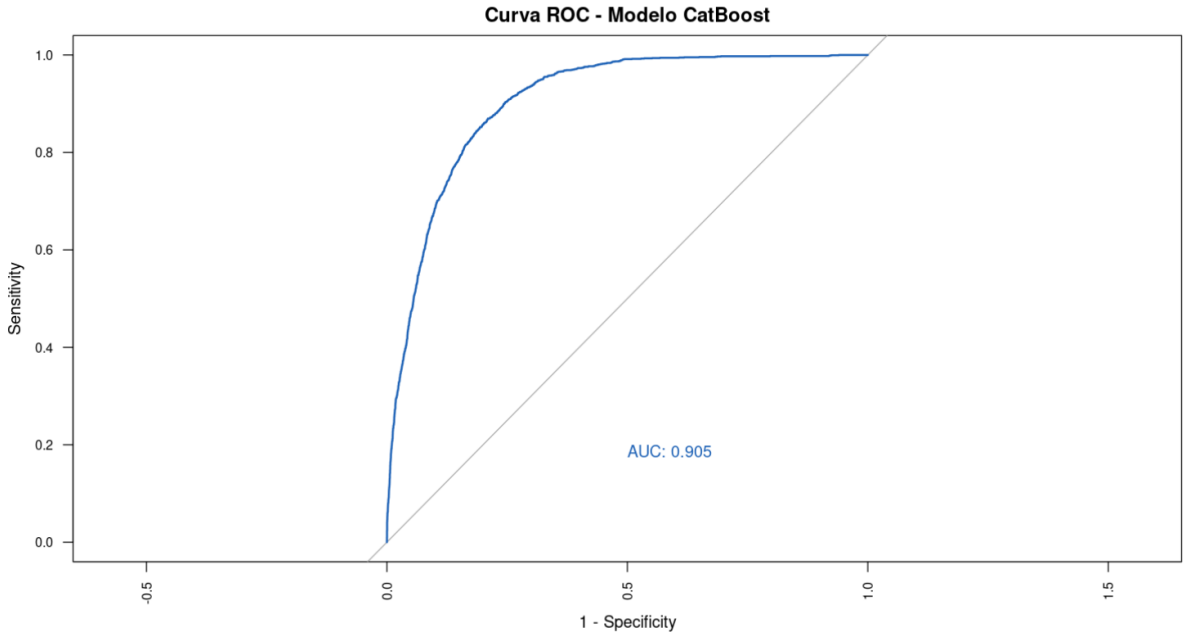

Figura 55: Gráfico Curva ROC. Fuente: elaboración propia.

Sobre la curva ROC y el AUC representados podemos decir que es bastante alto, de 0.905, muy cercano a 1. Un AUC de 0,905 implica que existe un 90,5 % de probabilidad de que el modelo asigne una puntuación mayor a un caso presencia que a uno de ausencia.

Modelo con información espacial

Cuadro 10: Matriz de confusión del modelo XGBoost

Predicción / Real	**No**	**Yes**
No	9451	541
Yes	255	1358

Cuadro 11: Métricas de evaluación del modelo XGBoost

Métrica	**Valor**	**Interpretación**
Accuracy	0.9314	Precisión global del modelo
Kappa	0.7333	Acuerdo con respecto al azar
Sensitivity	0.9737	Capacidad para detectar la clase No
Specificity	0.7151	Capacidad para detectar correctamente los Yes
Balanced Accuracy	0.8444	Media entre sensibilidad y especificidad
Pos Pred Value	0.9459	Precisión al predecir No
Neg Pred Value	0.8419	Precisión al predecir Yes

El modelo XGBoost entrenado sin información espacial presenta un accuracy del 93.14 %, considerablemente superior a la tasa base. La Kappa de 0.7333 indica un alto acuerdo entre predicciones y clases reales, más allá del azar.

El modelo logra una alta sensibilidad (97.37 %), lo que implica que predice correctamente la mayoría de las ausencias. Además, mejora bastante respecto al modelo anterior en especificidad (71.51 %), lo que indica una mayor capacidad de identificar correctamente las ausencias. Esto tiene mucho sentido, sabiendo que para este modelo estamos teniendo en cuenta la correlación espacial.

En conjunto, la balanced accuracy de 84.44 % demuestra un rendimiento sólido y equilibrado, a pesar del desequilibrio entre clases.

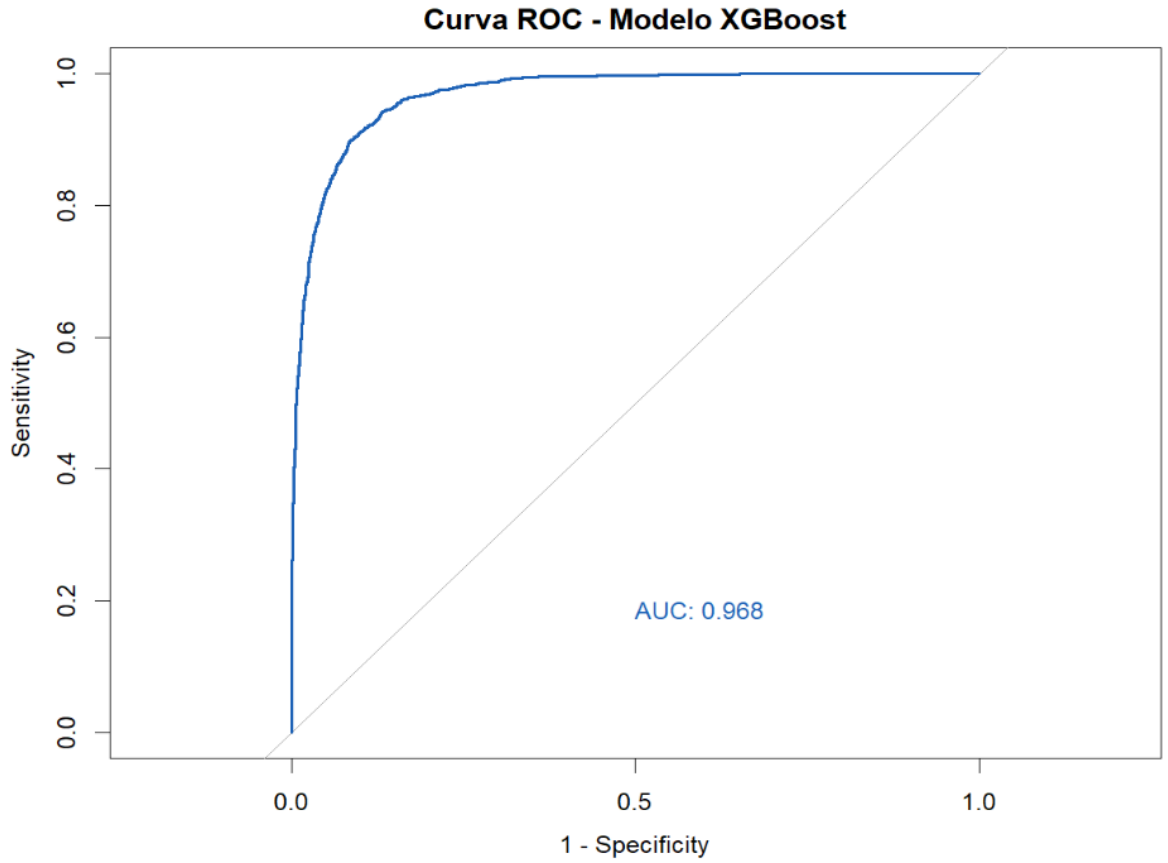

Figura 56: Gráfico Curva ROC. Fuente: elaboración propia.

Sobre la curva ROC y el AUC representados podemos decir que es bastante alto, de 0.968, ligeramente más alto que para el modelo sin información espacial. Un AUC de 0,968 implica que existe un 96,8 % de probabilidad de que el modelo asigne una puntuación mayor a un caso presencia, que a uno de ausencia.

6. Mapa de probabilidades

Una vez entrenado el modelo de clasificación final y evaluado su rendimiento mediante las métricas correspondientes (accuracy, curva ROC, AUC, etc.), el siguiente paso fue aplicar dicho modelo al conjunto de hexágonos espaciales previamente definidos en el estudio. Este proceso tiene como objetivo generar un mapa de predicción de probabilidades de presencia del lobo ibérico, basado exclusivamente en las variables explicativas disponibles para cada hexágono.

Para ello, se utilizó la grilla hexagonal completa como base espacial, en la cual cada celda contiene los valores de las variables predictoras seleccionadas. Sobre esta malla se aplicó el modelo entrenado, de forma que para cada hexágono se obtuvo una probabilidad estimada de presencia, es decir, el valor de salida del modelo para esa combinación concreta de variables.

Es importante destacar que este procedimiento no implica una nueva fase de entrenamiento, sino que consiste en una predicción sobre datos nuevos (out-of-sample), utilizando exclusivamente el modelo ya ajustado. La geometría espacial de los hexágonos se mantuvo

intacta, lo cual permitió asignar a cada celda una probabilidad y, en consecuencia, representar visualmente los resultados en forma de mapa continuo de distribución potencial.

El resultado de esta etapa es un mapa temático donde cada hexágono está coloreado en función de la probabilidad de presencia del lobo ibérico. Esta representación espacial facilita enormemente la interpretación de los resultados del modelo, ya que permite identificar áreas de mayor y menor probabilidad de presencia.

6.1. Mapas sin información de hexágonos adyacentes

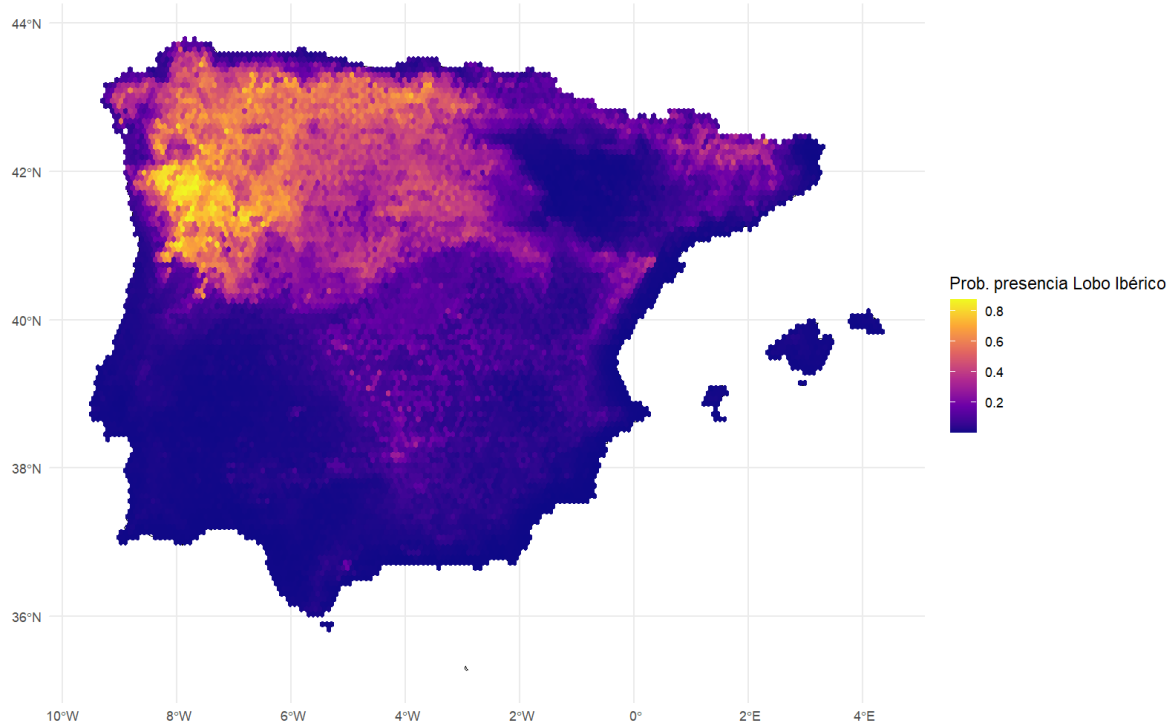

Figura 57: Mapa de probabilidades de presencia predichas por nuestro modelo definitivo sin información de hexágonos adyacentes. Fuente: elaboración propia.

En la figura se muestra el mapa de probabilidad de presencia del lobo ibérico en la península ibérica y Baleares, generado a partir de un modelo predictivo entrenado exclusivamente con variables ambientales, topográficas y antrópicas, pero sin considerar información espacial de hexágonos vecinos.

Cada hexágono representa una celda espacial sobre la cual se ha estimado la probabilidad de presencia de la especie, en función de las condiciones locales observadas. La escala de colores utilizada va del morado (probabilidades bajas, cercanas a 0.1) al amarillo (probabilidades altas, cercanas a 0.9).

Tal como se observa, las áreas con mayor probabilidad de presencia se concentran en el noroeste peninsular, especialmente en Galicia, Castilla y León y partes del norte de Portugal. Esta predicción se alinea con el área histórica y actual de distribución del lobo ibérico, lo que sugiere que el modelo, incluso sin variables espaciales explícitas, logra captar adecuadamente las zonas donde las condiciones ecológicas y humanas son más favorables para la especie.

En contraste, regiones como el sur de Andalucía y el este de Castilla-La Mancha presentan valores muy bajos de probabilidad, lo que refleja condiciones ambientales menos

adecuadas. También destaca la ausencia de probabilidad relevante en las Islas Baleares, lo cual es coherente con la realidad ecológica, ya que la especie no está presente en estos territorios insulares.

Cabe subrayar que, al excluir la información sobre hexágonos adyacentes, este mapa representa una predicción puramente basada en las variables internas de cada celda, sin incorporar patrones de dependencia espacial. Esto puede limitar la detección de núcleos de expansión o áreas potenciales de colonización conectadas, pero tiene el valor de reflejar las condiciones locales inherentes de idoneidad ambiental.

En definitiva, este mapa aporta una representación útil y coherente del hábitat potencial del lobo ibérico en la península, y constituye una base sobre la cual comparar el efecto que tendría la incorporación de variables espaciales, como se analiza en la siguiente sección.

6.2. Mapas con información de hexágonos adyacentes

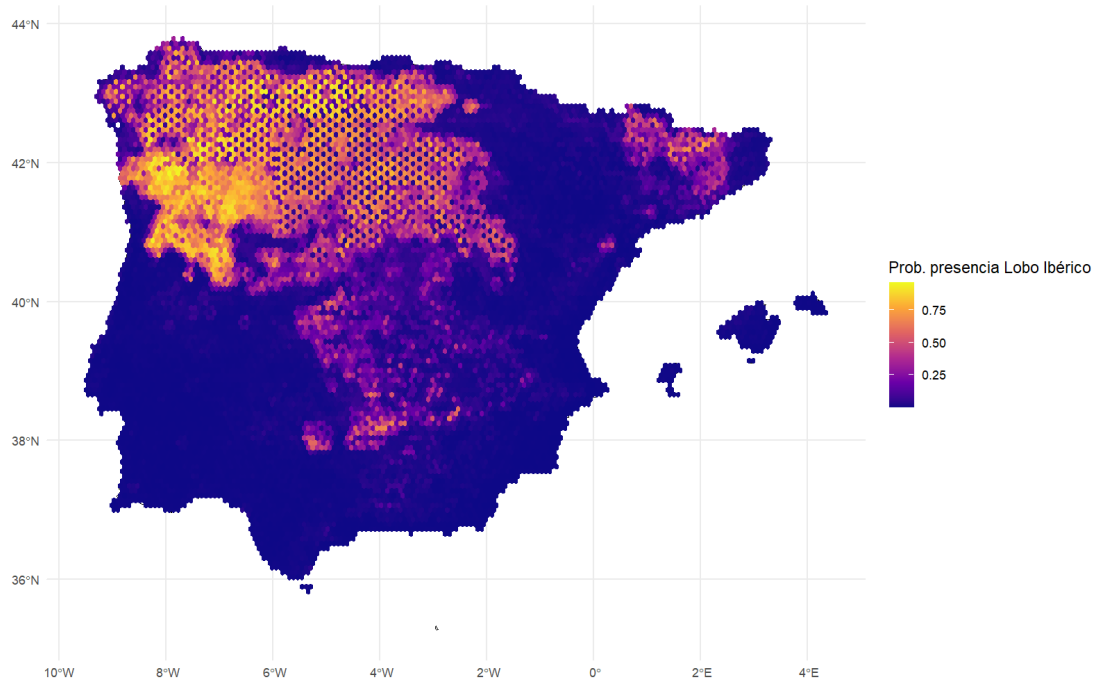

Figura 58: Mapa de probabilidades de presencia predichas por nuestro modelo definitivo con información de hexágonos adyacentes. Fuente: elaboración propia.

La figura muestra el mapa de probabilidades de presencia del lobo ibérico generado a partir del modelo entrenado con la inclusión de una variable espacial: la proporción de hexágonos adyacentes con presencia conocida de la especie.

Visualmente, el patrón general del mapa se mantiene en líneas similares al del modelo anterior (sin información espacial), con las mayores probabilidades concentradas en la zona norte y noroeste de la península. No obstante, se aprecian algunas diferencias importantes.

En primer lugar, la transición entre zonas de alta y baja probabilidad es ahora más abrupta, lo cual refleja la influencia de la presencia en celdas vecinas. Esta influencia se traduce en una mayor consistencia espacial de las áreas de alta probabilidad: se generan núcleos más definidos y continuos, particularmente en Castilla y León, Galicia y el norte de Portugal.

En segundo lugar, aparecen nuevas áreas con probabilidad intermedia en torno a zonas ya conocidas como favorables, lo que indica que el modelo reconoce patrones de dispersión o conectividad ecológica. Estas zonas intermedias probablemente no hubieran sido detectadas solo con las variables ambientales internas a cada celda.

Asimismo, se reduce ligeramente el número de hexágonos aislados con alta probabilidad fuera del área de distribución principal, ya que la ausencia de presencia en los hexágonos vecinos tiende a disminuir la probabilidad estimada en celdas aisladas. Esto puede considerarse una mejora desde el punto de vista ecológico, dado que refleja un patrón más realista de ocupación territorial.

7. Conclusiones

Esta sección recoge las principales conclusiones del trabajo. Se resumen los resultados más relevantes obtenidos mediante los distintos modelos de predicción, se contrastan con hallazgos en la literatura, y se discuten sus implicaciones desde un punto de vista ecológico y aplicado. Además, se valoran posibles utilidades futuras del modelo definitivo obtenido.

7.1. Interpretación de resultados

El análisis de los modelos predictivos desarrollados a lo largo del estudio ha permitido identificar diferencias claras en cuanto a su rendimiento. Los modelos que no incorporan técnicas de aprendizaje automático o machine learning (basados en regresión) presentan valores de AUC más bajos y mayor varianza en sus resultados, esto denota que tienen una capacidad predictiva más limitada y menos robusta.

En contraste, los modelos que sí incorporan algoritmos de aprendizaje automático, concretamente Gradient Boosting, CatBoost y XGBoost, muestran mejoras sustanciales tanto en AUC como en estabilidad. Entre ellos, XGBoost destaca cuando se incorpora información espacial (proporción de hexágonos vecinos con presencia del lobo). Este modelo alcanza un AUC de 0.968, el más alto entre todos los modelos evaluados, lo cual indica que tiene un 96.8 % de probabilidad de asignar un porcentaje más alto a un caso positivo (presencia) que a uno negativo (ausencia).

Además, el modelo XGBoost con información espacial presenta un excelente rendimiento en términos de Accuracy (93.14 %) y una Balanced Accuracy del 84.44 %, lo que indica un comportamiento equilibrado a pesar del desbalance de clases en los datos. También muestra valores altos tanto en sensibilidad (97.37 %) como en especificidad (71.51 %), lo que confirma su capacidad para detectar tanto las ausencias como las presencias de manera eficaz. Sin embargo, en el modelo sin información espacial, a pesar de mantener una sensibilidad alta (96.64 %), la especificidad baja drásticamente (37.28 %). Esta bajada es común en modelos con desbalance de clases, sin embargo el modelo con información espacial compensa esta pérdida con el conocimiento del estado de presencia/ausencia de sus hexágonos vecinos.

En definitiva, los resultados muestran que la inclusión de variables espaciales mejora significativamente el rendimiento del modelo. XGBoost, al combinar una alta precisión, un AUC sobresaliente y una baja varianza, se consolida como la opción óptima para modelizar la distribución potencial del lobo ibérico a partir de los datos disponibles.

7.2. Comparación con la literatura

Nos encontramos por tanto ante unos resultados que subrayan la gran importancia de la correlación espacial a la hora de obtener un modelo predictivo. Esta relevancia metodológica es consistente con lo planteado en trabajos como Dormann et al. (2007). Tal y como señalan, "*ignorar la autocorrelación espacial puede llevar a una subestimación de los errores estándar, inflar el R^2 y dar lugar a inferencias incorrectas sobre los efectos de las variables predictoras*".

En su revisión, estos autores exponen que la mayoría de los conjuntos de datos de presencia/ausencia de especies presentan cierto grado de dependencia espacial, la cual, si no se considera en nuestro modelo, puede llegar a comprometer su validez estadística, así como su capacidad predictiva. Proponen diferentes métodos para tenerla en cuenta, como incluir variables espaciales derivadas, modelos autorregresivos espaciales o el uso de técnicas de remuestreo adaptadas.

Siguiendo esta línea de pensamiento, en este trabajo se ha incorporado una variable espacial que representa la proporción de hexágonos vecinos con presencia del lobo, como una forma de tener en cuenta esta correlación espacial, y captar asi patrones. Esta estrategia ha demostrado ser eficaz, ya que el modelo que incluye dicha variable ha mejorado significativamente tanto predictivamente (AUC) como en coherencia espacial, en línea con lo que indica la literatura.

Sin embargo, existen muchos escenarios donde la proporción de presencia en hexágonos adyacentes no se conoce, o incluso donde se tiene información nula acerca de los vecinos; en cuyos casos sería útil tener a nuestra disposición un modelo entrenado sin esta información espacial.

En cuanto a los resultados de distribución finales (mostrados en los mapas de probabilidad), diversos estudios (Blanco et al., 2005 o Torres et al., 2015) han identificado ya el noroeste peninsular (concretamente Galicia, Castilla y León y el norte de Portugal) como las áreas de mayor presencia de la especie, lo cual coincide con las zonas marcadas con colores más claros en nuestros mapas de predicción. Asimismo, la falta de presencias en áreas como el sur de Andalucía o las Islas Baleares también ha sido previamente documentada y se ve confirmada en nuestro trabajo.

7.3. Aplicabilidad del modelo e implicaciones ecológicas y de conservación

Los resultados obtenidos en este trabajo tienen importantes implicaciones desde el punto de vista ecológico y para la gestión y conservación del lobo ibérico. En primer lugar, los mapas de probabilidad de presencia generados permiten identificar aquellas zonas con condiciones ambientales y humanas favorables para esta especie. Esta información podría ser bastante útil a la hora de orientar acciones de conservación, vigilancia o incluso restauración del hábitat. En particular, podría contribuir a estrategias de reintroducción del lobo ibérico en zonas donde estuvo presente pero actualmente ya no presenta registros, aumentando así las posibilidades de éxito mediante una correcta selección de ubicaciones óptimas.

De esta forma y gracias a su alta precisión, el modelo con información espacial permite identificar con claridad las zonas donde el lobo ya está presente de forma estable, así como otras áreas con potencial para su expansión. Este tipo de predicción es útil para analizar rutas de dispersión y evitar así conflictos con actividades humanas en zonas que podrían volver a ser ocupadas por la especie.

En conjunto, este enfoque basado en modelos predictivos no solo aporta conocimiento científico sobre los factores principales que determinan la presencia del lobo, sino que además también ofrece una herramienta práctica para la toma de decisiones en cuanto a conservación, planificación territorial o gestión de fauna silvestre.

Este trabajo ha demostrado la utilidad de los modelos de aprendizaje automático, y en particular de XGBoost con variables espaciales, para predecir la distribución del lobo ibérico en la Península Ibérica. La integración de variables ambientales, antrópicas y espaciales ha permitido generar predicciones precisas y coherentes con el conocimiento ecológico actual de la especie. Aunque este trabajo ya incorpora variables espaciales como parte del modelado de distribución, conviene señalar que existen enfoques estadísticos que permiten tratar explícitamente la autocorrelación espacial. Modelos geoestadísticos como el kriging (Cressie, 1993) o los modelos gaussianos espaciales (Rue, Martino, & Chopin, 2009) incorporan estructuras de covarianza que hacen que las predicciones dependan también de la distancia entre puntos. Por su parte, modelos jerárquicos bayesianos espaciales permiten incluir efectos aleatorios que varían suavemente en el espacio, proporcionando una representación más realista de los patrones espaciales subyacentes.

Asimismo, para evaluar de forma más rigurosa el rendimiento de los modelos en contextos espaciales, sería recomendable aplicar técnicas de validación cruzada espacial. A diferencia de la validación aleatoria, este enfoque divide los datos teniendo en cuenta la proximidad geográfica entre observaciones, lo que permite evitar el sesgo de sobreestimación del rendimiento. Métodos como el particionado por bloques, buffering o agrupamiento espacial ofrecen una estimación más realista de la capacidad predictiva en nuevas regiones (Roberts et al., 2017).

Más allá del caso concreto analizado, la metodología empleada ofrece un marco replicable para otras especies y contextos territoriales, con aplicaciones potenciales en conservación, planificación ambiental y gestión de fauna silvestre.

Este trabajo pretende contribuir, desde una perspectiva aplicada y basada en datos, a una mejor comprensión y gestión de una de las especies más emblemáticas del patrimonio natural ibérico.

8. Bibliografía

Referencias

[1] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control, 19*(6), 716–723. https://doi.org/10.1109/TAC.1974.1100705

[2] Ausband, D. E., Mitchell, M. S., Stenglein, J. L., & Waits, L. P. (2020). Environmental and social factors influencing wolf (Canis lupus) howling behavior. *Journal of Mammalogy*, 101(1), 1–10. https://doi.org/10.1093/jmammal/gyz212

[3] Birch, C. P. D., Oom, S. P., & Beecham, J. A. (2007). Rectangular and hexagonal grids used for observation, experiment and simulation in ecology. *Ecological Modelling*, *206*(3–4), 347–359. https://doi.org/10.1016/j.ecolmodel.2007.03.041

[4] Blanco, J. C., Reig, S., & Cuesta, L. (2005). Distribution, status and conservation problems of the wolf *Canis lupus* in Spain. In L. Boitani & P. C. Paquet (Eds.), *Wolf conservation and recovery* (pp. 65–80). Cambridge University Press.

[5] Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

[6] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group.

[7] Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

[8] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

[9] Chamberlain, S. A., Barve, V., Mcglinn, D., Oldoni, D., Desmet, P., Geffert, L., & Ram, K. (2022). *rgbif: Interface to the Global Biodiversity Information Facility API*. R package version 3.x. https://CRAN.R-project.org/package=rgbif

[10] European Environment Agency (2020). CORINE Land Cover (CLC) 2018 raster (Version 2020, 100m resolution). Copernicus Land Monitoring Service. https://land.copernicus.eu/pan-european/corine-land-cover/clc2018

[11] Cressie, N. (1993). *Statistics for Spatial Data*. Revised edition. Wiley.

[12] Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., & Lawler, J. J. (2007). Random forests for classification in ecology. *Ecology*, *88*(11), 2783–2792. https://doi.org/10.1890/07-0539.1

[13] Dormann, C. F., McPherson, J. M., Araújo, M. B., Bivand, R., Bolliger, J., Carl, G., ... & Wilson, R. (2007). Methods to account for spatial autocorrelation in the analysis of species distributional data: a review. *Ecography*, *30*(5), 609–628. https://doi.org/10.1111/j.2007.0906-7590.05171.x

[14] Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*. https://arxiv.org/abs/1810.11363

[15] Elith, J., & Leathwick, J. R. (2009). Species distribution models: Ecological explanation and prediction across space and time. *Annual Review of Ecology, Evolution, and Systematics*, *40*, 677–697. https://doi.org/10.1146/annurev.ecolsys.110308.120159

[16] Elith, J., Kearney, M., & Phillips, S. (2010). The art of modelling range-shifting species. *Methods in Ecology and Evolution*, *1*(4), 330–342. https://doi.org/10.1111/j.2041-210X.2010.00036.x

[17] Elton, C. S. (1927). *Animal Ecology*. London: Macmillan.

[18] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

[19] Fick, S. E., & Hijmans, R. J. (2017). WorldClim 2: New 1-km spatial resolution climate surfaces for global land areas. *International Journal of Climatology*, **37**(12), 4302–4315. https://www.worldclim.org/data/worldclim21.html

[20] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

[21] Martínez-Fernández, J., et al. (2022). Capítulo 18: Modelos de distribución de especies en ecosistemas forestales. En *Libro Geoforest*. Universidad de Valladolid. https://mastergeoforest.es/wp-content/uploads/2022/10/Libro-geoforest-Cap.-18-Modelos-de-distribucion-de-especies_Final.pdf

[22] Hijmans, R. J. (2023). geosphere: Spherical Trigonometry. R package version 1.5-18. https://CRAN.R-project.org/package=geosphere

[23] Grinnell, J. (1917). The niche-relationships of the California Thrasher. *The Auk*, *34*(4), 427–433. https://doi.org/10.2307/4072271

[24] Guisan, A., & Zimmermann, N. E. (2000). Predictive habitat distribution models in ecology. *Ecological Modelling*, *135*(2–3), 147–186. https://doi.org/10.1016/S0304-3800(00)00354-9

[25] Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning, 46*(1–3), 389–422. https://doi.org/10.1023/A:1012487302797

[26] Hales, T. C. (2001). The honeycomb conjecture. *Discrete & Computational Geometry*, *25*(1), 1–22. https://doi.org/10.1007/s004540010071

[27] Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*(1), 29–36. https://doi.org/10.1148/radiology.143.1.7063747

[28] Hutchinson, G. E. (1957). Concluding remarks. *Cold Spring Harbor Symposia on Quantitative Biology*, *22*, 415–427. https://doi.org/10.1101/SQB.1957.022.01.039

[29] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.

[30] Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. Springer. https://doi.org/10.1007/978-1-4614-6849-3

[31] Kuhn, M., & Wickham, H. (2022). *caret: Classification and Regression Training* (R package version 6.0-94). https://CRAN.R-project.org/package=caret

[32] Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection: A Practical Approach for Predictive Models*. CRC Press.

[33] Lobo, J. M. (2000). El papel de las condiciones ambientales en la distribución de la biodiversidad. *M3M: Monografías Tercer Milenio*, *1*, 55–64.

[34] Lobo, J. M., Jiménez-Valverde, A., & Real, R. (2008). AUC: a misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, *17*(2), 145–151. https://doi.org/10.1111/j.1466-8238.2007.00358.x

[35] National Geographic España. (2023). Lobo ibérico. *National Geographic*. Recuperado el 27 de abril de 2025, de https://www.nationalgeographic.es/animales/lobo-iberico

[36] Natural Earth (2023). Free vector and raster map data. https://www.naturalearthdata.com

[37] Portela García-Miguel, J. (2025). *Apuntes de clase de Predicción Avanzada*. Apuntes no publicados, Universidad Complutense de Madrid.

[38] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[39] South, A. (2017). rnaturalearth: World Map Data from Natural Earth. R package version 0.1.0. https://github.com/ropensci/rnaturalearth

[40] Chamberlain, S. A., Barve, V., Mcglinn, D., Oldoni, D., Desmet, P., Geffert, L., & Ram, K. (2022). *rgbif: Interface to the Global Biodiversity Information Facility API*. R package version 3.x. https://CRAN.R-project.org/package=rgbif

[41] Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., et al. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography*, *40*(8), 913–929. https://doi.org/10.1111/ecog.02881

[42] Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *71*(2), 319–392. https://doi.org/10.1111/j.1467-9868.2008.00700.x

[43] Said, S., Gaillard, J. M., Widmer, O., Débias, F., Bourgoin, G., Delorme, D., & Pellerin, M. (2016). What shapes European wolf distribution? Ecological constraints and anthropogenic pressures. *Biological Conservation*, 203, 252–259. https://doi.org/10.1016/j.biocon.2016.09.001

[44] Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6*(2), 461–464.

[45] Soberón, J., & Peterson, A. T. (2005). Interpretation of models of fundamental ecological niches and species' distributional areas. *Biodiversity Informatics*, *2*, 1–10.

[46] Torres, R. T., Silva, E., Brotas, G., & Fonseca, C. (2015). Influence of anthropogenic landscape features on wolf occurrence in southern Portugal. *Animal Conservation*, *18*(4), 285–294. https://doi.org/10.1111/acv.12174

[47] Tsamardinos, I., Brown, L. E., & Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning, 65*(1), 31–78. https://doi.org/10.1007/s10994-006-6889-7

[48] Vilà, C. (1993). *Aspectos morfológicos y ecológicos del lobo ibérico (Canis lupus Linnaeus, 1758)*. Tesis doctoral. Universidad de Barcelona.

[49] Venables, W. N., & Ripley, B. D. (2002). *Modern Applied Statistics with S* (4th ed.). Springer.

[50] Fick, S. E., & Hijmans, R. J. (2017). WorldClim 2: New 1-km spatial resolution climate surfaces for global land areas. *International Journal of Climatology*, **37**(12), 4302–4315. `https://www.worldclim.org/data/worldclim21.html`

[51] WorldPop. (2020). Global High Resolution Population Denominators Project. University of Southampton. `https://www.worldpop.org/`

Anexo I: Código en R

A continuación se presenta el código utilizado para la obtención de datos, construcción de la malla hexagonal, preprocesamiento y modelado de la distribución del lobo ibérico.

Carga de geometría y definición del área de estudio

```
# Cargar la geometría de España y Portugal
library(rnaturalearth)
library(sf)
iberia <- ne_countries(scale = "large",
                       country = c("Spain", "Portugal"),
                       returnclass = "sf") %>%
  st_union() %>%
  st_geometry() %>%
  st_transform(4326)
```

Listing 1: Cargar geometría de España y Portugal

Generación de malla hexagonal y delimitación de la Península Ibérica

```
# Cargar la geometría de España y Portugal (incluyendo islas)
iberia <- ne_countries(scale = "large",
                       country = c("Spain", "Portugal"),
                       returnclass = "sf") %>%
  st_union() %>%
  st_geometry() %>%
  st_transform(4326)

# Definir Bounding Box para la Península (Excluye Canarias)
peninsula_bbox <- st_bbox(c(xmin = -10, xmax = 5, ymin = 35, ymax =
    45), crs = 4326)
peninsula <- st_crop(iberia, peninsula_bbox)
```

Listing 2: Cargar geometría de España y Portugal y definir la Península

```r
# Generar la cuadrícula hexagonal dentro de la Península Ibérica
hexgrid <- st_make_grid(peninsula,
                        cellsize = 0.08,
                        what = 'polygons',
                        square = FALSE) %>%
  st_as_sf()
```

Listing 3: Construcción de la malla hexagonal

```r
# Filtrar hexágonos que están dentro de la Península Ibérica
hexgrid_peninsula <- hexgrid[st_intersects(hexgrid, peninsula,
   sparse = FALSE), ]

# Asignar ID único a cada hexágono
hexgrid_peninsula <- hexgrid_peninsula %>%
  mutate(hex_id = 1:n()) %>%
  select(hex_id, x)
```

Listing 4: Filtrar hexágonos dentro de la Península y asignar ID

```r
# Graficar la Península Ibérica con la malla hexagonal
ggplot() +
  geom_sf(data = peninsula, color = "black", fill = NA) +
  geom_sf(data = hexgrid_peninsula, fill = NA, color = "#698B22") +
  coord_sf() +
  theme_minimal()
```

Listing 5: Graficar la Península Ibérica con la malla hexagonal

Carga y depuración de datos de presencia del lobo

```r
# España
canis_lupus_data <- occ_data(
  scientificName = "Canis lupus",
  hasCoordinate = T,
  country = 'ES',
  limit = 10000
)

# Portugal
canis_lupus_pt <- occ_data(
  scientificName = "Canis lupus",
  hasCoordinate = TRUE,
  country = 'PT',
  limit = 10000
)
```

Listing 6: Descarga de datos de Canis lupus desde GBIF

```r
# Cargar desde disco (si ya se guardaron)
canis_lupus_data <- load(file="lobo_es.RData")
```

```r
canis_lupus_pt <- load(file="lobo_pt.RData")

# Unir y eliminar registros de perros
canis_lupus_iberia <- bind_rows(canis_lupus_data$data, canis_lupus_
   pt$data)
canis_lupus_iberia <- canis_lupus_iberia[!grepl("familiaris", canis
   _lupus_iberia$scientificName), ]

# Guardar limpio
save(canis_lupus_iberia, file = "lobos.RData")
```

Listing 7: Unión de datos y eliminación de registros erróneos

```r
# Convertir a objeto espacial con EPSG:4326
lobos_sf <- st_as_sf(canis_lupus_iberia,
                     coords = c("decimalLongitude", "
                        decimalLatitude"),
                     crs = 4326)

# Seleccionar columnas útiles y crear ID
lobos_sf <- lobos_sf %>%
  select(key, geometry) %>%
  mutate(id_lobo = 1:n()) %>%
  select(id_lobo, geometry)
```

Listing 8: Conversión a objeto espacial y selección de columnas

Extracción de variables climáticas y ambientales

```r
elev_raster <- rast("ruta/a/elev.tif")
peninsula_bbox <- ext(-10, 5, 35, 44)
elev_peninsula <- crop(elev_raster, peninsula_bbox)
plot(elev_peninsula, main = "Elevación - Península Ibérica")
hexgrid_con_lobos <- st_as_sf(hexgrid_con_lobos, wkt = "x")
elev_media <- terra::extract(elev_peninsula, hexgrid_vect, fun =
   mean, na.rm = TRUE)
hexgrid_con_lobos$elevacion <- elev_media[, 2]
```

Listing 9: Elevación - recorte y extracción

```r
bio1 <- rast("ruta/a/wc2.1_2.5m_bio_1.tif")
plot(bio1, main="Temperatura Media Anual (BIO1)")
bio1_media <- terra::extract(bio1, hexgrid_con_lobos, fun = mean,
   na.rm = TRUE)
hexgrid_con_lobos$bio1 <- bio1_media[, 2]
```

Listing 10: Temperatura media anual (BIO1)

```r
bio12_raster <- rast("ruta/a/wc2.1_2.5m_bio_12.tif")
plot(bio12_raster, main="Precipitación Anual (BIO12)")
bio12_media <- terra::extract(bio12_raster, hexgrid_con_lobos, fun
   = mean, na.rm = TRUE)
```

```
hexgrid_con_lobos$bio12 <- bio12_media[, 2]
```

Listing 11: Precipitación anual (BIO12)

```
bio5_raster <- rast("ruta/a/wc2.1_2.5m_bio_5.tif")
bio5_media <- terra::extract(bio5_raster, hexgrid_con_lobos, fun =
   mean, na.rm = TRUE)
hexgrid_con_lobos$bio5 <- bio5_media[, 2]
```

Listing 12: Temperatura máxima del mes más cálido (BIO5)

```
bio6_raster <- rast("ruta/a/wc2.1_2.5m_bio_6.tif")
bio6_media <- terra::extract(bio6_raster, hexgrid_con_lobos, fun =
   mean, na.rm = TRUE)
hexgrid_con_lobos$bio6 <- bio6_media[, 2]
```

Listing 13: Temperatura mínima del mes más frío (BIO6)

```
hexgrid_con_lobos$bio7 <- hexgrid_con_lobos$bio5 - hexgrid_con_
   lobos$bio6
```

Listing 14: Rango térmico anual (BIO7)

```
bio4 <- rast("ruta/a/wc2.1_2.5m_bio_4.tif")
bio4_media <- terra::extract(bio4, hexgrid_con_lobos, fun = mean,
   na.rm = TRUE)
hexgrid_con_lobos$bio4 <- bio4_media[, 2]
```

Listing 15: Estacionalidad de la temperatura (BIO4)

```
carpeta_viento <- "ruta/a/aire"
hex_vect <- vect(hexgrid_con_lobos)
ext_peninsula <- ext(-10, 4.5, 35.5, 44.5)
r_viento <- rast(list.files(carpeta_viento, pattern = "\\.tif$",
   full.names = TRUE))
r_viento_crop <- crop(r_viento, ext_peninsula)
r_viento_sd <- app(r_viento_crop, fun = sd, na.rm = TRUE)
r_viento_mean <- app(r_viento_crop, fun = mean, na.rm = TRUE)
hexgrid_con_lobos$viento_sd <- extract(r_viento_sd, hex_vect, fun =
    mean, na.rm = TRUE)[,2]
hexgrid_con_lobos$viento_mean <- extract(r_viento_mean, hex_vect,
   fun = mean, na.rm = TRUE)[,2]
```

Listing 16: Procesamiento de viento (velocidad media y desviación)

Cálculo del porcentaje de bosque por hexágono

```
library(terra)
library(sf)
library(dplyr)

# Cargar el raster CORINE (uso del suelo)
```

```
r <- rast("U2018_CLC2018_V2020_20u1.tif")

# Reproyectar el raster al CRS del estudio
r_wgs84 <- project(r, "EPSG:4326")

# Recorte del raster a la extensión de la Península Ibérica
ext_peninsula <- ext(-10.0, 4.5, 35.0, 44.5)
r_crop <- crop(r_wgs84, ext_peninsula)

# Clasificar píxeles de bosque: clases 23, 24, 25
r_bosques <- classify(r_crop,
                      rcl = matrix(c(23, 1,
                                     24, 1,
                                     25, 1),
                                   ncol = 2, byrow = TRUE),
                      others = NA)

# Convertir hexágonos a SpatVector para compatibilidad
hex_vect <- vect(hexgrid_con_lobos)

# Calcular la proporción de píxeles boscosos por hexágono
porc_bosque <- extract(r_bosques, hex_vect,
                       fun = function(x) {
                         sum(!is.na(x), na.rm = TRUE) / length(x)
                       })

# Añadir la variable al shapefile
hexgrid_con_lobos$porc_bosque <- porc_bosque[, 2]
```

Listing 17: Cálculo del porcentaje de bosque por hexágono a partir de CORINE

Código para el cálculo de variables poblacionales y de accesibilidad

```
# Cargar librerías necesarias
library(terra)
library(sf)
library(exactextractr)

# Cargar y unir rasters de población (WorldPop 2020)
esp_pop <- rast("esp_ppp_2020_constrained.tif")
prt_pop <- rast("prt_ppp_2020_constrained.tif")
pop_raster <- mosaic(esp_pop, prt_pop)

# Sumar la población por hexágono
hexgrid_con_lobos$poblacion <- exact_extract(pop_raster, hexgrid_
    con_lobos, 'sum')

# Calcular densidad (hab/km2)
hexgrid_con_lobos$area_km2 <- as.numeric(st_area(hexgrid_con_lobos)
    ) / 10^6
```

```
hexgrid_con_lobos$densidad <- hexgrid_con_lobos$poblacion / hexgrid
    _con_lobos$area_km2

# Identificar pueblos como hexágonos con > 500 hab.
pueblos_sf <- hexgrid_con_lobos %>%
  filter(poblacion > 500) %>%
  st_centroid()

# Calcular centroides de todos los hexágonos
hex_centroids <- st_centroid(hexgrid_con_lobos)

# Matriz de distancias entre cada hexágono y los pueblos
dist_matrix <- st_distance(hex_centroids, pueblos_sf)

# Calcular accesibilidad: suma de 1/(distancia + 1)
hexgrid_con_lobos$accesibilidad <- apply(dist_matrix, 1, function(x
    ) {
  sum(1 / (as.numeric(x) + 1))
}) * 100  # Escalado para visualización
```

Listing 18: Extracción de variables poblacionales y de accesibilidad

Código para el cálculo del porcentaje de vecinos con presencia de lobo

```
# Cálculo del porcentaje de vecinos con presencia de lobo

library(dplyr)
library(geosphere)

# Se parte del objeto 'hexgrid_con_lobos', que contiene una columna
    'centroide' y la variable binaria 'presencia'

# Extraer coordenadas de los centroides
coords <- st_coordinates(hexgrid_con_lobos$centroide)

# Crear matriz de distancias geodésicas (utilizando la fórmula de
    Haversine)
dist_matrix <- distm(coords, fun = distHaversine)

# Para cada hexágono, identificar los 6 vecinos más cercanos (
    excluyendo a sí mismo)
vecinos_list <- apply(dist_matrix, 1, function(x) {
  order(x)[2:7]  # El primero es el propio punto, se excluye
})

# Calcular el porcentaje de vecinos con presencia de lobo
porc_vecinos_con_lobo <- sapply(1:nrow(hexgrid_con_lobos), function
    (i) {
```

```
  vecinos <- vecinos_list[, i]
  mean(hexgrid_con_lobos$presencia[vecinos], na.rm = TRUE) * 100
})

# Añadir la nueva variable al dataset
hexgrid_con_lobos$porc_vecinos_con_lobo <- porc_vecinos_con_lobo
```

Listing 19: Cálculo del porcentaje de vecinos con presencia de lobo

Código para el cálculo de la presión humana

```
# La presión humana se calcula como el producto entre la
    accesibilidad
# (escalada) y el logaritmo de la densidad poblacional (también
    escalada)

lobos$presion_humana <- scale(log(lobos$poblacion + 1)) *
                        scale(lobos$accesibilidad)
```

Listing 20: Cálculo de la presión humana combinando población y accesibilidad

Código para el cálculo del índice de hábitat favorable

```
# El índice de hábitat favorable combina cobertura boscosa (
    positiva),
# densidad humana y accesibilidad (negativas). Todas las variables
    están escaladas.

lobos$indice_habitat <- scale(lobos$porc_bosque) -
                        scale(lobos$densidad) -
                        scale(lobos$accesibilidad)
```

Listing 21: Índice de hábitat favorable: bosque, densidad y accesibilidad

Código para el cálculo del viento relativo

```
# Se divide la desviación estándar del viento entre su media (con
    pequeño ajuste)
# para obtener una medida de inestabilidad o variabilidad relativa.

lobos$viento_relativo <- lobos$viento_sd /
                         (lobos$viento_mean + 0.1)
```

Listing 22: Cálculo del viento relativo como cociente entre variabilidad y media ajustada

Código para la selección de variables por todos los métodos vistos

Inicialización de procesamiento en paralelo

```
library(parallel)
library(doParallel)

GS_T0 <- Sys.time()

cluster <- makeCluster(detectCores() - 1)  # dejar 1 núcleo libre
   para el sistema
registerDoParallel(cluster)                # registrar el backend
   paralelo
```

Listing 23: Configuración del entorno paralelo para acelerar el proceso de selección

Definir variables independientes y dependientes

```
vardep <- "presencia"

lista_indep <- c("elevacion", "bio1", "bio12", "bio5", "bio6", "
   bio4", "densidad",
                 "accesibilidad", "porc_bosque", "viento_mean", "
                    indice_habitat",
                 "viento_relativo", "presion_humana", "porc_vecinos
                    _con_lobo")

archivo1 <- lobos_clima_geo[, c(lista_indep, vardep)]
archivo1 <- sf::st_drop_geometry(archivo1)
archivo1 <- na.omit(archivo1)

x <- archivo1[, lista_indep]
y <- archivo1[, vardep]
```

Listing 24: Selección de variables y eliminación de geometría

a) Selección de variables con Stepwise AIC

```
# Modelo completo con todas las variables
full <- glm(presencia ~ ., data = archivo1, family = binomial(link
   = "logit"))

# Modelo nulo sin ninguna variable
null <- glm(presencia ~ 1, data = archivo1, family = binomial(link
   = "logit"))

library(MASS)
modelo_aic <- stepAIC(null, scope = list(upper = full),
```

```
                        direction = "both", trace = FALSE)

vec <- names(coef(modelo_aic))[-1]  # eliminar el intercepto
length(vec)
dput(vec)
```

Listing 25: Aplicación de Stepwise con criterio AIC

b) Selección de variables con Stepwise BIC

```
# Modelo completo y nulo (como en AIC)
full <- glm(presencia ~ ., data = archivo1, family = binomial(link
   = "logit"))
null <- glm(presencia ~ 1, data = archivo1, family = binomial(link
   = "logit"))

library(MASS)

# Penalización BIC: k = log(n), siendo n = número de observaciones
step_bic <- stepAIC(null, scope = list(upper = full), direction = "
   both",
                    family = binomial(link = "logit"), trace =
                       FALSE,
                    k = log(nrow(archivo1)))

vec <- names(step_bic[[1]])[-1]
length(vec)
dput(vec)
```

Listing 26: Aplicación de Stepwise con penalización BIC

c) Stepwise con penalización general (k personalizado)

```
full <- glm(presencia ~ ., data = archivo1, family = binomial(link
   = "logit"))
null <- glm(presencia ~ 1, data = archivo1, family = binomial(link
   = "logit"))

library(MASS)

selec1 <- stepAIC(null, scope = list(upper = full), direction = "
   both",
                  family = binomial(link = "logit"), trace = FALSE,
                      k = 15)

vec <- names(selec1[[1]])[-1]
length(vec)
dput(vec)
```

Listing 27: Aplicación de Stepwise con penalización personalizada k = 15

d) Selección de variables mediante RFE (Recursive Feature Elimination)

```
library(caret)

# Configuración del control para RFE usando random forest
control <- rfeControl(functions = rfFuncs, method = "cv", number =
   4)

# Ejecución del algoritmo RFE
results <- rfe(x, y, sizes = c(1:25), rfeControl = control)

# Visualización de resultados por número de variables
cosa <- as.data.frame(results$results)

# Gráfico del MAE según número de variables
ggplot(cosa, aes(y = MAE, x = Variables)) +
  geom_point() + geom_line() +
  scale_y_continuous(breaks = cosa$MAE) +
  scale_x_continuous(breaks = cosa$Variables) +
  labs(title = "RFE")
```

Listing 28: Aplicación del método RFE con random forest y validación cruzada

```
# Seleccionar las 6 primeras variables del ranking óptimo
selecrfe <- results$optVariables[1:6]
dput(selecrfe)

# También se puede ampliar a las 9 mejores
selecrfe <- results$optVariables[1:9]
dput(selecrfe)
```

Listing 29: Extracción de variables seleccionadas por RFE

e) Selección de variables con MMPC

```
library(MXM)

mmpc1 <- MMPC(y, x, max_k = 5, threshold = 0.01,
              test = "testIndLogistic", hash = FALSE)

# Extraer nombres de variables seleccionadas
a <- dput(names(archivo1[, mmpc1@selectedVars]))
length(a)
```

Listing 30: MMPC con umbral de significación 0.01

```
mmpc5 <- MMPC(y, x, max_k = 5, threshold = 0.1,
              test = "testIndLogistic", hash = FALSE)

a <- dput(names(archivo1[, mmpc5@selectedVars]))
```

```
length(a)
```

Listing 31: MMPC con umbral de significación 0.1

```
mmpc10 <- MMPC(y, x, max_k = 5, threshold = 0.1,
               test = "testIndLogistic", hash = FALSE)

a <- dput(names(archivo1[, mmpc10@selectedVars]))
length(a)
```

Listing 32: MMPC con umbral de significación 0.1 y diferente semilla

f) Comparación de modelos según las variables seleccionadas

```
library(caret)
library(reshape)
library(ggplot2)

# Función cruzalin para validar modelos y obtener AUCs
cruzalin <- function(vardep, listconti, data, repe, seed,
   nombremodelo) {

  control <- trainControl(method = "cv",
                          number = 4,
                          savePredictions = "all",
                          classProbs = TRUE,
                          summaryFunction = twoClassSummary)

  formula <- as.formula(paste(vardep, "~", paste(listconti,
     collapse = " + ")))
  resultados <- data.frame()

  for (i in 1:repe) {
    set.seed(seed * i)

    modelo <- train(formula,
                    data = data,
                    method = "glm",
                    trControl = control,
                    family = binomial,
                    metric = "ROC")

    auc <- modelo$results$ROC
    resultados <- rbind(resultados, data.frame(AUC = auc))
  }

  resultados$modelo <- nombremodelo
  return(resultados)
}
```

Listing 33: Función para validación cruzada y cálculo de AUC en modelos GLM

```r
archivo1$presencia <- factor(archivo1$presencia)

medias1 <- cruzalin(vardep, listconti1, archivo1, repe = 20, seed =
    123, nombremodelo = "StepAIC-11")
medias2 <- cruzalin(vardep, listconti2, archivo1, repe = 20, seed =
    123, nombremodelo = "StepBIC-10")
medias3 <- cruzalin(vardep, listconti3, archivo1, repe = 20, seed =
    123, nombremodelo = "K-9")
medias4 <- cruzalin(vardep, listconti4, archivo1, repe = 20, seed =
    123, nombremodelo = "RFE-6")
medias5 <- cruzalin(vardep, listconti5, archivo1, repe = 20, seed =
    123, nombremodelo = "RFE-9")
medias6 <- cruzalin(vardep, listconti6, archivo1, repe = 20, seed =
    123, nombremodelo = "MMPC-6")
medias7 <- cruzalin(vardep, listconti7, archivo1, repe = 20, seed =
    123, nombremodelo = "rf-7")

union1 <- rbind(medias1, medias2, medias3, medias4, medias5,
   medias6, medias7)
```

Listing 34: Comparación de métodos de selección de variables usando AUC

```r
library(ggplot2)

grafico <- ggplot(union1, aes(x = modelo, y = AUC, fill = modelo))
   +
  geom_boxplot(alpha = 0.7) +
  labs(title = "Comparación de modelos según AUC",
       x = "Modelo",
       y = "AUC (Area bajo la curva ROC)") +
  theme_minimal()


grafico
```

Listing 35: Gráfico de comparación de modelos según AUC

f) Entrenamiento de un modelo Random Forest

```r
set.seed(12345)

# Definimos una cuadrícula para el hiperparámetro mtry
rfgrid <- expand.grid(mtry = 10)

# Configuramos la validación cruzada
control <- trainControl(method = "cv", number = 4,
                        savePredictions = "all")

# Entrenamos el modelo Random Forest
rf <- train(presencia ~ ., data = archivo1,
```

```
              method = "rf",
              trControl = control,
              linout = TRUE,
              tuneGrid = rfgrid,
              replace = TRUE,
              importance = TRUE)

rf   # Resumen del modelo entrenado
```

Listing 36: Entrenamiento de un modelo Random Forest con validación cruzada

```
# Extraemos el modelo final entrenado
final <- rf$finalModel

# Obtenemos la tabla de importancia y la ordenamos
tabla <- as.data.frame(final$importance)
tabla <- tabla[order(-tabla[, 1]), ]
tabla   # Mostramos la tabla ordenada por importancia

# Gráfico de importancia de variables
barplot(tabla$`%IncMSE`, names.arg = rownames(tabla), las = 2)

# Extraemos los nombres de las variables más importantes
dput(row.names(tabla))
```

Listing 37: Extracción y visualización de la importancia de variables en el Random Forest

f.1) Estimación del número óptimo de árboles (OOB)

```
library(randomForest)

# Entrenamos un modelo para observar el error OOB con diferentes nú
   meros de árboles
rfbis <- randomForest(presencia ~ ., data = archivo1,
                      mtry = 10,
                      ntree = 500,
                      nodesize = 10,
                      replace = TRUE)

# Visualizamos el error medio cuadrático por número de árbol
plot(rfbis$mse)
```

Listing 38: Cálculo del error OOB para estimar el número óptimo de árboles

f.2) Random Forest con subconjuntos de variables seleccionadas

```

# Modelo 1 - Variables de lista 4 (por ejemplo: RFE-6)
rfgrid <- expand.grid(mtry = c(2,3,4,5,6,7,8,9))
datos2 <- archivo1[, c(lista4, vardep)]
```

```r
datos2$presencia <- factor(datos2$presencia, levels = c(0, 1),
   labels = c("No", "Yes"))

rf <- train(presencia ~ ., data = datos2, method = "rf",
            trControl = control, linout = TRUE,
            tuneGrid = rfgrid, replace = TRUE,
            importance = TRUE, ntree = 200)
rf   # mtry óptimo: 3

# Modelo 2 - Variables de lista 5 (RFE-9)
rfgrid <- expand.grid(mtry = c(2:11))
datos2 <- archivo1[, c(lista5, vardep)]
datos2$presencia <- factor(datos2$presencia, levels = c(0, 1),
   labels = c("No", "Yes"))

rf <- train(presencia ~ ., data = datos2, method = "rf",
            trControl = control, linout = TRUE,
            tuneGrid = rfgrid, replace = TRUE,
            importance = TRUE, ntree = 200)
rf   # mtry óptimo: 2

# Modelo 3 - Variables de lista 6 (MMPC-6)
rfgrid <- expand.grid(mtry = c(2:9))
datos2 <- archivo1[, c(lista6, vardep)]
datos2$presencia <- factor(datos2$presencia, levels = c(0, 1),
   labels = c("No", "Yes"))

rf <- train(presencia ~ ., data = datos2, method = "rf",
            trControl = control, linout = TRUE,
            tuneGrid = rfgrid, replace = TRUE,
            importance = TRUE, ntree = 200)
rf   # mtry óptimo: 6
```

Listing 39: Comparación de Random Forest con diferentes listas de variables

g.1) Comparación de modelos Random Forest (cruzada por AUC)

```r
cruzarf <- function(vardep, listconti, data, repe, seed,
   nombremodelo,
                    mtry = 3, ntree = 200, nodesize = 10) {

  # Variable dependiente como factor
  archivo1[[vardep]] <- factor(archivo1[[vardep]], levels = c(0, 1)
     , labels = c("No", "Yes"))

  control <- trainControl(
    method = "cv",
    number = 4,
    savePredictions = "final",
    classProbs = TRUE,
    summaryFunction = twoClassSummary
```

```
  )

  formula <- as.formula(paste(vardep, "~", paste(listconti,
     collapse = " + ")))
  resultados <- data.frame()

  for (i in 1:repe) {
    set.seed(seed * i)

    modelo <- train(
      formula, data = data, method = "rf", metric = "ROC",
      trControl = control, tuneGrid = data.frame(mtry = mtry),
      ntree = ntree, nodesize = nodesize, replace = TRUE,
         importance = TRUE
    )

    auc <- modelo$results$ROC
    resultados <- rbind(resultados, data.frame(AUC = auc))
  }

  resultados$modelo <- nombremodelo
  return(resultados)
}
```

Listing 40: Función de validación cruzada para comparar modelos Random Forest

```
medias1 <- cruzarf(vardep, lista1, archivo1, repe = 10, seed = 123,
                   mtry = 3, ntree = 200, nodesize = 10,
                   nombremodelo = paste("lista1rf", length(lista1),
                       sep = "-"))

medias2 <- cruzarf(vardep, lista2, archivo1, repe = 10, seed = 123,
                   mtry = 2, ntree = 200, nodesize = 10,
                   nombremodelo = paste("lista2rf", length(lista2),
                       sep = "-"))

medias3 <- cruzarf(vardep, lista3, archivo1, repe = 10, seed = 123,
                   mtry = 6, ntree = 200, nodesize = 10,
                   nombremodelo = paste("lista3rf", length(lista3),
                       sep = "-"))

# Unimos los resultados
union1 <- rbind(medias1, medias2, medias3)
```

Listing 41: Comparación de 3 modelos RF usando listas de variables distintas

```
num_modelos <- length(unique(union1$modelo))
colores <- rainbow(num_modelos)

par(cex.axis = 0.8, las = 2)
grafi1 <- boxplot(AUC ~ modelo, data = union1, col = colores)
```

Listing 42: Boxplot comparativo de modelos Random Forest por AUC

h.1) Búsqueda de hiperparámetros para GBM

```
gbmgrid <- expand.grid(
  shrinkage = c(0.04, 0.03, 0.01, 0.02, 0.008, 0.005, 0.003, 0.002)
     ,
  n.minobsinnode = c(20),
  n.trees = c(1000, 2000, 3000),
  interaction.depth = c(2)
)

control <- trainControl(
  method = "cv",
  number = 4,
  savePredictions = "final",
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)

# Entrenamiento GBM
gbm <- train(
  presencia ~ ., data = datos2,
  method = "gbm",
  trControl = control,
  tuneGrid = gbmgrid,
  metric = "ROC",
  distribution = "bernoulli",
  bag.fraction = 1,
  verbose = FALSE
)

plot(gbm)
gbm
```

Listing 43: Grid search de hiperparámetros en GBM con validación cruzada

h.2) Búsqueda de hiperparámetros para XGBoost

```
xgbmgrid <- expand.grid(
  min_child_weight = c(5, 10, 20),
  eta = c(0.1, 0.05, 0.03, 0.01, 0.001),
  nrounds = c(100, 500, 1000, 5000),
  max_depth = 6,
  gamma = 0,
  colsample_bytree = 1,
  subsample = 1
)

control <- trainControl(
  method = "cv",
  number = 4,
  savePredictions = "final",
```

```
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)

xgbm <- train(
  prediccion ~ ., data = datos2,
  method = "xgbTree",
  trControl = control,
  tuneGrid = xgbmgrid,
  metric = "ROC",
  verbose = FALSE
)

plot(xgbm)
xgbm
```

Listing 44: Optimización de hiperparámetros para XGBoost usando caret

h.3) Validación cruzada manual en CatBoost con cálculo de AUC

```
library(catboost)
library(pROC)

# Convertir presencia a factor
datos2$presencia <- factor(datos2$presencia, levels = c(0, 1),
   labels = c("No", "Yes"))

# Crear folds
set.seed(123)
folds <- createFolds(datos2$presencia, k = 4)
auc_list <- c()

# Loop de validación cruzada
for (i in 1:4) {
  test_idx <- folds[[i]]
  train_data <- datos2[-test_idx, ]
  test_data  <- datos2[ test_idx, ]

  train_pool <- catboost.load_pool(
    data = train_data[, setdiff(names(train_data), "presencia")],
    label = as.numeric(train_data$presencia == "Yes")
  )

  test_pool <- catboost.load_pool(
    data = test_data[, setdiff(names(test_data), "presencia")],
    label = as.numeric(test_data$presencia == "Yes")
  )

  modelo <- catboost.train(
    train_pool,
    params = list(
```

```
      loss_function = "Logloss",
      iterations = 500,
      depth = 6,
      learning_rate = 0.1,
      eval_metric = "AUC",
      logging_level = "Silent"
    )
  )

  pred_probs <- catboost.predict(modelo, test_pool, prediction_type
      = "Probability")
  auc_val <- auc(test_data$presencia, pred_probs)
  auc_list[i] <- auc_val
}

# AUC medio
mean_auc <- mean(auc_list)
print(paste("AUC promedio en validación cruzada:", round(mean_auc,
    4)))
```

Listing 45: Entrenamiento manual de CatBoost con validación cruzada y cálculo de AUC

i.1) Validación cruzada para GLM (regresión logística)

```
cruzalin <- function(vardep, listconti, data, repe, seed,
    nombremodelo) {
  control <- trainControl(method = "cv", number = 4,
      savePredictions = "all", classProbs = TRUE)
  formula <- as.formula(paste(vardep, "~", paste(listconti,
      collapse = " + ")))
  resultados <- data.frame()

  for (i in 1:repe) {
    set.seed(seed * i)
    modelo <- train(formula, data = data, method = "glm",
                    family = "binomial", trControl = control)
    acc <- modelo$results$Accuracy
    resultados <- rbind(resultados, data.frame(Accuracy = acc))
  }

  resultados$modelo <- nombremodelo
  return(resultados)
}
```

Listing 46: Función de validación cruzada para modelo GLM

i.2) Validación cruzada para GBM (Gradient Boosting Machine)

```
cruzagbm <- function(vardep, listconti, data, repe, seed,
    nombremodelo,
```

```
                        shrinkage = 0.1, n.trees = 200, n.minobsinnode
                            = 10, bag.fraction = 1) {
  control <- trainControl(method = "cv", number = 4,
     savePredictions = "all", classProbs = TRUE)
  formula <- as.formula(paste(vardep, "~", paste(listconti,
     collapse = " + ")))
  gbmgrid <- expand.grid(interaction.depth = c(2), n.trees = n.
     trees,
                         shrinkage = shrinkage, n.minobsinnode = n.
                            minobsinnode)

  resultados <- data.frame()

  for (i in 1:repe) {
    set.seed(seed * i)
    modelo <- train(formula, data = data, method = "gbm",
                    trControl = control, tuneGrid = gbmgrid, bag.
                       fraction = bag.fraction,
                    distribution = "bernoulli", verbose = FALSE)

    acc <- modelo$results$Accuracy
    resultados <- rbind(resultados, data.frame(Accuracy = acc))
  }

  resultados$modelo <- nombremodelo
  return(resultados)
}
```

Listing 47: Función de validación cruzada para modelo GBM

i.3) Validación cruzada para XGBoost

```
vcruzaxgbm <- function(vardep, listconti, data, repe, seed,
   nombremodelo,
                       eta = 0.1, nrounds = 200, min_child_weight =
                           10,
                       colsample_bytree = 1, subsample = 1) {
  control <- trainControl(method = "cv", number = 4, classProbs =
     TRUE)
  formula <- as.formula(paste(vardep, "~", paste(listconti,
     collapse = " + ")))

  xgbmgrid <- expand.grid(
    min_child_weight = min_child_weight,
    eta = eta,
    nrounds = nrounds,
    max_depth = 6,
    gamma = 0,
    colsample_bytree = colsample_bytree,
    subsample = subsample
  )
```

```
  resultados <- data.frame()

  for (i in 1:repe) {
    set.seed(seed * i)
    modelo <- train(formula, data = data, method = "xgbTree",
                    trControl = control, tuneGrid = xgbmgrid)

    acc <- modelo$results$Accuracy
    resultados <- rbind(resultados, data.frame(Accuracy = acc))
  }

  resultados$modelo <- nombremodelo
  return(resultados)
}
```

Listing 48: Función de validación cruzada para XGBoost

i.4) Validación cruzada para CatBoost

```
cruzadacatboostbin <- function(data, vardep = "vardep", listconti =
    "listconti", listclass = "listclass",
                               grupos = 4, sinicio = 1234, repe =
                                  5,
                               learning_rate = 0.05, iterations =
                                  500, depth = 6,
                               rsm = 0.8, l2_leaf_reg = 0, border_
                                  count = 400,
                               min_data_in_leaf = 10, subsample =
                                  1) {

  library(caret)
  library(catboost)
  library(pROC)

  databis <- if (any(listclass != "")) data[, c(vardep, listconti,
     listclass)] else data[, c(vardep, listconti)]

  set.seed(sinicio)
  control <- trainControl(method = "repeatedcv", number = grupos,
     repeats = repe,
                          savePredictions = "all", classProbs =
                             TRUE)

  catgrid <- expand.grid(learning_rate = learning_rate, iterations
     = iterations,
                         depth = depth, rsm = rsm, l2_leaf_reg = l2
                            _leaf_reg, border_count = border_count)

  x <- databis; x[, vardep] <- NULL; y <- databis[, vardep]
```

```
  catboo <- train(x, y, method = catboost.caret, preProc = NULL,
                  tuneGrid = catgrid, trControl = control,
                  min_data_in_leaf = min_data_in_leaf, subsample =
                     subsample)

  pred <- as.data.frame(catboo$pred)
  pred$prueba <- strsplit(pred$Resample, "[.]")
  pred$Fold <- sapply(pred$prueba, "[", 1)
  pred$Rep <- sapply(pred$prueba, "[", 2)
  pred$prueba <- NULL

  tasafallos <- function(x, y) 1 - confusionMatrix(x, y)[[3]][1]
  tasa <- aggregate(pred[, c("pred", "obs")], by = list(pred$Rep),
     FUN = function(df) 1 - tasafallos(df$pred, df$obs))

  aucs <- aggregate(pred$Yes, by = list(pred$Rep), FUN = function(p
     ) {
    suppressMessages(auc(pred$obs, p))
  })

  medias <- data.frame(tasa = 1 - tasa$pred, auc = aucs$x)
  return(medias)
}
```

Listing 49: Validación cruzada para CatBoost con cálculo de AUC y tasa de acierto

k.1) Entrenamiento del modelo final con CatBoost

```
# Definimos la cuadrícula con hiperparámetros óptimos
catgrid <- expand.grid(
  learning_rate = 0.1,
  iterations = 100,
  depth = 6,
  rsm = 0.8,
  l2_leaf_reg = 1,
  border_count = 100
)

# Entrenamos el modelo final
modelo_catboost <- train(x, y,
                         method = catboost.caret,
                         tuneGrid = catgrid,
                         trControl = fit_control,
                         logging_level = "Silent",
                         min_data_in_leaf = 10,
                         subsample = 1)
```

Listing 50: Configuración y entrenamiento del modelo CatBoost

k.2) Evaluación con matriz de confusión

```r
library(caret)

# Predicciones del modelo
pred <- predict(modelo_catboost, newdata = x)

# Matriz de confusión
confusionMatrix(pred, y)
```

Listing 51: Cálculo de predicciones y matriz de confusión para CatBoost

k.3) Cálculo del AUC y representación de la curva ROC

```r
library(pROC)

# Probabilidades de predicción para la clase positiva
probs <- predict(modelo_catboost, newdata = x, type = "prob")

# Curva ROC
roc_obj <- roc(response = y, predictor = probs[, "Yes"])

# AUC
auc(roc_obj)

# Gráfico ROC
plot(roc_obj,
     col = "#1c61b6", lwd = 2,
     main = "Curva ROC - Modelo CatBoost",
     print.auc = TRUE)
```

Listing 52: Cálculo y representación gráfica de la curva ROC

l.1) Preparación de variables y predicción sobre el grid

```r
variables_modelo <- c("bio1", "viento_mean", "bio12",
                      "presion_humana", "porc_bosque", "bio6", "
                         bio4",
                      "viento_relativo", "elevacion", "densidad")

# Seleccionar variables relevantes del grid completo
hexgrid_modelo <- hexgrid_completo %>%
  dplyr::select(all_of(variables_modelo))

# Eliminar geometría para predicción
datos_pred <- st_drop_geometry(hexgrid_modelo)

# Realizar predicciones de probabilidad con XGBoost
predicciones <- predict(modelo_xgb, newdata = datos_pred, type = "
   prob")[, 2]
```

Listing 53: Generación de predicciones sobre hexágonos usando el modelo XGBoost

l.2) Visualización del mapa de probabilidades de presencia

```
library(ggplot2)
library(sf)

ggplot() +
  geom_sf(data = peninsula, color = "black", fill = NA) +
  geom_sf(data = hexgrid_modelo, aes(fill = predicciones), color =
     NA) +
  scale_fill_viridis_c(option = "plasma", name = "Prob. presencia
     Lobo Ibérico") +
  coord_sf() +
  theme_minimal()
```

Listing 54: Mapa de probabilidad de presencia del Lobo Ibérico según XGBoost