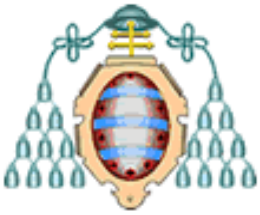


UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE MIERES

GRADO EN INGENIERÍA FORESTAL Y DEL MEDIO NATURAL

DEPARTAMENTO DE BIOLOGÍA DE ORGANISMOS Y SISTEMAS
AREA DE INGENIERÍA AGROFORESTAL

TRABAJO FIN DE GRADO

DESARROLLO DE SOFTWARE SELECTOR DE ESPECIES PARA REPOBLACIONES FORESTALES EN PYTHON

AUTOR: DANIEL PFITZER LÓPEZ

TUTORES: CARLOS ANTONIO LÓPEZ SÁNCHEZ & MARCOS BARRIO ANTA

MAYO, 2022

Índice general

Índice de figuras

Índice de tablas

Siglas

AFLIBER *Atlas of the Flora Iberica Database*.

BBDD Base de datos.
Bin Binomial.

CC Caracteres culturales.
CMD *Command* - Símbolo del sistema.
CR En peligro crítico.
CRFAA Catálogo Regional de Flora Amenazada de Asturias.
CSV *Comma-separated values*
Valores separados por comas.

DCC Diccionario.
DD Datos insuficientes.

EN En peligro.

FFC Funciones de comprobación.
FI Fundamentos de informática.

HT Sensibilidad a heladas tardías.

IDE *Integrated Development Environment*
Entorno de desarrollo integrado.

LC Preocupación menor.

MCA Montaña Central Asturiana.

MF25 Mapa Forestal de Asturias.

NT Casi amenazado.

PM Profundidad mínima.
PMA Precipitación media anual.
PV Precipitación estival.

RC Rango de datos compatibles.
RCO Rango de datos compatibles y óptimos.

SE Sensibilidad a encharcamientos.
SGBD Sistema gestor de la base de datos.
SIG Sistema de información geográfica.
SQL *Structured Query Language*
Lenguaje de consulta estructurada.
sspp. Especies forestales.

TMA Temperatura media anual.
TMC Temperatura media del mes más cálido.
TMF Temperatura media del mes más frío.

UICN Unión Internacional para la Conservación de la Naturaleza.

VNU Valor numérico único.
VU Vulnerable.

1 Justificación y objetivos

Desde Nicolás y Gandullo (1967), primer estudio sobre las principales especies arbóreas españolas, el volumen de investigaciones sobre autoecología[1] de las especies con mayor representación territorial e importancia económica se ha visto notablemente incrementada (Lopez-Senespleda y col., 2018, ver Figura 1.1). Aún con recopilaciones como Zazo y col. (2000) o Garcia y col. (2014), la información se encuentra disgregada y el manejo de tan extensa literatura científica resulta tedioso, complicando la labor de identificación de especies forestales compatibles para repoblaciones[2]. Además, a raíz de la alteración de distribuciones y abundancias de las especies forestales producida por el cambio climático (Hughes, 2000), se actualiza e incrementa la cantidad de estudios de la materia, dificultando aun más el manejo de tanta información.

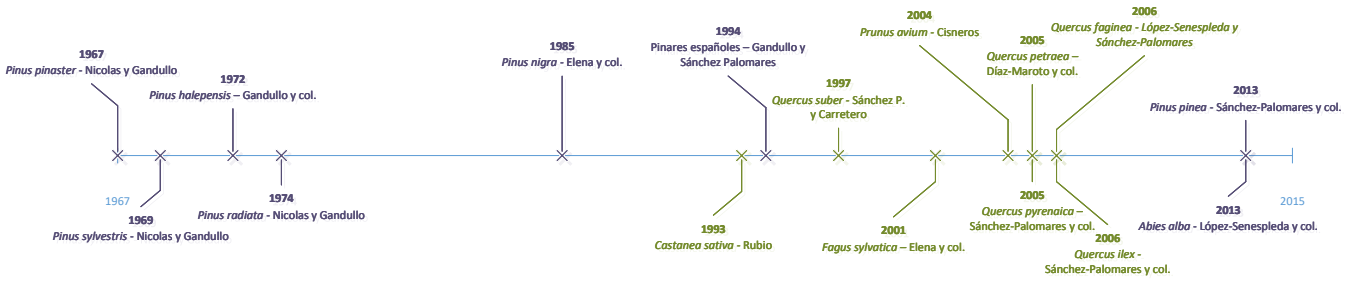


Figura 1.1: Cronología de algunas de las publicaciones de autoecología de especies forestales en España. En verde frondosas, en morado coníferas. Elaboración propia a partir de Lopez-Senespleda y col. (2018)

Este trabajo consiste en el planteamiento de un programa informático que proporcione una ventana única a la consulta de información sobre especies forestales en Asturias, centrado en los caracteres culturales y el estado de conservación de las mismas. Tal programa informático se compone de dos piezas fundamentales: una base de datos (BBDD), que contiene toda la información recopilada sobre las especies, y un algoritmo capaz de leer esta BBDD, manejarla y realizar consultas, es decir, un sistema gestor de la BBDD. Es evidente que la calidad de los datos en la BBDD condiciona la validez del resultado final, sin embargo, el foco del trabajo está más centrado en el gestor de la BBDD: cómo se leen y manejan los datos. Los **objetivos** que se persiguen, esquematizados en la Figura 1.2 , son los siguientes:

1. Elaborar un algoritmo (sistema gestor de la BBDD) capaz de:

    a) Realizar consultas de idoneidad climática, edáfica y fisiográfica en base a parámetros conocidos de una zona para determinar especies forestales potenciales para trabajos de forestación o repoblación forestal.

[1] Ciencia que estudia las relaciones de las plantas o comunidades vegetales con el medio en el que habitan

[2] Conjunto de técnicas necesarias para crear una masa forestal, formada por especies vegetales leñosas, que sea estable con el medio, en un terreno cuya vegetación actual es ineficaz en mayor o menor grado según el uso asignado al territorio y que cumpla los fines que de ella se demanden (Serrada, 1993)

b) Realizar consultas sobre cualquier tipo de información recogida en la BBDD (requerimientos climáticos, edáficos, botánica sistemática, estado de conservación...).

2. Elaborar una BBDD con las principales especies forestales de Asturias, cuya estructura sea flexible, que permita una fácil incorporación o actualización de datos con ayuda del gestor de BBDD y permita validar el correcto funcionamiento del algoritmo.

3. Documentar el funcionamiento y estructura de la base de datos y del algoritmo (SGBD), de tal forma que cualquier usuario pueda entender su mecánica interna[3] y emplear la herramienta.

4. Validar si el programa funciona correctamente o no.

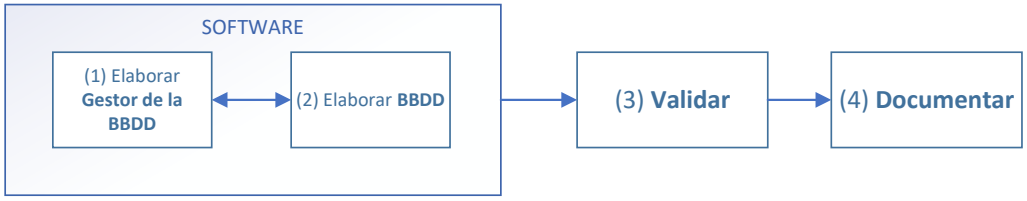


Figura 1.2: Esquema de los objetivos del trabajo. Los procesos (1) y (2) son simultáneos. En caso de que la validación (3) falle, se revisaría y alteraría (1) y (2)

Otra de las justificaciones del trabajo es la falta de antecedentes de proyectos similares, aparte de la reciente aplicación informática ModERFoRest[4] (Alonso y col., 2022), y que, aunque en el plan de estudios del Grado en Ingeniería Forestal y del Medio Natural exista una asignatura de fundamentos de informática (FI), estas competencias no se acaban de desarrollar, aprovechar ni integrar en el resto de asignaturas. Parte de la motivación del trabajo parte de buscar integrar nociones y competencias propias del título[5] con las adquiridas en FI. Es también por este motivo por el que se ha elegido Python (Van Rossum y Drake Jr, 1995) como lenguaje de programación; a pesar de SQL[6] se adapta ligeramente mejor al planteamiento del trabajo, se ha optado por Python por ser uno de los lenguajes integrados en plan de estudios.

También se plantea la posibilidad de, en un futuro, continuar con la labor mejorando y añadiendo funcionalidades de la herramienta, dotarla de una BBDD de mayor calidad e integrarla en un Sistema de Información Geográfica (SIG), lo cual la volvería mucho más accesible e intuitiva.

[3] No es imprescindible comprender todo el funcionamiento interno del software para su uso, pero sí recomendable

[4] *Modeling Environmental Requirements for Forest Restoration*, desarrollado por Alonso y col. (2022). Permite llevar a cabo diversos procesos de modelización y cálculo en el campo de la ecología forestal: distribución de especies, adecuación de especies para la restauración forestal y similitud entre estaciones forestales

[5] Tales como Selvicultura, Botánica Forestal, Gestión de Especies Amenazadas, SIG Aplicado a la Gestión de Sistemas Naturales, Edafología y Geomorfología...

[6] *Structured Query Language*, lenguaje de programación orientado a trabajar con conjuntos de datos y sus relaciones

Parte I

Descripción y funcionamiento interno del programa

2 Introducción

2.1. Descripción y funcionalidades del programa

La principal función consiste en asistir y optimizar el proceso de selección de especie forestal en trabajos de repoblación o reforestación: conocida la calidad de estación[1] en un emplazamiento determinado, el programa sugiere una serie de especies forestales compatibles según sus requerimientos (caracteres culturales), ordenadas de mayor a menor idoneidad, para que el usuario decida la especie definitiva. Es decir, el programa asiste proponiendo una serie de especies forestales compatibles con la estación, la cual define el usuario a partir de hasta 10 de los 23 parámetros[2] totales a través de una consulta. En este sentido, puede entenderse el programa como una «calculadora forestal».

También permite realizar búsquedas sobre de cualquier tipo de información en la BBDD: nombres comunes de cierta especie, sus requerimientos, cuántos géneros contiene, qué especies están incluidas en el *Catálogo Regional de Flora Amenazada de Asturias* (AA, 2016), endemismos, sspp. invasoras, procedencias, categoría en la UICN...en este sentido es un pequeño motor de búsqueda forestal.

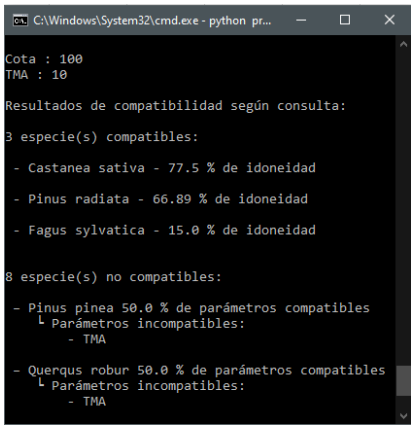


Figura 2.1: Programa ejecutándose en CMD de Windows. También es posible ejecutarlo en través de la terminal de Python en un entorno de desarrollo interactivo (IDE)

2.2. Flujos de procesos

Como se aprecia en la figura 2.1, se interactúa con el programa mediante el símbolo del sistema de Windows (CMD) o a través de la terminal de Python en una IDE[3]. El algoritmo lee la consulta propuesta y la compara con la base de datos, devolviendo una lista de especies (ver Figura 2.2). Tanto la BBDD como la consulta propuesta son archivos *CSV independientes del código del programa.

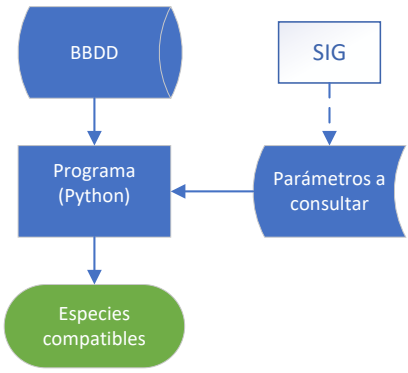


Figura 2.2: Flujo de procesos general. El programa se nutre de la BBDD y de la consulta para calcular las especies compatibles. Se plantea una conexión a un SIG que podría sustituir el archivo de la consulta. En caso de realizar búsquedas, solo se realiza lectura de la BBDD

El conjunto está compuesto por un algoritmo escrito en lenguaje de programación Python — *script* o Sistema gestor de la base de datos (SGBD), motor del programa — que se alimenta de la base

[1] Limitación productiva y de crecimiento marcada por el clima, el suelo y la fisiografía (Serrada, 1993)

[2] Tales parámetros se explican en la sección 4.1 Parámetros y atributos

[3] Se ha elegido Spyder 4.0.1 como IDE

de datos y del archivo de consultas. Se puede reducir a esas dos partes: Python (SGBD) + BBDD y el archivo de consultas. La Figura 2.3 ilustra un nivel más detallado del flujo de procesos, donde el SGBD:

1. Realiza una lectura de la BBDD.
2. Crea sus propias variables donde contiene los datos de cada especie, siguiendo una estructura concreta (ver 5.1 Variables y estructura de datos).
3. Notifica posibles errores de valores en la BBDD.
4. Lee los *N* parámetros consultados en el archivo de la consulta.
5. Realiza *N* selecciones comparando los parámetros consultados con las variables.
6. Ordena los resultados de mayor a menor compatibilidad o idoneidad.

Los *datos* pasan de la BBDD a *variables* dentro del SGBD y, tras pasar el proceso de selección de consulta del algoritmo y por el criterio del usuario, se convierten en *información*.

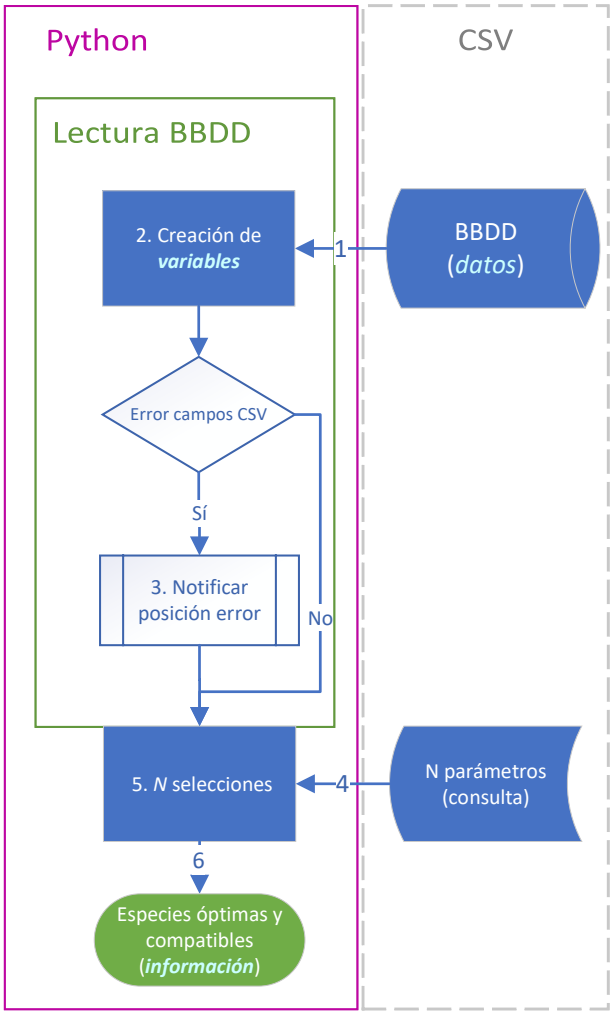


Figura 2.3: Flujo de procesos semi-detallado. equivalente a la Figura 2.2. El SGBD lee la BBDD y almacena sus datos en variables, que compara con los parámetros consultados para definir las especies compatibles

2.3. La base de datos (BBDD)

Se entiende como *Base de Datos* un conjunto de datos, numéricos y alfanuméricos, estructurado y almacenado de forma sistemática con objeto de facilitar su posterior utilización (Olaya, 2009). Para este trabajo, se ha elegido un **modelo de base de datos relacional**: una tabla con datos organizados en filas y columnas. Las columnas representan los distintos ***atributos*** (cualidades de los árboles) asociados a la tabla, mientras que las filas conforman los distintos ***registros*** (árboles). Una fila se forma con un conjunto de n atributos, constituyendo una tupla (ver Figura 2.4). Se denomina ***parámetro*** a un conjunto de atributos acerca de un tipo de dato, por ejemplo: *requerimiento de altitudes* es un parámetro, cuyos atributos son: 'altitud. mín. compatible', 'alt. máx. compatible', 'alt. mín. óptima' y 'alt. máx. óptima.'.

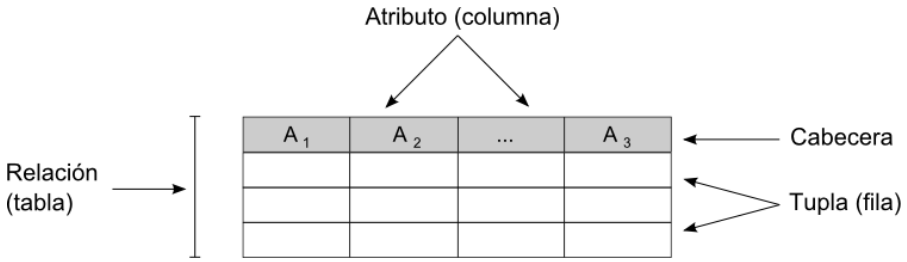


Figura 2.4: Elementos del modelo relacional de una base de datos. Cada fila (registro) equivale a una especie forestal. La cabecera contiene los nombres de los atributos A_1-A_n (cotas, precipitación...) de cada registro, con un atributo por columna. Fuente: Olaya (2009)

La BBDD contiene, entre otros, datos sobre caracteres culturales (CC) de las especies. Los caracteres culturales o selvícolas se definen como el conjunto de características de las especies forestales cuyo conocimiento es útil para su selvicultura, bien en la aplicación de tratamientos que aseguren la persistencia y estabilidad de sus masas, bien para su posible introducción por repoblación forestal (González-Vázquez, 1938). Responden, respecto de una especie forestal, a las siguientes cuestiones: dónde vive; cuáles son sus límites óptimos y de tolerancia respecto de los factores ecológicos abióticos; cómo se comporta en relación con la competencia y competición; cuál es su forma; cómo se desarrolla; cuánto vive; y cómo se reproduce (Serrada, 1993). En la sección 4. Base de datos se desarrolla con detalle qué datos almacena exactamente.

2.4. El sistema gestor de la base de datos (SGBD)

Es la pieza más importante: maneja todos los datos y se encarga de calcular y transferir todos los resultados. Está desarrollado en Python, pero se asemeja a un lenguaje SQL por su tratamiento estructurado de los datos. Como se ha visto en la Figura 2.3, el SGBD lee la BBDD, genera variables y calcula los resultados de las consultas. ¿Qué es una **variable**? Técnicamente, es *un nombre que se refiere a un objeto que reside en la memoria* (Trespaderne y Echevarría, 2019); se puede ver como una caja en la memoria del ordenador que almacena datos. El nombre es un parámetro[4] y el objeto al que apunta es una estructura de datos que contiene los valores de dicho parámetro para cada especie forestal. Por ejemplo, para las altitudes, se tendrían dos variables: cotas compatibles (`cotas_umbral`) y las cotas óptimas (`cotas_optimo`). Centrándose en la primera, las cotas compatibles, «`cotas_umbral`» sería el nombre de la variable, que apuntaría a una estructura de datos que contiene, para cada árbol, dos valores: la cota mínima y la máxima donde la especie puede existir. Los parámetros no siempre son del mismo tipo: las altitudes son rangos, la sensibilidad a encharcamiento es una variable booleana[5], el nombre científico una cadena de texto...por lo tanto, por cada tipo de parámetro, se tienen distintos tipos de estructuras de datos, pero con una característica en común: son siempre *diccionarios*.

Para entender el concepto de ***diccionario***, se puede imaginar un armario en cuyo interior hay cajones, uno por especie forestal, identificados con sus nombres científicos. Dentro de cada cajón están los datos de la especie forestal de la variable, estructurados de forma distinta según el tipo de parámetro al que pertenezca. A la especie forestal, el identificador, se le llama «llave del diccionario», y a los datos que almacena «elemento del diccionario» (ver Figura 2.5). Los nombres científicos de cada especie forestal son siempre las llaves en todos los diccionarios, por lo tanto, con un solo nombre científico de una especie se puede acceder a todos los datos para cada parámetro de esa especie.

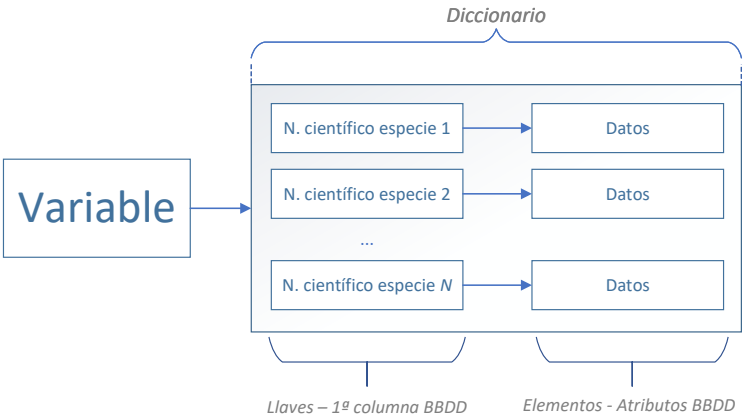


Figura 2.5: Estructura de un diccionario. Las variables en el SGBD (ver 5.1: Variables y estructura de datos) apuntan siempre a diccionarios, cuyas llaves son los nombres científicos de las especies forestales y cuyos elementos los atributos en la BBDD. La estructura de sus elementos depende del tipo de parámetro de la variable

[4] Cotas; Precipitaciones; Nombres vernáculos...ver en 5.1: Variables y estructura de datos
[5] Sólo puede tomar dos posibles valores: *True* (verdadero) o *False* (falso)

En Python, los diccionarios se escriben entre corchetes[6] con sus llaves y elementos separados por comas, tal como ilustra la Figura 2.6 a continuación:

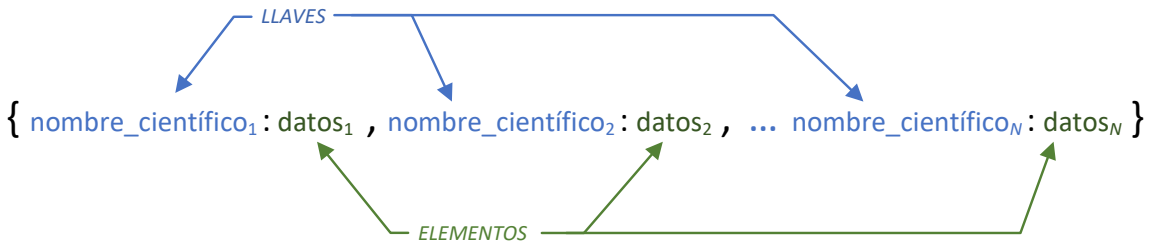
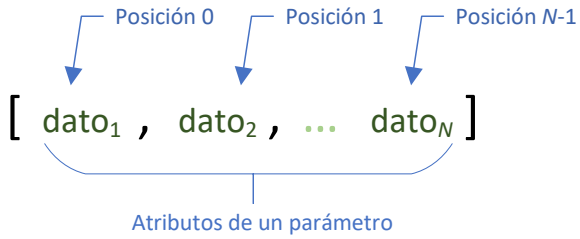


Figura 2.6: Estructura del código de un diccionario en Python, equivalente a la figura Figura 2.5. Las *llaves* son siempre los nombres científicos de las especies forestales, los *elementos* son subestructuras de datos que contienen los *atributos* del *parámetro* asociado a la *variable* que apunta al *diccionario*. En otras palabras, los elementos son las cualidades de cada especie forestal para la temática de la variable (por ejemplo: si la variable es «cotas_umbral», los datos o elementos serían sus límites inferior y superior, y el parámetro sería «altitudes». Estas subestructuras tienen forma distinta según el tipo de parámetro al que pertenezcan los atributos. Por cada parámetro, existen una o varias variables (ver en 5.1: Variables y estructura de datos). Las llaves y sus elementos se agrupan separados entre sí por dos puntos y se separan del resto por comas. Más adelante, en la figura Figura 2.8 se clarifican todas estas relaciones a partir de un ejemplo

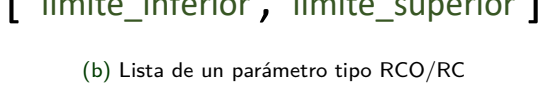
Por cada variable, se tiene un diccionario, el cual alberga los atributos de un parámetro concreto. La estructura interna de los datos (elementos) en cada diccionario depende del tipo de parámetro. El esquema a continuación sintetiza la tipología de los parámetros, incluyendo algún ejemplo en cursiva:

- Tipos parámetros
  - Rangos de valores
    - Valores compatibles y óptimos (RCO) – *Altitudes*
    - Valores compatibles (RC) – *Precipitación estival*
  - Valores numéricos únicos (VNU) – *Profundidad mínima*
  - Binomial o booleano (Bin) – *Sensibilidad a heladas tardías*
  - Cadena de texto – *Nombres científicos*

Los parámetros tipo VNU, Bin y de texto se trasladan a variables con elementos en sus diccionarios que contienen el valor directamente[7], mientras que los elementos para RCO y RC se estructuran a partir de **listas**. Las *listas* son conjuntos ordenados de (sub)elementos (atributos) con posiciones, contando desde 0 hasta los *N*-1 elementos de la lista (ver Figura 2.7a). En caso de los rangos, la posición 0 de la lista representa el límite inferior y 1 el superior sistemáticamente (ver Figura 2.7b).



(a) Estructura del código de una lista en Python



(b) Lista de un parámetro tipo RCO/RC

Figura 2.7: (a) Estructura de una lista en Python y (b) estructura de una lista para un RCO/RC. Las listas se escriben con corchetes, con elementos ordenados en posiciones (0 a *N*-1) separados por comas. Son los elementos de los diccionarios (ver Figura 2.6)

[6] Corchetes tipo llave, referido al símbolo «{ }», no confundir con la *llave* (identificador) del diccionario

[7] Con excepciones

Hasta ahora, se han introducido un buen número de conceptos acerca de la estructura de la BBDD y del SGBD: *parámetros*, *atributos*, *variables*, *diccionarios*, *listas*, *RCO*...la Figura 2.8 resume todas las relaciones entre estos conceptos a partir de un ejemplo: el almacenamiento de los datos de dos parámetros, uno de tipo RCO (altitudes) y otro VNU (profundidad mínima). En la sección 4: Base de datos se explica exactamente qué significan tales parámetros y en 5.1: Variables y estructura de datos su estructura.

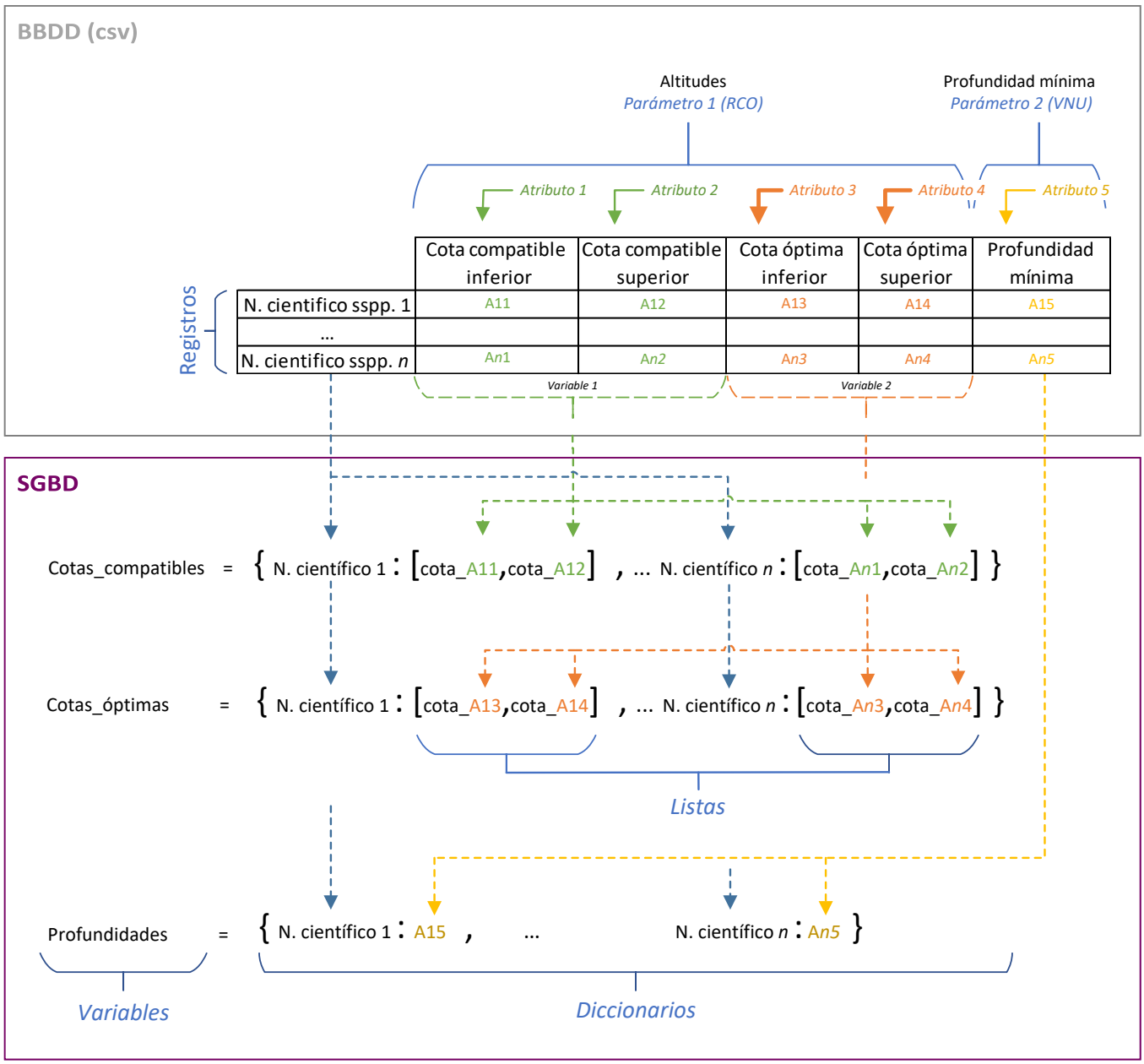


Figura 2.8: Ejemplo de almacenamiento de datos para dos parámetros distintos (RCO y VNU) y relación de conceptos introducidos. Se tienen dos parámetros: altitudes y profundidad mínima. El primero es tipo RCO, está compuesto a partir de cuatro atributos (valores de los rangos compatible y óptimo) y se define mediante dos variables («Cotas_compatibles» y «Cotas_óptimas»), una para cada rango. Las variables siempre apuntan a diccionarios, cuyas llaves son siempre los nombres científicos de las especies forestales. La forma de sus elementos depende del tipo de parámetro; en caso del primero (altitudes - RCO), los elementos son listas, que contienen los datos inferior-superior del rango en las posiciones 0 y 1. En el caso de VNU, los datos se almacenan de forma directa en el elemento. Este esquema equivale a los pasos 1 y 2 de la Figura 2.3, integra todos los conceptos de las figuras 2.4 a 2.7 y se desarrolla en 5.1: Variables y estructura de datos

La forma en la que el SGBD lee la BBDD, crea variables, muestra resultados, etc. es mediante funciones. Una **función** es *un conjunto de instrucciones al que se asigna un nombre, opcionalmente inputs y outputs, y que puede ser llamada desde otras partes de un programa para realizar una tarea concreta* (Trespaderne y Echevarría, 2019). Las funciones son las encargadas de que el SGBD «entienda» la estructura de las variables y pueda operar con ellas. Una función puede recurrir a otra función (por ejemplo, la función `cotas()`, que es la encargada de trasladar los datos de altitudes de la BBDD a variables en el SGBD, emplea las función `get_posiciones()` para buscar en qué columnas están los atributos de las altitudes) o puede incluso llamarse a si misma, técnica conocida como recursividad.

El programa se compone de dos funciones principales: `consulta()` y `buscar()`. La primera se encarga de realizar consultas y la segunda actúa como un motor de búsqueda de información en la BBDD. El funcionamiento de las funciones se detalla en 5.2: Funciones y flujo de procesos.

3 | Material y métodos

3.1. Desarrollo del software

Las fuentes de datos para la BBDD han sido muy variadas. Para los caracteres culturales, se recurrió principalmente a Bravo y Montero (2008), completando en varios casos la información con de la Torre y col. (2006), Garcia y col. (2014), Nicolás y Gandullo (1967) y Zazo y col. (2000). El área de estudio comprende toda Asturias, considerando únicamente especies forestales, invasoras y con interés de conservación distribuidas dentro de sus límites administrativos; las procedencias y naturaleza invasora de las especies se comprobaron a partir de Prieto y col. (2014). Los nombres comunes se obtuvieron de Anthos (2011). En cuanto a endemismos y sistemática, se extrajo del set de datos de AFLIBER (Ramos y col., 2021). Respecto a las amenazas, se consultó UICN (2021) y AA (2016).

El desarrollo del código se llevó a cabo en la IDE *Spyder* (Raybaut, 2009, ver Figura 3.1). Está basado en funciones, así que el procedimiento consistió en desarrollar y testear cada función de forma individual. Primero se desarrollaron las funciones más simples y, una vez comprobado que no producían ningún error, se comenzó a elaborar funciones más complejas cuyo funcionamiento se basa en las anteriores. De esta forma, resultó sencillo ubicar los fallos en el código y subsanarlos, lo que se conoce como depuración o *debugging*. Además, esta naturaleza modular del programa permite la fácil incorporación de nuevos parámetros en la BBDD o funciones en el SGBD.

Además, se desarrolló un código independiente capaz de convertir el formato de texto de los nombres vernáculos en Anthos al formato de entrada en la BBDD, evitando tener que escribirlos manualmente uno a uno. Este código está recogido en el apéndice C.3, a continuación del código principal del programa.

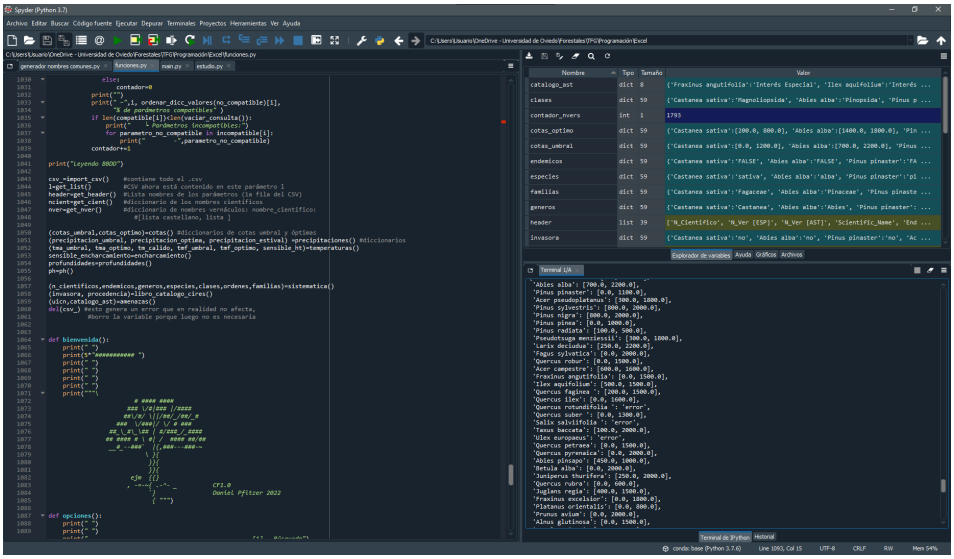


Figura 3.1: Desarrollo del código en *Spyder* 4.0.1

3.2. Validación del programa

En el apartado anterior se habla de la depuración del código para comprobar la falta de errores. Esto asegura que el programa haga exactamente lo previsto, sin embargo, no significa que los resultados que proporcione se adecúen a la realidad, es necesario realizar una validación. El método de validación consiste en realizar una simulación en un entorno controlado lo más «natural» posible. Controlado significa que se conocen detalladamente sus características fisiográficas y climáticas, además de las especies forestales que forman bosques en la zona y sus ocupaciones. «Natural» se refiere a que el origen de estos bosques no sea antrópico o, en caso de que sean repoblaciones, estén integradas con la vegetación preexistente. Si se introducen las características reales del entorno al programa, este debería sugerir una lista de especies igual a las documentadas en la realidad.

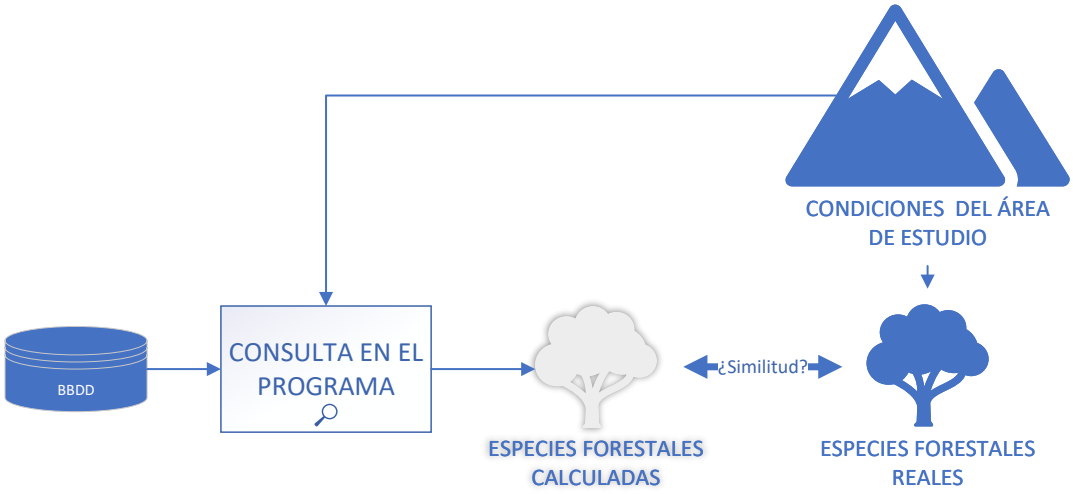


Figura 3.2: Metodología de validación del programa. A partir de las condiciones conocidas de un entorno, se comprueba si el programa calcula las especies reales

El área de estudio seleccionado es la Montaña Central Asturiana (MCA), formada por por los concejos de Ayer, Lena, Mieres, Morcín, Ribera de Arriba y Riosa. Se conocen las principales especies forestales allí presentes y sus ocupaciones a partir del MF25[1] y de Castaño-Santamara y col. (2013). Además, según el MFE25, estas masas son de origen natural. Se evita realizar una validación a partir de masas cuyo origen sea plantación porque es posible que crezcan, de forma subóptima, fuera de sus requerimientos compatibles.

Las altitudes varían a lo largo de la MCA, existiendo distintos pisos de vegetación[2] en la zona de estudio. Las consultas del programa aceptan un valor por parámetro, es decir, se pueden consultar varios parámetros (altitud, PMA, TMA...) simultáneamente, pero correspondientes a un único valor. Dado que las altitudes van desde 100 a casi 2500 metros, han de realizarse varias consultas. La solución fue automatizar estas consultas mediante Python, desarrollando un código que itera en las distintas alturas a partir de las funciones en el SGBD, memorizando los resultados de cada iteración.

La BBDD también fue validada mediante un análisis de las frecuencias de los parámetros[3], donde no se descubrió ninguna tendencia anómala en los datos.

[1] Mapa Forestal de Asturias, extraído de IDEE
[2] Conjunto ordenado de agrupaciones vegetales presentes en una latitud constante al variar la altitud (Serrada, 1993)
[3] ver en B.2: Frecuencias de los parámetros (CC))

4 | Base de datos

Como se ha expuesto en el apartado 2.3: La base de datos, se tiene un modelo relacional para la BBDD: cada registro (fila) de la tabla es una especie y cada columna un atributo. El formato de archivo de la tabla es CSV[1], con el separador de punto y coma. A pesar de ser un formato «sensible» (saltos accidentales de línea o la alteración de un valor), resulta fácil de visualizar y editar a través de hojas de cálculo de Microsoft Excel. Como se verá más adelante, el Sistema gestor de la base de datos está dotado de métodos en caso de que existan valores erróneos en alguna celda. Se trabaja con un total de 23 parámetros, definidos a partir de 39 atributos.

La BBDD contiene 59 especies, de las cuales 37 son sspp. forestales potenciales para repoblación. El resto incluye especies invasoras o flora con algún tipo de interés de conservación orientada a ser objeto de búsqueda[2].

4.1. Parámetros y atributos

Los datos recopilados y almacenados para cada especie (ver Tabla 4.1) se dividen en cuatro bloques:

(a) Botánica sistemática

(b) Requerimientos climáticos

(c) Requerimientos edafológicos

(d) Estado de conservación

Tabla 4.1: Resumen de los atributos en la BBDD (fichero CSV). Más adelante, en la Tabla 5.12 (pág. 23), se relaciona los atributos con las variables generadas en el SGBD

Bloque	Atributo	Abreviatura	Nombre en la cabecera	Tipo de dato	Fuente(s)
Botánica sistemática	Nombres vernáculos	-	N_Ver [ESP]; N_Ver [AST]	Texto	Anthos (2011)
	Nombre científico	-	Scientific_Name	Texto	Ramos y col. (2021)
	Género	-	Genus	Texto	
	Clase	-	Species	Texto	
	Orden	-	Class	Texto	
	Familia	-	Order	Texto	
	Endémico	-	Endemic	Bin	
Clima	Altitud	-	cota_min; cota_max; op_cota_min; op_cota_max	RCO	Bravo y Montero (2008), Garcia y col. (2014) y Zazo y col. (2000)
	Precipitación media anual	PMA	PMA_min; PMA_max; op_PMA_min; op_PMA_max	RCO	
	Precipitación estival	PV	PV_min; PV_max	RC	
	Temperatura media anual	TMA	TMA_max; TMA_min; op_TMA_min; op_TMA_max	RCO	
	Temperatura media del mes más calido	TMC	TMC_min; TMC_max	RC	
	Temperatura media del mes más frío	TMF	TMF_min; TMF_max; op_TMF_min; op_TMF_max	RCO	
	Sensibilidad a heladas tardías	HT	Sen Hel T	Bin	
Edafología	Profundidad mínima	-	Prof_min	VNU	
	pH	-	pH_min; pH_max	RC	
	S. encharcamiento	SE	Sen Encharcamiento	Bin	

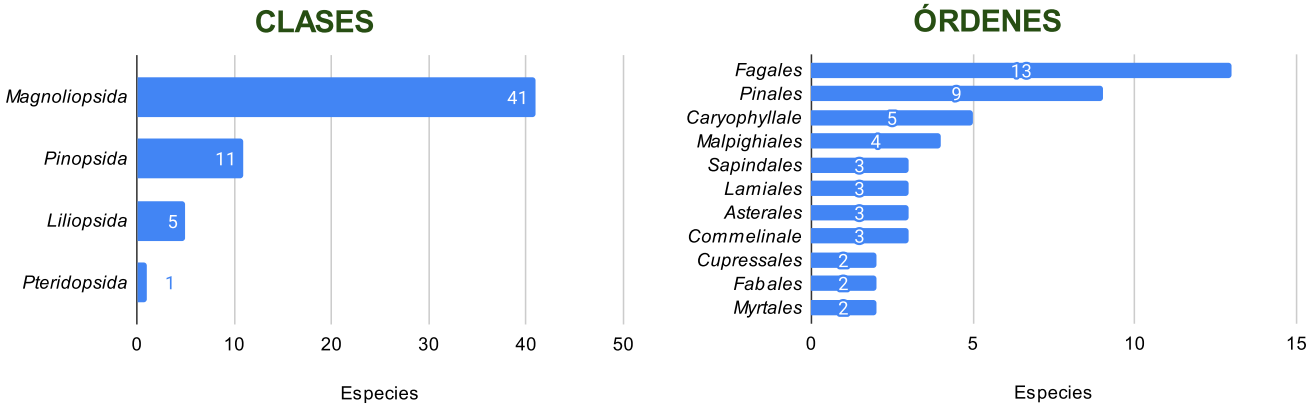
[1] *Comma-separated Values* o valores separados por comas. Las columnas se separan por comas y las filas por saltos de línea

[2] Ver 5.2.2: Búsqueda

Tal como se explica en el apartado 1 Justificación y objetivos, se ha priorizado el manejo de los datos por encima de los datos en sí. Esto se traduce una selección de atributos suficientemente representativos en vez de incluir *todos los posibles*. Por ejemplo, entre otros parámetros, no se han añadido orientaciones o temperamentos, pero no porque se consideren intrascendentes; se considera que con el resto de parámetros controlados se cumple el segundo objetivo del trabajo.

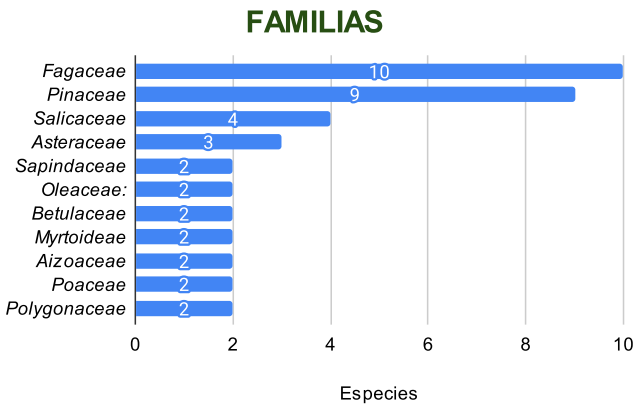
4.1.1. Botánica sistemática

Se incluye: nombre científico (con y sin autor), nombres vernáculos en asturiano y castellano (Anthos, 2011), género, clase, orden y familia. Como puede verse en la Tabla 4.1, «Endémico» está incluido en este bloque de atributos, con el motivo de pertenecer al set de datos de información taxonómica de AFLIBER (Ramos y col., 2021) empleada como fuente. Tal como muestra la Figura 4.1, las 59 especies en la BBDD están distribuidas a lo largo de 4 clases distintas, 21 órdenes, 30 familias y 40 géneros, donde los más destacados son: *Quercus*, *Pinus* y *Populus*.

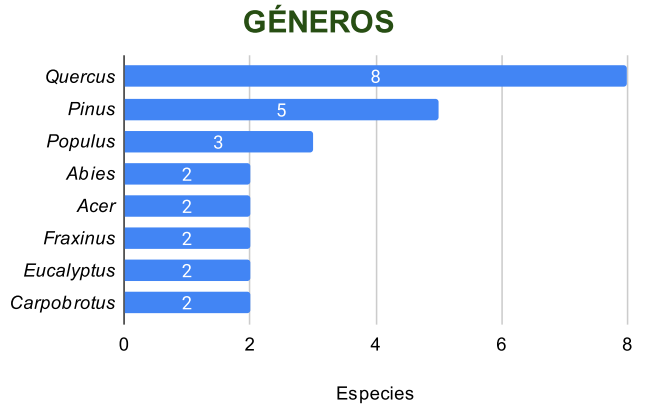


(a) Clases

(b) Órdenes más representados de los 21 totales



(c) Familias más representadas de las 30 totales



(d) Géneros más representados de los 40 totales

Figura 4.1: Resumen taxonómico de la BBDD: (a) 4 clases en total (*Pteridopsida* pertenece a *Azolla filiculoides*, añadida como sspp. invasora); (b) 21 órdenes; (c) 30 familias y (d) 40 géneros

Se han recogido 1793 nombres vernáculos, con 316 repeticiones para especies diferentes (por ejemplo, «roble» puede referirse a *Quercus faginea*, *Q. petraea*, *Q. pyrenaica*, *Q. robur* y *Q. rubra*). Los nombres vernáculos más repetidos son: «pino», «bellota», «roble», «carrasco», «chaparro»...(ver en el anexo B.3: Nombres vernáculos).

4.1.2. Requerimientos climáticos

El clima es el factor con mayor influencia sobre la distribución de las especies y condiciona el éxito o fracaso en el establecimiento de nuevas repoblaciones. Los parámetros controlados son:

1. **Altitud**: se trata realmente de un factor fisiográfico, pero ligado directamente a la climatología. Al aumentar la altitud, disminuye la temperatura y aumentan las radiaciones del espectro visible y ultravioleta, así como la precipitación. Se definen los rangos compatibles y óptimos para cada especie: por debajo de estos rangos se tendrían problemas de competencia con otras sspp.; por encima no estarían adaptadas a las bajas temperaturas. La Figura 4.2 muestra, para cada altitud, la frecuencia de especies con un rango que incluye tal altitud. El 50 % de las especies se ubican entre 450 y 1300 metros de altitud que, según Díaz-González (2015), se corresponde con los piso bioclimáticos colino y montano.

2. **Régimen pluviométrico**: se definen rangos óptimos y compatibles para la precipitación media anual (PMA) y el rango óptimo de precipitación estival (PV) para cada especie. Como se aprecia en la Figura 4.3, el 50 % de las especies tienen rangos compatibles entre 851 y 1551 mm para la PMA, correspondiente a los ombrotipos subhúmedo superior y húmedo, según Díaz-González (2015).

3. **Régimen térmico:** se definen los rangos óptimos y compatibles de:

   a) La temperatura media anual (TMA)

   b) La temperatura media del mes más frío (TMF)

   Quedan también definidos los RC de la temperatura media del mes más cálido (TMC) y la sensibilidad a heladas tardías (HT). Tal como muestra la Figura 4.4, el 50 % de las especies poseen rangos compatibles entre 8.4 y 13.79 °C.

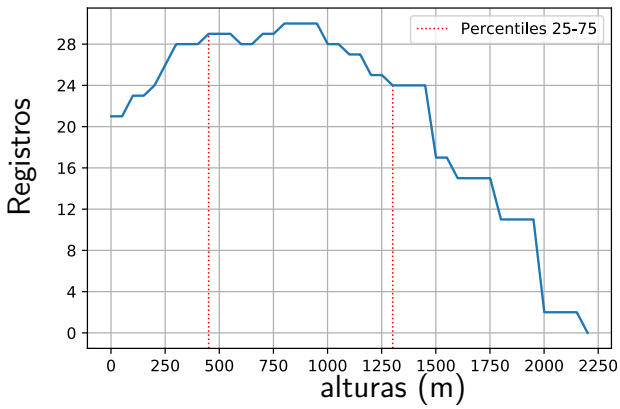


Figura 4.2: Distribución de alturas compatibles en la BBDD

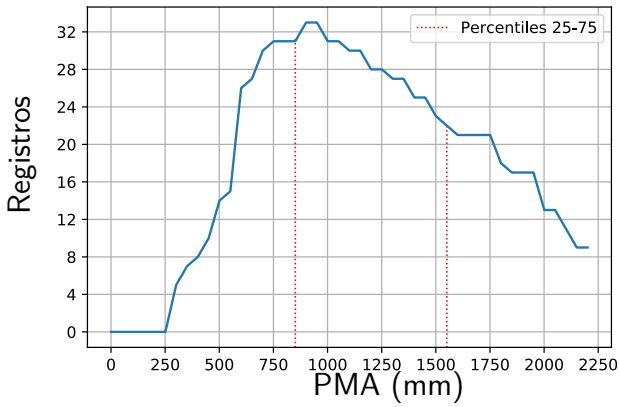


Figura 4.3: Distribución de PMA compatible en la BBDD

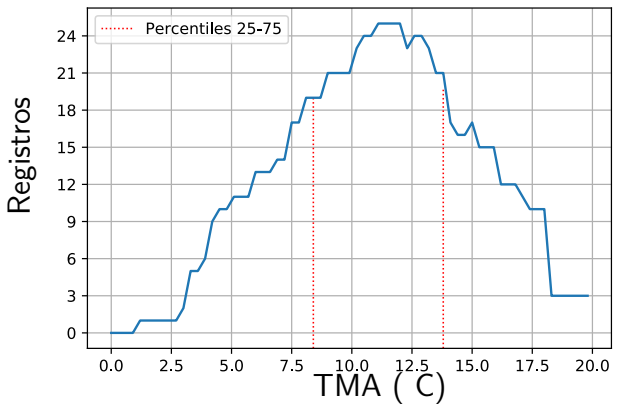


Figura 4.4: Distribución de TMA compatible en la BBDD

4.1.3. Requerimientos edafológicos

Se define la profundidad mínima del estrato (PM), los rangos compatibles de pH para cada especie y su sensibilidad a encharcamientos (SE). Respecto al pH, el 50 % de las especies están comprendidas en el rango 5.5 y 7.0 (ver Figura 4.5). La mediana está en 6.0, revelando una tendencia ligeramente acidófila de las especies en la BBDD. Respecto a encharcamientos, el 25.42 % presenta sensibilidad. La profundidad media ronda en torno a 30 cm.

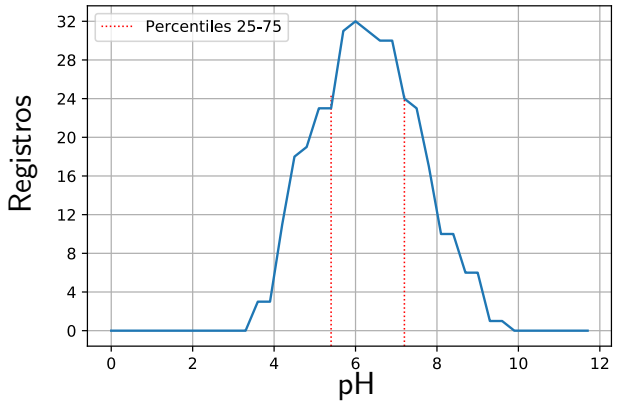


Figura 4.5: Distribución de pH compatible en la BBDD

4.1.4. Estado de conservación

Para cada especie, se define si tiene algún grado de amenaza según la *Lista Roja de Especies Amenazadas de la Unión Internacional para la Conservación de la Naturaleza* (UICN, 2021) y en el *Catálogo Regional de Flora Amenazada de Asturias* (AA, 2016). También, basándose en Prieto y col. (2014), se incluye su procedencia[3] y si se trata de una sspp. invasora o no.

En el caso de de la *Lista Roja de Especies Amenazadas de la UICN*, las categorías de amenaza son las siguientes:

- En peligro crítico (CR): la especie enfrenta un riesgo extremadamente alto de extinción en estado silvestre en el futuro inmediato. No se han recogido especies en esta categoría.
- En peligro (EN): se considera que se está enfrentando a un riesgo muy alto de extinción en estado silvestre. Se han recogido dos especies en esta categoría: *Pinus radiata* y *Abies pinsapo*.
- Vulnerable (VU): se considera que se está enfrentando a un riesgo alto de extinción en estado silvestre. No se han recogido especies en esta categoría.
- Casi amenazado (NT): cuando ha sido evaluado según los criterios y no satisface los criterios para las anteriores categorías; pero posiblemente los satisfaga en el futuro cercano. No se han recogido especies en esta categoría.
- Preocupación menor (LC): la especie no cumple ninguno de los criterios que definen las categorías anteriores. Se incluyen en esta categoría taxones abundantes y de amplia distribución. 33 de las 59 especies pertenecen a esta categoría.
- Datos insuficientes (DD) - dos especies: *Populus nigra* y *Opuntia ficus-indica*.

La protección de flora de Asturias está regulada por Decreto 65/1995, de 27 de abril, por el que se crea el Catálogo Regional de Especies Amenazadas de la Flora del Principado de Asturias con las siguientes categorías:

- «En peligro de extinción» (PE): su supervivencia es poco probable si los factores causales de su actual situación siguen actuando.

[3] Autóctona o alóctona

- «Sensibles a la alteración de su hábitat» (SAH): sus hábitats característicos están particularmente amenazados, en grave regresión, fraccionados o muy limitados.
- «Vulnerables» (VU): corren riesgo de pasar a las categorías anteriores en un futuro inmediato si los factores adversos que actúan sobre ellas no son corregidos.
- «De interés especial» (IE): poseen cierto valor científico, cultural o son singulares, sin pertenecer a ninguna otra categoría. Se han recogido ocho especies con esta categoría.

Se han incluido las 21 especies invasoras presentes en Asturias, las cuales están incluidas *Catálogo español de especies exóticas invasoras* (Real Decreto 630/2013).

5 | Sistema gestor de la base de datos

En este apartado se detalla con qué variables y funciones se ha desarrollado el SGBD, así como su flujo de procesos.

5.1. Variables y estructura de datos

A continuación, se enumeran todos los parámetros y sus variables en el SGBD, así como las principales funciones que emplean. Todas las tablas en esta sección están referidas a la especie *Castanea sativa*. Al final de la enumeración y descripción, la Tabla 5.12 sintetiza todas las variables/parámetros/atributos.

1. **Nombre científico**: valor de texto que **se usa como llave en los diccionarios** del resto de variables. Se asocia como identificador de la especie a cada variable. La función `get_cient()` genera una variable (`ncient`) que contiene un diccionario con todos los nombres científicos y su fila la BBDD:

   $$\texttt{ncient} = \{\text{Especie}_1 : \text{posición fila especie}_1, \ldots , \text{Especie}_n : \text{posición fila especie}_n\}$$

   Ocupa la primera columna de la BBDD. Se introducen en cada celda de forma literal (ver Tabla 5.1). Solo incluye la denominación binomial[1], existe otra variable `n_cientificos` generada mediante la función `sistematica()` que incluye los autores.

   Tabla 5.1: Ejemplo de nombre científico en la BBDD

Especie
Castanea sativa

2. **Nombres vernáculos**: valores de texto asociados a cada nombre científico. La información fue extraída de Anthos (2011). La función `get_nver()` genera una variable (`nver`) que contiene un diccionario, cuyos elementos son listas, compuestas a su vez de dos sublistas (a) y (b):

   $$\texttt{nver} = \{\text{Especie}_1 : [\ a_1, b_1\ ], \ldots , \text{Especie}_n : [\ a_n, b_n\ ]\}$$

   Donde:

   a = *N_Ver [ESP]*[2] — Nombres vernáculos en castellano
   b = *N_Ver [AST]* — Nombres vernáculos en asturiano

[1] Género y epíteto específico
[2] Nombres de los atributos en la BBDD.

Ocupan la segunda y tercera columna de la BBDD. Se introducen separados por '_' (ver Tabla 5.2).

Tabla 5.2: Ejemplo de nombres vernáculos en el la BBDD

N_Ver [ESP]	N_Ver [AST]
castaño_regoldo	*arizu_castañu*

Existe la posibilidad de manejar más nombres vernáculos en otros idiomas añadiendo columnas sin falta de modificar el código. Esto se traduciría en más sublistas (c, d, e...) adjuntas en la lista. Para ello, se añade una columna en cualquier posición con el nombre *N_ver [XXX]*, entre corchetes el identificador del idioma. Por ejemplo, si añadimos francés, el nombre de la columna podría ser *N_ver [FR]*.

3. **Cotas** o altitudes (m): valores numéricos que reflejan los rangos óptimos y compatibles de altitud de cada especie (RCO). La función `cotas()` genera dos variables: `cotas_optimo` y `cotas_umbral`. Cada una contiene un diccionario, cuyos elementos son listas que incluyen el valor inferior y superior del rango correspondiente, (a) - (b) para el óptimo; (c) - (d) para el compatible:

$$\texttt{cotas\_optimo} = \{\text{Especie}_1 : [\ a_1,\ b_1\ ],\ \ldots\ ,\ \text{Especie}_n : [\ a_n,\ b_n\ ]\}$$
$$\texttt{cotas\_umbral} = \{\text{Especie}_1 : [\ c_1,\ d_1\ ],\ \ldots\ ,\ \text{Especie}_n : [\ c_n,\ d_n\ ]\}$$

Donde:

$a =$ *op_cota_min* — Límite inferior de la altitud óptima
$b =$ *op_cota_max* — Límite superior de la altitud óptima
$c =$ *cota_min* — Límite inferior de la altitud compatible
$d =$ *cota_max* — Límite superior de la altitud compatible

Estos atributos ocupan cuatro columnas (ver Tabla 5.3) en la BBDD con posición móvil. Esto significa que, a diferencia de los anteriores atributos, cada columna puede estar en cualquier posición[3] **siempre y cuando se respete el nombre del atributo** en la cabecera. Por ejemplo, *cota_max* está en la columna 'L', pero podría intercambiar posición con cualquier otro atributo o moverse a la última columna sin afectar a la lectura del programa. Esto se logra con la función `get_posiciones()` y permite cierta flexibilidad a la hora de modificar o incluir nuevos parámetros en la BBDD. Esto se aplica también al resto de atributos continuación.

Tabla 5.3: Ejemplo de altitudes en la BBDD

cota_min	cota_max	op_cota_min	op_cota_max
0	1200	200	800

[3] A excepción de las 3 primeras columnas.

4. **Precipitaciones** (mm): valores numéricos que muestran el RCO de las precipitaciones medias anuales (PMA) y el RC de las precipitaciones estivales mínimas y máximas (PV). La función encargada de generar las variables (`precipitacion_umbral` y `precipitacion_optima` para PMA; `precipitacion_estival` para PV) es `precipitaciones()`. La estructura de los diccionarios es la siguiente:

$$\begin{aligned}\texttt{precipitacion\_optima} &= \{\text{Especie}_1 : [\ a_1,\ b_1\ ],\ \dots\ ,\ \text{Especie}_n : [\ a_n,\ b_n\ ]\}\\ \texttt{precipitacion\_umbral} &= \{\text{Especie}_1 : [\ c_1,\ d_1\ ],\ \dots\ ,\ \text{Especie}_n : [\ c_n,\ d_n\ ]\}\\ \texttt{precipitacion\_estival} &= \{\text{Especie}_1 : [\ e_1,\ f_1\ ],\ \dots\ ,\ \text{Especie}_n : [\ e_n,\ f_n\ ]\}\end{aligned}$$

   Donde:

$$\begin{aligned}a &= \textit{op\_PMA\_min} \text{ — Límite inferior de la PMA óptima}\\ b &= \textit{op\_PMA\_max} \text{ — Límite superior de la PMA óptima}\\ c &= \textit{PMA\_min} \text{ — Límite inferior de la PMA compatible}\\ d &= \textit{PMA\_max} \text{ — Límite superior de la PMA compatible}\\ e &= \textit{PV\_min} \text{ — Límite inferior de la PV compatible}\\ f &= \textit{PV\_max} \text{ — Límite superior de la PV compatible}\end{aligned}$$

   Los atributos ocupan seis columnas en la BBDD con posición móvil (ver Tabla 5.4).

Tabla 5.4: Ejemplo de precipitaciones en la BBDD

PMA_min	PMA_max	op_PMA_min	op_PMA_max	PV_min	PV_max
450	2000	600	1100	75	150

5. **Temperaturas** (°C): valores numéricos que reflejan los RCO de la temperatura media anual (TMA) y la temperatura media del mes más frío (TMF), RC para la temperatura media del mes más cálido (TMC) y Bin para la sensibilidad a heladas tardías (HT). La función `temperaturas()` genera las variables `tma_optimo`, `tma_umbral`, `tm_calido`, `tmf_umbral`, `tmf_optimo` y `sensible_ht`, cuyas estructuras son:

$$\begin{aligned}\texttt{tma\_optimo} &= \{\text{Especie}_1 : [\ a_1,\ b_1\ ],\ \dots\ ,\ \text{Especie}_n : [\ a_n,\ b_n\ ]\}\\ \texttt{tma\_umbral} &= \{\text{Especie}_1 : [\ c_1,\ d_1\ ],\ \dots\ ,\ \text{Especie}_n : [\ c_n,\ d_n\ ]\}\\ \texttt{tm\_calido} &= \{\text{Especie}_1 : [\ e_1,\ f_1\ ],\ \dots\ ,\ \text{Especie}_n : [\ e_n,\ f_n\ ]\}\\ \texttt{tmf\_umbral} &= \{\text{Especie}_1 : [\ g_1,\ h_1\ ],\ \dots\ ,\ \text{Especie}_n : [\ g_n,\ h_n\ ]\}\\ \texttt{tmf\_optimo} &= \{\text{Especie}_1 : [\ i_1,\ j_1\ ],\ \dots\ ,\ \text{Especie}_n : [\ i_n,\ j_n\ ]\}\\ \texttt{sensible\_ht} &= \{\text{Especie}_1 : k_1,\ \dots\ ,\ \text{Especie}_n : k_n\}\end{aligned}$$

   Donde:

$$\begin{aligned}a &= \textit{op\_TMA\_min} \text{ — Límite inferior de la PMA óptima}\\ b &= \textit{op\_TMA\_max} \text{ — Límite superior de la PMA óptima}\\ c &= \textit{TMA\_min} \text{ — Límite inferior de la PMA compatible}\\ d &= \textit{TMA\_max} \text{ — Límite superior de la PMA compatible}\\ e &= \textit{PV\_min} \text{ — Límite inferior de la PV compatible}\end{aligned}$$

f = *PV_max* — Límite superior de la PV compatible
g = *TMF_min* — Límite inferior de la TMF compatible
h = *TMF_max* — Límite superior de la TMF compatible
i = *op_TMF_min* — Límite superior de la TMF óptima
j = *op_TMF_max* — Límite superior de la TMF óptima
k = *Sen_Hel_T* — Valor binomial de HT

Sus atributos ocupan once columnas en la BBDD con posición móvil (ver Tabla 5.5)

Tabla 5.5: Ejemplo de temperaturas en la BBDD

TMA_min	TMA_max	op_TMA_min	op_TMA_max	TMC_min	TMC_max	TMF_min	TMF_max	op_TMF_min	op_TMF_max	Sen_Hel_T
6	14	9.8	13.4	15	24	1	5	1	5	si

6. **Sensibilidad a encharcamiento**: parámetro tipo Bin. Consta de una única variable generada mediante la función `encharcamiento()`. Ocupa una única columna en la BBDD con posición móvil (ver Tabla 5.6).

$$\texttt{sensible\_encharcamiento} = \{\text{Especie}_1 : a_1, \ldots, \text{Especie}_n : a_n\}$$

Donde:

a = *Sen_Encharcamiento* — Valor Bin de SE

Tabla 5.6: Ejemplo de sensibilidad a encharcamiento en la BBDD

Sen_Encharcamiento
si

7. **Profundidad mínima** (cm): parámetro tipo VNU, la función `profundidades()` genera la variable `profundidades`. Ocupa una única columna en la BBDD con posición móvil (ver Tabla 5.7).

$$\texttt{profundidades} = \{\text{Especie}_1 : a_1, \ldots, \text{Especie}_n : a_n\}$$

Donde:

a = *Prof_min* — VNU de SE

Tabla 5.7: Ejemplo de profundidad en la BBDD

Prof_min
60

8. **Ph**: se define el RC a través de la función `pH()` que genera la variable `pH`. Ocupa dos columnas en la BBDD con posición móvil (ver Tabla 5.8).

$$\texttt{pH} = \{\text{Especie}_1 : [\ a_1,\ b_1\ ],\ \ldots\ ,\ \text{Especie}_n : [\ a_n,\ b_n\ ]\}$$

Donde:

$a =$ *pH_min* — Límite inferior de la pH compatible
$b =$ *pH_max* — Límite superior del pH compatible

Tabla 5.8: Ejemplo de pH en la BBDD

pH_min	pH_max
4.5	6.5

9. **Procedencia** e **invasora**: el primero tipo texto, el segundo Bin. La función `libro_catalogo_cires()` genera dos variables: `invasora` y `procedencia`. Sus atributos ocupan dos columnas en la BBDD (ver Tabla 5.9). Se incluyen juntos porque los datos fueron extraídos de la misma fuente: *Catálogo de las plantas vasculares del Principado de Asturias*, de Prieto y col. (2014).

$$\begin{aligned}\texttt{invasora} &= \{\text{Especie}_1 : a_1,\ \ldots\ ,\ \text{Especie}_n : a_n\}\\ \texttt{procedencia} &= \{\text{Especie}_1 : b_1,\ \ldots\ ,\ \text{Especie}_n : b_n\}\end{aligned}$$

Donde:

$a =$ *invasora* — Valor Bin de «invasora»
$b =$ *procedencia* — Valor de texto de procedencia

Tabla 5.9: Ejemplo de procedencias e «invasoras» en la BBDD

procedencia	invasora
autoctona	no

10. **Amenazas**: valores de texto, se recogen tanto el nivel de amenaza en la *Lista Roja de Especies Amenazadas de la UICN* (UICN, 2021) y el *Catálogo Regional de Flora Amenazada de Asturias* AA, 2016. La función `amenazas()` genera las variables `uicn` y `catalogo_ast`. Sus atributos ocupan dos columnas en la BBDD con posición móvil.

$$\begin{aligned} \texttt{uicn} &= \{\text{Especie}_1 : a_1, \ldots, \text{Especie}_n : a_n\} \\ \texttt{catalogo\_ast} &= \{\text{Especie}_1 : b_1, \ldots, \text{Especie}_n : b_n\} \end{aligned}$$

Donde:

a = *uicn* — Valor de texto de la categoría de amenaza en la UICN
b = *catalogo_ast* — Valor de texto de categoría de amenaza en AA (2016)

Tabla 5.10: Ejemplo de amenazas en la BBDD

uicn	catalogo_ast
LC	no

11. **Sistemática**: valores de texto para el género, la especie, la clase, el orden y la familia; Bin para si es endémico o no. Todos los datos fueron extraídos de AFLIBER (Ramos y col., 2021). La función `sistematica()` genera las variables `n_cientificos`, `endemicos`, `generos`, `especies`, `clases`, `ordenes` y `familias`. Sus atributos ocupan siete columnas en la BBDD con posición móvil (ver Tabla 5.11).

$$\begin{aligned} \texttt{n\_cientificos} &= \{\text{Especie}_1 : a_1, \ldots, \text{Especie}_n : a_n\} \\ \texttt{endemicos} &= \{\text{Especie}_1 : b_1, \ldots, \text{Especie}_n : b_n\} \\ \texttt{generos} &= \{\text{Especie}_1 : c_1, \ldots, \text{Especie}_n : c_n\} \\ \texttt{especies} &= \{\text{Especie}_1 : d_1, \ldots, \text{Especie}_n : d_n\} \\ \texttt{clases} &= \{\text{Especie}_1 : e_1, \ldots, \text{Especie}_n : e_n\} \\ \texttt{ordenes} &= \{\text{Especie}_1 : f_1, \ldots, \text{Especie}_n : f_n\} \\ \texttt{familias} &= \{\text{Especie}_1 : g_1, \ldots, \text{Especie}_n : g_n\} \end{aligned}$$

Donde:

a = *Scientific_Name* — Valor de texto del nombre científico de la especie, incluyendo al autor
b = *Endemic* — Bin del endemismo
c = *Genus* — Valor de texto del género de la especie
d = *Species* — Valor de texto del epíteto específico de la especie
e = *Class* — Valor de texto de la clase de la especie
f = *Order* — Valor de texto del orden de la especie
g = *Family* — Valor de texto de la familia de la especie

Tabla 5.11: Ejemplo de nombre científico (con autor), endemismos, géneros, especies, clases, órdenes y familias en la BBDD

Scientific_Name	Endemic	Genus	Species	Class	Order	Family
Castanea sativa Mill.	FALSE	Castanea	sativa	Magnoliopsida	Fagales	Fagaceae

Tabla 5.12: Resumen de variables, parámetros y atributos en el SGBD. Se tienen un total de 23 parámetros, que se trasladan a 27 variables definidas a partir de 39 atributos. Los nombres vernáculos son un caso especial ya que, a pesar de esta definidos mediante una única cadena de texto en la BBDD, se almacenan en sublistas y no de forma directa como el resto de datos de de cadenas de texto

Parámetro		**Variable**	**Tipo**	**Atributos**
Sistemática	-	`ncient`	-	1
Nombres vernáculos	-	nver	-	2
Cotas	-	`cotas_optimo` cotas_umbral	RCO	4
Precipitaciones	PMA	`precipitacion_optima` precipitación_umbral	RCO	4
	PV	`precipitacion_estival`	RC	2
Temperaturas	TMA	`tma_optimo` `tma_umbral`	RCO	4
	TMF	`tmf_optimo` `tmf_umbral`	RCO	4
	TMC	`tm_calido`	RC	2
	HT	`sensible_ht`	Bin	1
Sensibilidad a encharcamiento	SE	`sensible_encharcamiento`	Bin	1
Profundidad mínima	PM	`profundidades`	VNU	1
pH	-	`pH`	RC	2
Procedencia	-	`procedencia`	Texto	1
Invasora	-	`invasora`	Texto	1
Amenazas	UICN	`uicn`	Texto	1
	cat_ast	`catalogo_ast`	Texto	1
Sistemática	NC	`n_cientificos`	Texto	1
	genero	`generos`	Texto	1
	especie	`especies`	Texto	1
	clase	`clases`	Texto	1
	orden	`ordenes`	Texto	1
	familia	`familias`	Texto	1
Endemico	-	`endemicos`	Bin	1
Total	***23***	***27***	-	***39***

Todas las variables que se acaban de describir son las que usa el SGBD para almacenar y trabajar con los datos. En realidad, existen *más variables*: aquellas que usa el programa para funcionar que no contienen datos, o variables intermedias entre la BBDD y las variables definitivas. Por ejemplo, la variable `header` contiene el encabezado de la BBDD en forma de lista, o la variable `csv`, que se usa de forma temporal para almacenar todos los datos de la BBDD en forma de «bloque caótico» antes de ser ordenados.

5.2. Funciones y flujo de procesos

En esta sección se describe brevemente el flujo de procesos en el SGBD. La Figura A.1 (Apéndice A, pág. 36) representa visualmente todo el proceso. En primer lugar, el SGBD importa toda la BBDD como un «bloque» mediante la función `import_csv()`. A continuación, divide el «bloque» para diferenciar el encabezado de los datos a partir de las funciones `get_list()` y `get_header()`, que generan las variables `l` (datos) y `header` (encabezado) respectivamente. Seguidamente, se generan las 27 variables descritas en el anterior apartado mediante sus respectivas funciones[4]. Estas funciones emplean a su vez la función `get_posiciones` para identificar en qué columnas están los atributos, lo cual permite que la BBDD tenga cualquier orden y que resulte sencillo añadir nuevos parámetros. Además, estas funciones encargadas de generar las variables también notifican errores en la BBDD: celdas vacías, valores incongruentes o con los que el SGBD no puede trabajar (cotas negativas, p. ej.). En los siguientes apartados se detallan las dos funciones principales que trabajan a partir de las variables: la **consulta** y la **búsqueda**.

5.2.1. Consulta

La consulta, principal uso del programa, es la encargada de determinar qué especies forestales son compatibles con los parámetros de la estación consultados. Lee el archivo de consultas y devuelve dos grupos de resultados (ver Figura 5.1):

1. Las especies compatibles, ordenadas según su idoneidad.
2. Las especies no compatibles, ordenadas de mayor a menor requerimientos compatibles con la consulta e indicando qué parámetros son incompatibles.

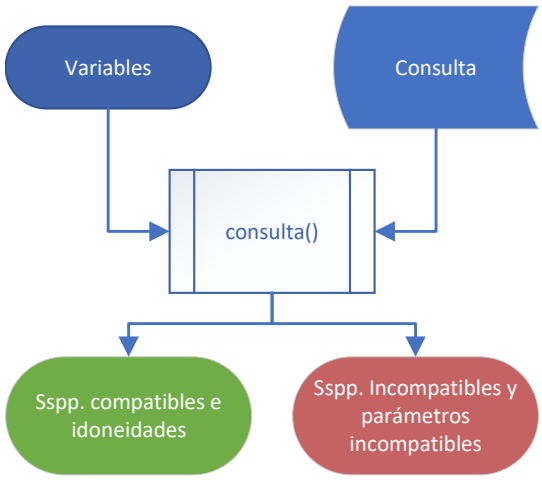


Figura 5.1: Flujo general de una consulta. La función consulta() lee el archivo de consulta, lo compara con los datos en las variables, devuelve especies compatibles ordenadas por idoneidad y las no compatibles, ordenadas de menor a mayor parámetros consultados incompatibles

Toda la consulta se lleva a cabo mediante una única función, `consulta()`, que, a partir de otras funciones secundarias, hace lo siguiente (ver Figura 5.2):

1. Leer el archivo de consulta y descartar todos los parámetros en blanco (no consultados) → función `vaciar_consulta()`.
2. Separar en dos grupos las especies en la BBDD: compatibles e incompatibles. Esto se consigue a partir de una serie de **funciones de comprobación** (FFC) que se encargan de comprobar si un valor dado está dentro o fuera de un RCO o RC, si cumplen un criterio Bin o si superan un valor mínimo VNU. En el caso de RCO, RC y VNU, suponen el cambio de un dato de entrada cuantitativo a uno de salida cualitativo (apto o no apto). Por ejemplo, la función `comprobar_cotas()` toma cualquier valor numérico de entrada y comprueba para cada especie si ese valor cae dentro o fuera del RCO de altitudes (definido por las varibales `cotas_umbral` y `cotas_optimo`), indicando si la especie es apta o no. Si todos los parámetros consultados son aptos para una especie, la especie se considera compatible.

[4] Ver en 5.1: Variables y estructura de datos

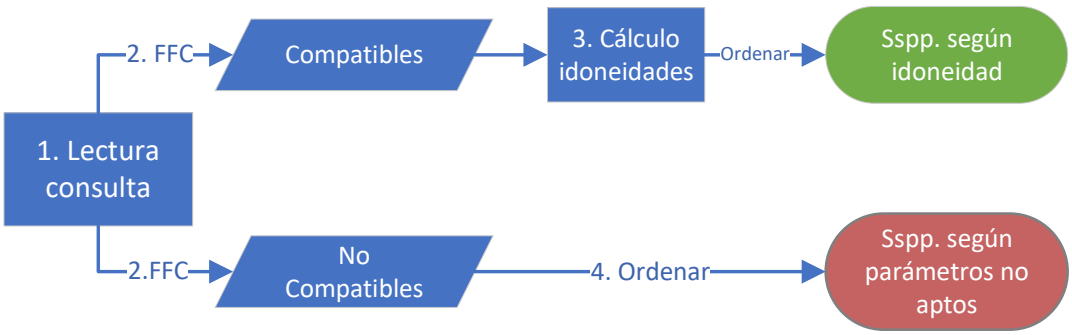


Figura 5.2: Flujo de procesos detallado de la consulta, equivalente a la Figura 5.1. En primer lugar, se realiza una lectura del archivo de la consulta. A continuación, a traves de las FFC, las especies forestales en la BBDD que contienen al menos 1 parámetro apto se separan en dos grupos : compatibles y no compatibles. Las compatibles se ordenan mediante idoneidad (3) y las no compatibles según el número de parámetros compatibles. Más adelante, la Figura 5.4 detalla el tercer paso

3. Diferenciar, dentro del grupo de las especies compatibles, cuáles presentan mayor aptitud. Esto se logra con las funciones `idoneidad_RCO()` e `idoneidad_RC()`, que evalúan cómo de óptimo es un valor para un RCO o RC. La **idoneidad** resultante para una especie es el promedio de todas las aptitudes individuales para cada parámetro consultado. Una vez determinadas sus idoneidades, se ordenan de mayor a menor. Más adelante se explica cómo funciona el proceso de evaluación exactamente.
4. Ordenar las especies no compatibles de mayor a menor número de parámetros consultados aptos. Se presentan especies con parámetros aptos y no aptos, indicando cuáles fallan.

Evaluación de idoneidad

El programa asigna una puntuación numérica entre 10 y 100 a las sspp. compatibles. Este resultado final es la media numérica para cada evaluación de cada parámetro. En caso de parámetros tipo Bin o VNU, la puntuación del parámetro es máxima directamente. Por ejemplo, si la profundidad mínima para una especie son 40 cm y se consulta un valor igual o por encima, tendrá la máxima puntuación. En cambio, para parámetros RCO y RC se calcula a través de dos funciones: `idoneidad_RCO()` e `idoneidad_RC()`.

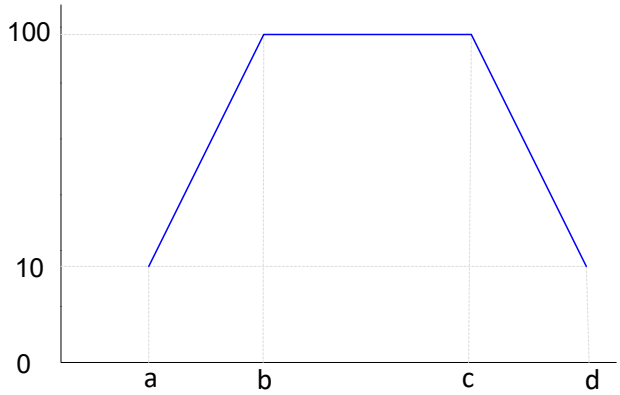


Figura 5.3: Interpolación idoneidades para RCO y RC. Los valores de la a la d equivalen a los límites compatibles (rango a-d) y óptimos (rango b-c). La idoneidad para el rango compatible [b,c] es 100; en los extremos compatibles (a,b)∪(c,d) se calcula mediante interpolación (ver ecuaciones (5.1) y (5.2)). Si el parámetro es RC, se genera un rango óptimo [b,c] como el 10 % central del rango [a,d] y se calcula como un RCO

La función `idoneidad_RCO()` toma de entrada los cuatro valores de RCO y el valor consultado. Si el valor consultado está dentro del rango óptimo, se devuelve la máxima puntuación (100). Si el valor está fuera del rango óptimo, es decir, entre alguno de los extremos inferior o superior compatible y los óptimos (en el rango [a,b) o (c,d] según la Figura 5.3), la función interpola, considerando que los valores extremos compatibles equivalen a una puntuación de

idoneidad de 10. Las ecuaciones (5.1) y (5.2) a continuación muestran las expresiones de interpolación para los rangos compatibles-no-óptimos superior e inferior respectivamente.

$$\frac{90 \cdot (v - a)}{b - a} + 10 \tag{5.1}$$

$$\frac{90 \cdot (d - v)}{d - c} + 10 \tag{5.2}$$

Donde:

$v =$ Valor consultado
$a =$ Límite inferior compatible
$b =$ Límite inferior óptimo
$c =$ Límite superior óptimo
$d =$ Límite superior compatible

Dado que un parámetro RC no posee rango óptimo, solamente compatible (rango [a,d] en la Figura 5.3), la función `idoneidad_RC()` considera un rango óptimo intermedio «imaginario» que ocupa un 10 % del rango compatible en su centro. Una vez creado ese rango, llama a la función `idoneidad_RCO()` indicando los 4 valores: rango [a,d] original y rango [b,c] «imaginario», además del valor consultado, de tal forma que `idoneidad_RCO()` tiene sus cinco valores de entrada para calcular la idoneidad.

Existe la posibilidad de que un registro en la BBDD no tenga los 4 atributos correspondientes a un parámetro RCO, faltando los asociados al rango óptimo. En este caso, la función `idoneidad_RCO()` lo detecta y recurre a `idoneidad_RC()`, indicando el rango compatible, de tal forma que `idoneidad_RCO()` le calcula el rango imaginario intermedio. La Figura 5.4 resume el todo el proceso para determinar la idoneidad de un parámetro para una especie compatible. Este proceso se repite para cada parámetro, almacenando el resultado en la memoria para efectuar un promedio después de la última evaluación.

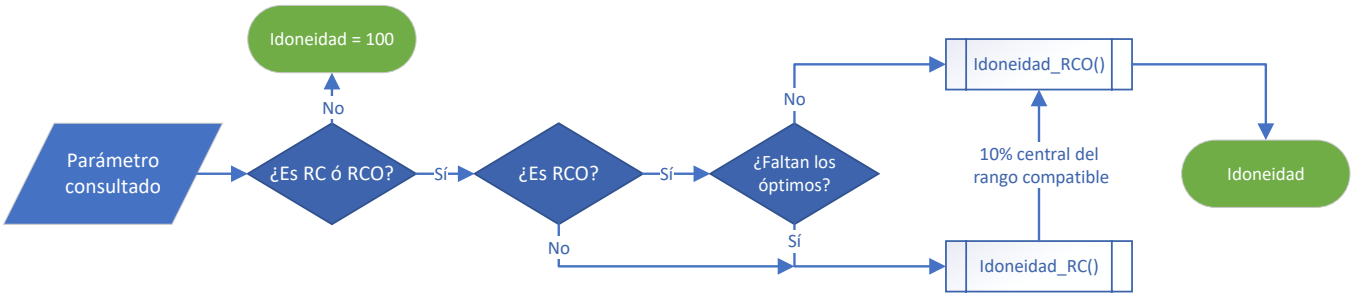


Figura 5.4: Determinación de idoneidad de un parámetro consultado mediante recursividad, equivalente al tercer paso en la Figura 5.2. La recursión está en el paso de determinar si faltan o no los óptimos, que se ejecuta dentro de la propia función «idoneidad_RCO()», que acaba llamándose a si misma a través de «idoneidad_RC()». La función «idoneidad_RCO()» recurre a ecuaciones (5.1) y (5.2) una vez que consigue todos los valores de entrada. Si por cualquier motivo cayera en un bucle infinito, devuelve «error» como resultado

5.2.2. Búsqueda

La función `buscar()` es un pequeño motor de búsqueda. Según el término ingresado devuelve un tipo de resultado. Mediante condicionales anidados y diccionarios, es capaz de entender los siguientes tipos de búsquedas:

1. Si se introduce un nombre científico (género y epíteto específico) devuelve toda la información del registro en la BBDD:
   a) Nombre científico con su autor.
   b) Nombres vernáculos en castellano y asturiano.
   c) Clase, orden y familia.
   d) Procedencia.
   e) Se indica si está presente en el *Catálogo Regional de Flora Amenazada de Asturias* (AA, 2016) o en la *Lista Roja de Especies Amenazadas de la UICN* (UICN, 2021).
   f) Se indica si es invasora o endémica.
   g) Todos sus requerimientos.
2. Si es un género, se indican todas las especies pertenecientes a tal género.
3. Si es un nombre vernáculo, se devuelven todas las especies que contienen tal nombre vernáculo.
4. Si es una categoría de la *Lista Roja de Especies Amenazadas* (UICN, 2021) o del Catálogo Regional de Flora Amenazada de Asturias (AA, 2016), se devuelven todas las especies pertenecientes a esa categoría.
5. Si se introduce «invasora», «invasoras» o «especies invasoras», devuelve todo el listado de especies invasoras. Ocurre de la misma forma para especies autóctonas y alóctonas.
6. Si se introduce la palabra «géneros», se devuelve un listado de todos los géneros incluidos en la BBDD. Lo mismo para las clases, órdenes, familias y especies.

Todos los resultados se ordenan alfabéticamente. Además, se ignoran tildes y mayúsculas: si, por ejemplo, se introduce la búsqueda «pLatanó», el programa entiende «plátano» y devuelve «*Acer pseudoplatanus*».

5.3. Código

Se estructura en dos bloques:

1. `Funciones`: código extenso que contiene todas las funciones del programa.
2. `Main`: pequeño código que importa el bloque `funciones` para ejecutar la consulta y/o la búsqueda.

El código completo se puede leer en los apartados C.1 y C.2 de los apéndices, a partir de la página 58.

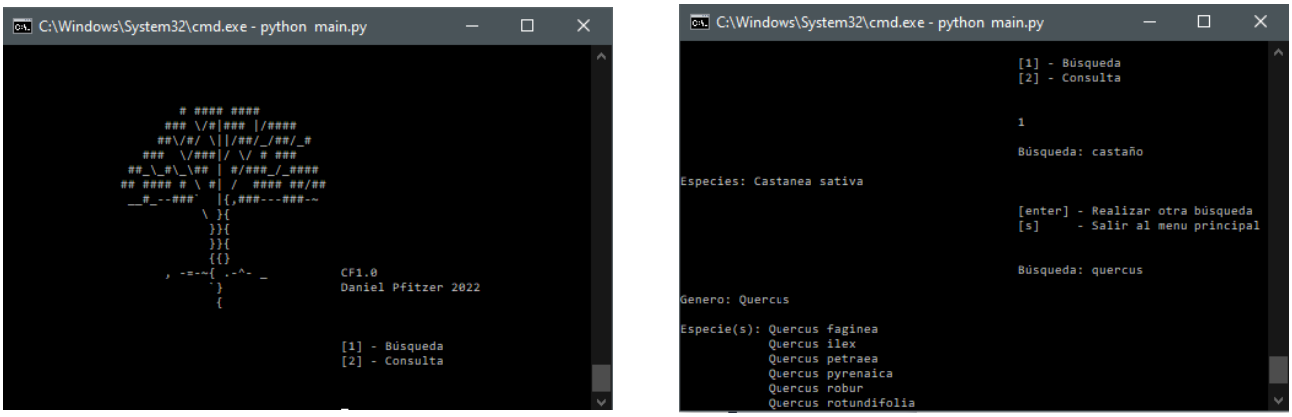
Parte II

Uso del programa

6 | Operaciones

6.1. Interfaz

El programa se usa a través del CMD de Windows, aunque también se puede emplear desde el propio intérprete de Python en la IDE (Spyder). Al ejecutar el programa, muestra una pequeña presentación con un menú (ver Figura 6.1a). También muestra una lista todos los errores encontrados en la BBDD. El usuario interactúa con el programa a partir de estos menús, ubicados siempre en la zona derecha, escribiendo la operación que desea realizar. El programa siempre devuelve los resultados en la parte izquierda, simulando una conversación. Después de cada operación, el programa pregunta cuál es la siguiente (ver Figura 6.1b).



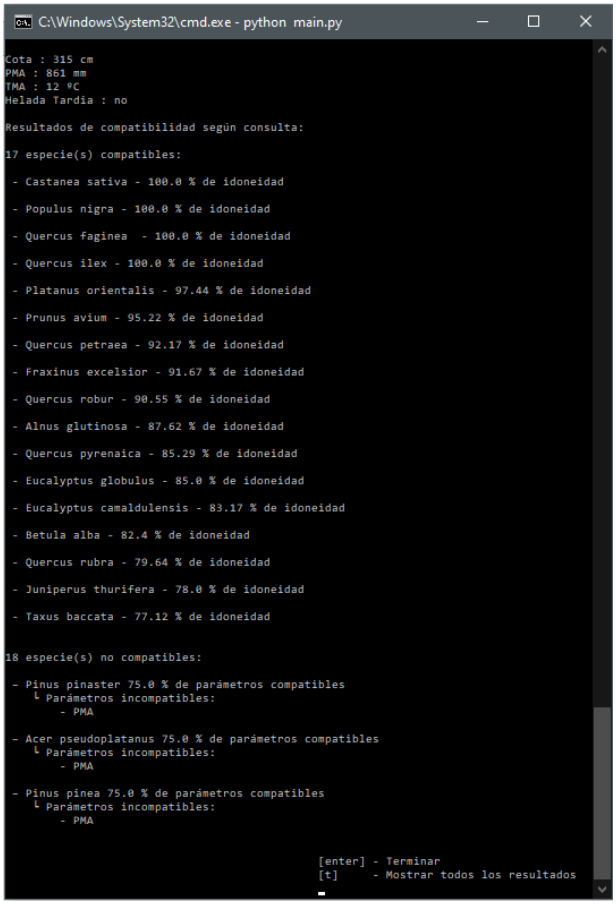
(a) Menu principal del programa

(b) Navegación por el menú

Figura 6.1: Interfaz del programa

6.2. Consulta

La consulta se realiza cumplimentado previamente el archivo `consulta.csv`, indicando qué parámetros se quieren examinar (ver Figura 6.2b). Si se dejan en blanco, el programa no los tiene en cuenta y no los considera como factores limitantes. Una vez realizada, primero devuelve los valores consultados, a continuación qué especies son compatibles, sus idoneidades y las especies no compatibles, indicando qué parámetros fallan (ver Figura 6.2a). Se puede cambiar el archivo CSV de consulta mientras el programa se ejecuta, de tal forma que se pueden realizar todas las consultas con parámetros distintos que se desee en una misma sesión. La limitación que presenta el sistema es la necesidad del usuario de conocer todos los parámetros para que el algoritmo afine lo más posible. Una solución interesante a este problema sería hacer que el programa detectara, en base a los rangos de los requerimientos de cada especie, si se trata de una especie generalista o especialista.



(a) Resultado de la consulta para (b)

Cota	315	m
PMA	861	mm
P Estival		mm
TMA	12	žC
TM (calido)		žC
TM (frío)		žC
Helada Tardia	no	si/no
Encharcamiento		si/no
Profundidad		cm
pH		$1_1 2$

(b) Parámetros consultados a través del archivo csv de consulta. El programa considera: las altitudes, PMA, PE y SE

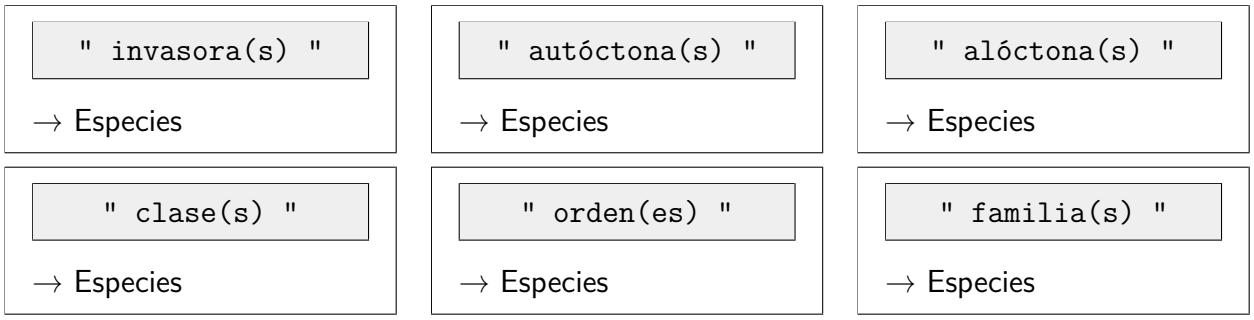
Figura 6.2: Ejemplo de consulta

6.3. Búsqueda

Se ingresa cualquier término[1], el programa «piensa» y devuelve un resultado. Por ejemplo, si introducimos «Quercus», identifica que es un género y devuelve todas las especies pertenecientes a ese género. En 5.2.2: Búsqueda se detallan sus posibilidades, pero se resumen a continuación:

Término ingresado	→ Información devuelta
N. científico	→ Información completa
Género	→ Especies
" Especie(s) "	→ Especies (todas)
N. vernáculo	→ Especies
Categoría UICN	→ Especies
Categoría CRFAA	→ Especies

[1] Sin importar tildes ni palabras en mayúscula o minúscula



En la Figura 6.1b se muestra un ejemplo de búsqueda.

7 | Validación del programa

Para comprobar si el programa funciona o no adecuadamente, se han llevado a cabo varias consultas simulando la calidad de estación en la Montaña Central Asturiana (MCA), donde se conoce detalladamente las principales especies forestales allí presentes. El experimento consiste en realizar una serie de consultas empleando los parámetros correspondientes las características climáticas y fisiográficas de la Montaña Central, esperando que las especies que el programa sugiera coincidan, por orden de compatibilidad e idoneidad, con las más distribuidas en la zona.

7.1. Datos de entrada, resultados esperados y metodología

La Tabla 7.1 muestra los parámetros introducidos al programa para simular la MCA. La Figura 7.1 detalla las especies forestales esperadas según la consulta, conocidas según Castaño-Santamara y col. (2013) y el Mapa Forestal de Asturias (MF25).

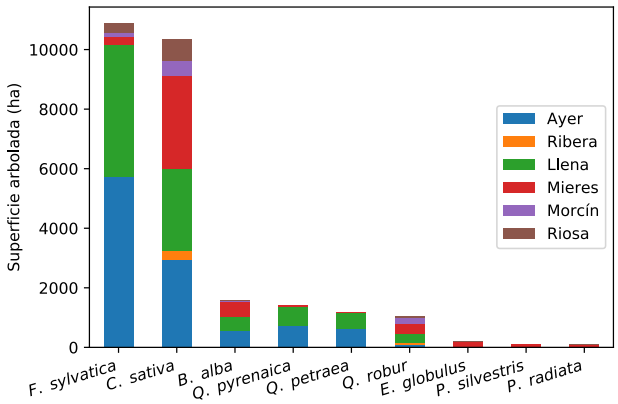


Figura 7.1: Especies forestales en la MCA, calculada a partir de las superficies arboladas en cada tesela del MF25 dentro del área de estudio

Dado que se tiene que consultar en un rango de alturas en vez de un solo valor, primero se consultan los parámetros a excepción de la altitud para comprobar qué especies compatibles devuelve el programa y qué idoneidades. Al no incluir la altitud, el programa considera que no es un factor limitante y que cualquier especie puede vegetar a cualquier altura, así que este primer resultado será solo una aproximación a partir de los otros cinco parámetros. A continuación, para cada altura posible en la MCA y de forma automatizada, se realiza una consulta manteniendo fijos el resto de parámetros. De 105 a 2405 metros de 5 en 5 metros, resultan un total de 460 consultas o simulaciones, donde el programa acumula las especies compatibles que va obteniendo. Se espera que las especies forestales más acumuladas en las simulaciones del programa coincidan con las que más ocupaciones presentan en el territorio (ver Figura 7.1).

Tabla 7.1: Parámetros de la MCA. Datos extraídos de Atlas climático de AEMET (precipitaciones y temperaturas) y del CNIG (Modelo Digital de Elevaciones)

PMA (mm)	PE (mm)	TMA (°C)	TMC (°C)	TMF (°C)	Cota mín.. (m)	Cota máx. (m)
861	136	8.3	14.7	2	105	2405

7.2. Resultados

Primera aproximación

La Figura 7.2 muestra los resultados de la consola del programa para la primera aproximación, donde no se tiene en cuenta la altitud. De estos resultados, 5 de las 6 especies propuestas coinciden con las reales. Esto son cinco especies de nueve reales acertadas, es decir, un 55 % de afinidad con un 83.33 % de acierto (5 de 6) para las sspp. propuestas. Además, el orden de idoneidades se corresponde con el orden de mayor a menor superficie ocupada por las especies forestales (a excepción del castaño (*Castanea sativa*) y del abedul (*Betula alba*).

¿Por qué sugiere el aliso (*Alnus glutinosa*)? El SGBD no cometió ningún error: los datos en la BBDD acerca de los requerimientos del aliso son correctos y compatibles con la estación de la MCA. Sin embargo, el SGBD desconoce un matiz importante acerca de esta especie: el aliso está ligado a cauces de agua, dado que sus raíces necesitan contacto casi permanente con el agua para sobrevivir (Bravo y Montero, 2008).

```
PMA : 861 mm
P Estival : 136 mm
TMA : 8.3 žC
TM (calido) : 17.7 žC
TM (frío) : 2 žC
Helada Tardia : no

Resultados de compatibilidad según consulta:

6 especie(s) compatibles:

 - Alnus glutinosa - 84.87 % de idoneidad

 - Fagus sylvatica - 82.45 % de idoneidad

 - Quercus pyrenaica - 81.55 % de idoneidad

 - Castanea sativa - 80.85 % de idoneidad

 - Betula alba - 79.23 % de idoneidad

 - Pinus sylvestris - 70.9 % de idoneidad
```

Figura 7.2: Primera aproximación: idoneidad de especies compatibles para la MCA sin tener en cuenta altitudes

Segunda aproximación: simulaciones en el rango de alturas

La Figura 7.2 muestra los resultados de la consola del programa para la segunda aproximación. Mantiene el porcentaje de afinidad y acierto, pero el número acumulado de especies compatibles a lo largo de las simulaciones no guarda correlación con las superficies de ocupación reales. Esto probablemente se debe a que las simulaciones no tienen en cuenta la frecuencia real de altitudes en el terreno. Se ha realizado una simulación desde la altura mínima hasta la máxima cada 5 metros, asumiendo que la orografía real sigue una progresión lineal perfecta.

```
   Fagus sylvatica 380 de 460 82.61 %
   Quercus pyrenaica 380 de 460 82.61 %
   Betula alba 380 de 460 82.61 %
   Alnus glutinosa 280 de 460 60.87 %
   Pinus sylvestris 241 de 460 52.39 %
   Castanea sativa 220 de 460 47.83 %
```

Figura 7.3: Segunda aproximación: resultados en el programa - idoneidad de especies compatibles para la MCA teniendo en cuenta el rango completo de altitudes (460 simulaciones)

7.3. Conclusiones

El programa **funciona**, pero con limitaciones. Un 83.33 % de acierto para las consultas es aceptable, demuestra poco margen de error y que las especies sugeridas son válidas. Sin embargo, el 55 % de afinidad evidencia que, a pesar de su validez, no son las *mejores* o las más aptas posibles. Esto está determinado por la calidad mejorable de la BBDD y por la falta de integración en un SIG. Un SIG, además de evitar la entrada manual de parámetros uno a uno desde el archivo externo de consultas, permitiría un análisis de de parámetros más preciso al conocer sus variaciones en el espacio; con la versión actual *solo* se pueden consultar zonas con parámetros homogéneos (rodales).

Que el programa desconozca matices trascendentes acerca de algunas especies[1] resulta una limitación únicamente cuando el usuario no realiza una interpretación crítica de los resultados. Tal como se describe en 2.1: Descripción y funcionalidades del programa, el programa está diseñado para asistir y optimizar el proceso proponiendo una lista preliminar. En ese sentido, cumple su función, pero hay que recordar que no aporta un resultado final perfectamente pulido, siendo necesario un último paso de interpretación de los resultados para transformarlos a información. De todas formas, sería ideal mejorar el programa, dotándolo de criterio propio válido para acercar su uso a un posible mayor número de usuarios.

Respecto al motor de búsqueda integrado, dado que se basa en principios puramente mecánicos de manejo de información (*querys*), funciona correctamente y es accesible para cualquier usuario. Que el buscador interprete cualquier termino ingresado y, en función de este, elija la información devuelta resulta cómodo e intuitivo.

[1] Como la dependencia del encharcamiento del aliso

Apéndices

A | Flujo de procesos de detalle

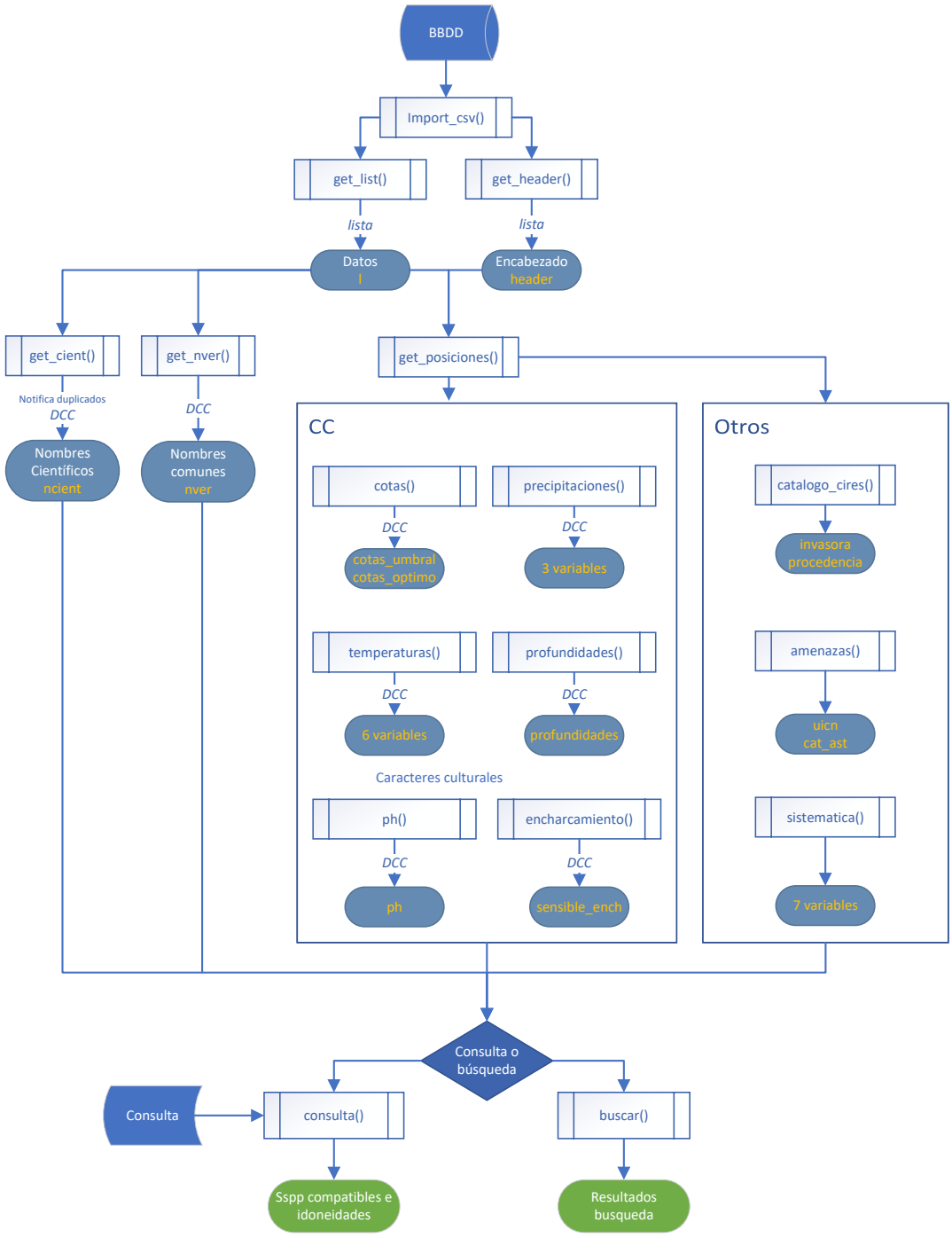


Figura A.1: Detalle del flujo de procesos en el SGBD, introducido en la sección 5.2: Funciones y flujo de procesos. La función de consulta se explica en 5.2.1: Consulta; búsqueda en 5.2.2: Búsqueda

B | Contenidos de la BBDD

B.1. Sistemática

A continuación se muestran taxones - número de especies presentes en la BBDD:

Clases

- *Liliopsida* - 5
- *Magnoliopsida* - 41
- *Pinopsida* - 11
- *Pteridopsida* - 1

Órdenes

- *Aquifoliales* - 1
- *Asparagales* - 1
- *Asterales* - 3
- *Caryophyllales* - 5
- *Commelinales* - 3
- *Cupressales* - 2
- *Cyperales* - 1
- *Fabales* - 2
- *Fagales* - 13
- *Gentianales* - 1
- *Lamiales* - 3
- *Malpighiales* - 4
- *Myrtales* - 2
- *Oxalidales* - 1
- *Pinales* - 9
- *Proteales* - 1
- *Rosales* - 1
- *Salviniales* - 1
- *Sapindales* - 3
- *Sonalaes* - 1
- *Zingiberales* - 1

Familias

- *Aizoaceae* - 2
- *Apocynaceae* - 1
- *Aquifoliaceae* - 1
- *Asparagaceae* - 1
- *Asteraceae* - 3
- *Azollaceae* - 1
- *Betulaceae* - 2
- *Cactaceae* - 1
- *Commelinaceae* - 1
- *Convolvulaceae* - 1
- *Cupressaceae* - 1
- *Fabaceae* - 1
- *Fagaceae* - 10
- *Juglandaceae* - 1
- *Mimosoideae* - 1
- *Myrtoideae* - 2
- *Oleaceae* - 2
- *Oxalidaceae* - 1
- *Pinaceae* - 9
- *Platanaceae* - 1
- *Poaceae* - 2
- *Polygonaceae* - 2
- *Pontederiaceae* - 1
- *Rosaceae* - 1
- *Salicaceae* - 4
- *Sapindaceae* - 2
- *Scrophulariaceae* - 1
- *Simaroubaceae* - 1
- *Taxaceae* - 1
- *Zingiberaceae* - 1

Géneros

- *Abies* - 2
- *Acacia* - 1
- *Acer* - 2
- *Agave* - 1
- *Ailantus* - 1
- *Alnus* - 1
- *Ambrosia* - 1
- *Araujia* - 1
- *Azola* - 1
- *Baccharis* - 1

- *Betula* - 1
- *Buddleja* - 1
- *Carpobrotus* - 2
- *Castanea* - 1
- *Cortaderia* - 1
- *Eichhornia* - 1
- *Eucalyptus* - 2
- *Fagus* - 1
- *Fallopia* - 2
- *Fraxinus* - 2
- *Hedychium* - 1
- *Ilex* - 1
- *Ipomoea* - 1
- *Juglans* - 1
- *Juniperus* - 1
- *Larix* - 1
- *Opuntia* - 1
- *Oxalis* - 1
- *Pinus* - 5
- *Platanus* - 1
- *Populus* - 3
- *Prunus* - 1
- *Pseudotsuga* - 1
- *Quercus* - 8
- *Salix* - 1
- *Senecio* - 1
- *Spartina* - 1
- *Taxus* - 1
- *Tradescantia* - 1
- *Ulex* - 1

Especies

- *Abies alba*
- *Abies pinsapo*
- *Acacia dealbata*
- *Acer campestre*
- *Acer pseudoplatanus*
- *Agave americana*
- *Ailanthus altissima*
- *Alnus glutinosa*
- *Ambrosia artemisiifolia*
- *Araujia sericifera*
- *Azolla filiculoides*
- *Baccharis halimifolia*
- *Betula alba*
- *Buddleja davidii*
- *Carpobrotus acinaciformis*
- *Carpobrotus edulis*
- *Castanea sativa*
- *Cortaderia selloana*
- *Eichhornia crassipes*
- *Eucalyptus camaldulensis*
- *Eucalyptus globulus*
- *Fagus sylvatica*
- *Fallopia baldschuanica*
- *Fallopia japonica*
- *Fraxinus angutifolia*
- *Fraxinus excelsior*
- *Hedychium gardnerianum*
- *Ilex aquifolium*
- *Ipomoea indica*
- *Juglans regia*
- *Juniperus thurifera*
- *Larix deciudua*
- *Opuntia ficus-indica*
- *Oxalis pes-caprae*
- *Pinus nigra*
- *Pinus pinaster*
- *Pinus pinea*
- *Pinus radiata*
- *Pinus sylvestris*
- *Platanus orientalis*
- *Populus alba*
- *Populus nigra*
- *Populus tremula*
- *Prunus avium*
- *Pseudotsuga menziessii*
- *Quercus faginea*
- *Quercus ilex*
- *Quercus petraea*
- *Quercus pyrenaica*
- *Quercus robur*
- *Quercus rotundifolia*
- *Quercus rubra*
- *Quercus suber*
- *Salix salviifolia*
- *Senecio inaequidens*
- *Spartina patens*
- *Taxus baccata*
- *Tradescantia fluminensis*
- *Ulex europaeus*

B.2. Frecuencias de los parámetros (CC)

Las figuras B.1 a B.6 muestran frecuencias de cada parámetro en el bloque de Requerimientos climáticos (sección 4.1.2). En caso de parámetros RCO, se incluyen ambas curvas en el mismo gráfico (compatible-óptimo). Se aprecia que las curvas óptimas guardan correlación con la forma de las compatibles, con menor frecuencia. Para los parámetros RC se han añadido los cuartiles Q_1, Q_2 y Q_3.

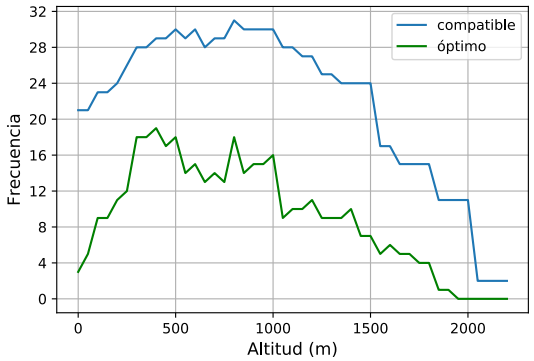


Figura B.1: Frecuencias de altitudes (RCO) en la BBDD

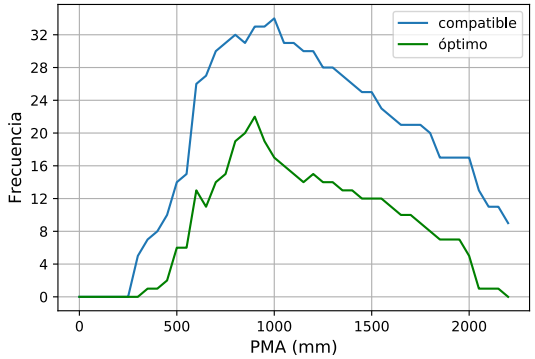


Figura B.2: Frecuencias de PMA (RCO) en la BBDD

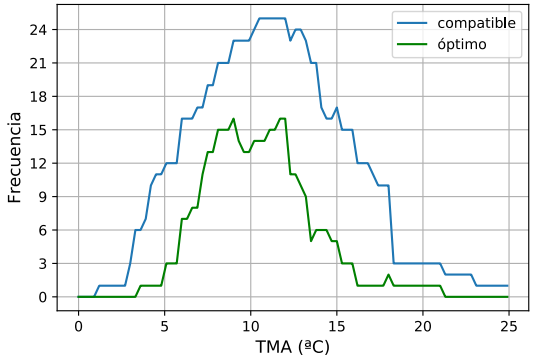


Figura B.3: Frecuencias de TMA (RCO) en la BBDD

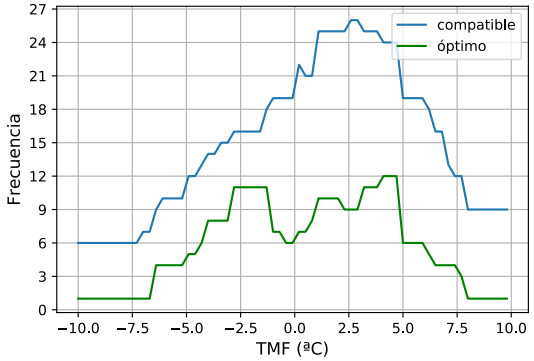


Figura B.4: Frecuencias de TMF (RCO) en la BBDD

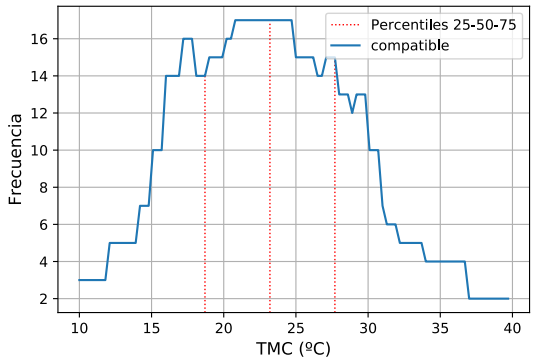


Figura B.5: Frecuencias de TMC (RC) en la BBDD

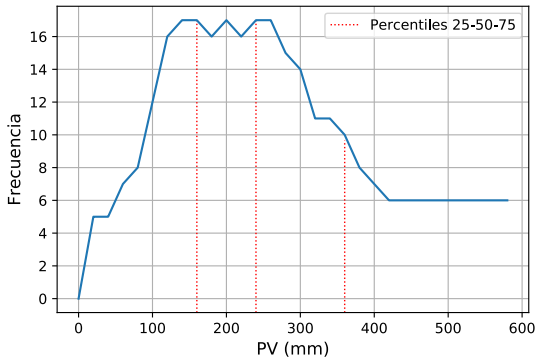


Figura B.6: Frecuencias de PV (RC) en la BBDD

B.3. Nombres vernáculos

A continuación se muestran los 1793 nombres vernáculos, recogidos de Anthos (2011) para cada especie.

Abies alba

- Español:
  - Abete
  - Abeto
  - Abeto blanco
  - Abeto común
  - Abeto de escocia
  - Abeto de normandía
  - Abeto de hoja de tejo
  - Abeto de hojas de tejo
  - Abeto noble
  - Abeto pectinado
  - Abeto plateado
  - Abeto-branco
  - Abetuna
  - Abetunas
  - Picea
  - Pinabete
  - Pinabete común
  - Pinavete
  - Pino-abeto
  - Sapino
  - Árbol de navidad altoaragonés
  - Abet
  - Abete
  - Abetina
  - Abeto

Abies pinsapo

- Español:
  - Abeto de andalucía
  - Abeto de españa
  - Abeto moro
  - Abeto noble de españa
  - Carajuelos piña
  - Pinabete de andalucía
  - Pino
  - Pino de grazalema
  - Pino pinsapo
  - Pino-pinsapo
  - Pinsapo
  - Pinzapo

Acacia dealbata

- Español:
  - Acacia
  - Acacia de hoja azul
  - Acacia francesa
  - Alcacia
  - Alcarcia
  - Mimal
  - Mimosa
- Asturiano:
  - Mimosa
  - Mimosal

Acer campestre

- Español:
  - Acer
  - Acere
  - Acere blando
  - Acero
  - Acerón
  - Aciron
  - Acirón
  - Amapolo
  - Arce
  - Arce común
  - Arce menor
  - Arce moscón
  - Arce-quejigo
  - Arcero
  - Aviones
  - Avión
  - Azcarro
  - Azre
  - Erable
  - Escarro
  - Escarrón
  - Gavilán
  - Malbillo
  - Marialbillo
  - Moscon
  - Moscón
  - Moscón común
  - Quejigo-arce
  - Rompe-caldera
  - Rompecaldera
  - Sacere
  - Samapol
  - Samapul
  - Volandero
  - Ácere
  - Ácere blando
- Asturiano:
  - Plàganu

Acer pseudoplatanus

- Español:
  - Acer
  - Acirón
  - Arce
  - Arce blanco
  - Arce blanco de otros
  - Arce menor
  - Arracadero
  - Asar
  - Azar
  - Blada
  - Falso plátano
  - Falso sicómoro
  - Falso-plátano
  - Moscon
  - Moscón
  - Orón
  - Platanero
  - Pládano
  - Plágano
  - Plánago
  - Plátano
  - Silvestre
  - Prádana
  - Prádano
  - Samapol
  - Sicomoro
  - Sicomoro de algunos
  - Sicómoro
- Asturiano:
  - Arce
  - Plagano
  - Plágamu
  - Plágano
  - Pláganu
  - Prádana
  - Texón
  - Teyón

Agave americana

- Español:
  - Acíbara
  - Agave
  - Alcibara
  - Alcibarón
  - Alcibera
  - Alcimara
  - Alcimarón
  - Aloe
  - Aloe americana
  - Aloe americano
  - Alsabara
  - Alzabara
  - Alzavara
  - Arzabara
  - Arzabarón
  - Atzahara
  - Atzavara
  - Azabara
  - Azabarón
  - Cabuyá
  - Cardón
  - Cimbara
  - Donarda
  - Figarasa
  - Javila
  - Maguey
  - Magüey
  - Metl
  - Pita
  - Pita común
  - Pitaca
  - Pitaco
  - Pitera
  - Pitón
  - Rafia
  - Rapia
  - Sábila
  - Zabila
  - Zabilla
  - Zábila
  - Ágave
- Asturiano:
  - Cáctus

Ailanthus altissima

- Español:
  - Ailanto
  - Árbol del cielo
  - Malhuele

Alnus glutinosa

- Español:
  - Haumero
  - Homero
  - Abedul
  - Aleso
  - Alisa
  - Alisio
  - Aliso
  - Aliso común
  - Aliso negro
  - Almiero
  - Alno
  - Alyso
  - Amieira
  - Amieiro
  - Amiero
  - Avellano moral
  - Bernazo
  - Berneazo
  - Chopo mulato
  - Chopo rizado
  - Chopo verde
  - Homeiro
  - Homero
  - Humeiro
  - Humero
  - Leña floja
  - Omeiro
  - Omero
  - Panblando
  - Pantierno
  - Ramajeira
  - Ramajera
  - Ramajero
  - Rameira
  - Umeiro
  - Umeiru
  - Umero

- Umeru
- Vinagrera altoaragonés: alberniz
- Aliso
- Alno
- Auno
- Bern
- Aumero
- Omero

- Asturiano:
  - Alisa
  - Aliso
  - Alisu
  - Humeiru
  - Humero
  - Humeru
  - Humiro
  - Llumeru
  - Omero
  - Omeru
  - Oumeiru
  - Umeiru
  - Umero
  - Umeru

Ambrosia artemisiifolia

Araujia sericifera

- Español:
  - Arauja

Azolla filiculoides

- Español:
  - Helecho de agua
  - Helecho flotante
  - Doradila
  - Yerba del agua

Baccharis halimifolia

- Español:
  - Chiclca
  - Carqueja

Betula alba

- Español:
  - Abedul
  - Abedul blanco
  - Abedul común
  - Abedul péndulo
  - Abedul verrucoso
  - Albar
  - Aliso blanco
  - Beduch
  - Bedul
  - Bidueiro
  - Bierzo
  - Bieso
  - Biezo
  - Bédul
  - Chopa blanca
  - Chopo silvestre

Buddleja davidii

- Español:
  - Arbusto de las mariposas
  - Budleya
  - Lila de verano
  - Lilar

Carpobrotus acinaciformis

- Español:
  - Flor del cuchillo

Carpobrotus edulis

- Español:
  - Bálsamo
  - Carpobrotus
  - Cuchilla
  - Flor del cuchillo
  - Flores del cuchillo
  - Gázules
  - Hierba del cuchillo
  - Uña de gato

Castanea sativa

- Español:
  - Cahtañera
  - Cahtañero
  - Cahtaño
  - Cahtañu
  - Castaña
  - Castañal
  - Castañal bravo
  - Castañar
  - Castañas
  - Castañas fruto
  - Castañas bravas
  - Castañeda
  - Castañeiro
  - Castañera
  - Castañero
  - Castaño
  - Castaño
  - Castaño bravo
  - Castaño bravío
  - Castaño común
  - Castaño escajo
  - Castaño injerto
  - Castaño inxerto
  - Castaño macho
  - Castaño molar
  - Castaño rapón
  - Castaño real

- Castaño regoldano
- Castaño regoldo
- Castaño regueldo
- Castaño salvaje
- Castaño silvestre
- Castañu
- Concha
- Concho
- Engertos
- Erizo
- Machial
- Mayo
- Mayuca
- Mayuelo
- Molso
- Morso
- Rebolda-nos comu-nes
- Reboldos
- Regoldana
- Regoldano
- Regolda-nos
- Regoldo
- Regoldo-nas
- Regoldos
- Sanmarti-niegos
- Tagarnizos
- Valdunas
- Verdeja
- Verdeña
- Asturiano:
  - Aricio
  - Arizu
  - Castaneiru
  - Castaña
  - Castañal
  - Castañar
  - Castañar montesina
  - Castañeiro
  - Castañero
  - Castañeu
  - Castaño
  - Castañu
  - Castañu montesín
  - Castañu montés

Cortaderia selloana

- Español:
  - Carrizo de las pampas
  - Carrizos de las pampas
  - Flor del plumero
  - Hierba de la pampa
  - Hierba de las pampas
  - Plumero
  - Plumeros
- Asturiano:
  - Plumeros

Eichhornia crassipes

- Español:
  - Jacinto de agua
  - Lirio acuá-tico
  - Flor de bo-ra
  - Buchón de agua
  - Camalote
  - Aguapey
  - Lechuguín tarope
  - Tarulla

Eucalyptus camaldulensis

- Español:
  - Alcolitos
  - Arcolitos
  - Arlcolito
  - Calipto
  - Caliptro
  - Calisto
  - Calistro
  - Calixto
  - Calixto hembra
  - Calixto macho
  - Carlisto
  - Ecalisto
  - Eucalipto
  - Eucalipto colorado
  - Eucalipto colorao
  - Eucalipto hembra
  - Eucalipto macho
  - Eucalipto medicinal
  - Eucalipto negro
  - Eucalipto rojo
  - Eucalipto rostrado
  - Eucaliptro
  - Eucaliptus
  - Eucalisto
  - Eucalistro
  - Eucalito
  - Ocalipto
  - Ocalisto
  - Ocalistro
  - Ocalito
  - Ocalixto
  - Ocálito
  - Ogalito
  - Quinino
  - Quino
  - Ucalito
  - Ucálito

Eucalyptus globulus

- Español:
  - Calipes
  - Calipse
  - Calipto
  - Calipto blanco
  - Calisto
  - Calisto blanco
  - Calistro
  - Calitro
  - Carlisto

- Encalistro
- Eucalihto
- Eucalipto
- Eucalipto azul
- Eucalipto blanco
- Eucalipto glóbulo
- Eucaliptu
- Eucaliptus
- Eucalito
- Eucálito
- Garlito
- Gomero azulado
- L-ucalipto
- Nogalito
- Ocalipto
- Ocalito
- Ocálito
- Quinino
- Ucalihto
- Ucalito
- Ucalitu
- Ucaritu
- Ucálito

- Asturiano:
  - Eucalitu
  - Ocalito
  - Ocalitu
  - Ocálitu
  - Secalitu

Fagus sylvatica

- Español:
  - Aguases
  - Ayolín
  - Ayorna
  - Ayorno
  - Ayurna
  - Ayurno
  - Bacua
  - Caxiga
  - Chaparro
  - Chara
  - Characo
  - Chirpia
  - Coscojo
  - Fabeta
  - Fabuco
  - Fabucos fruto
  - Fago
  - Fau
  - Fava
  - Faya
  - Fayal
  - Fayedal
  - Fayones
  - Fayuco
  - Fue
  - Gabaces
  - Grana
  - Hagüey
  - Hai
  - Haiga
  - Hay
  - Haya
  - Haya
  - Haya roja
  - Hayorna
  - Hayorno
  - Hayuca
  - Hayuco
  - Hayucus
  - Jaya
  - Jayuco
  - Matorra
  - Ove
  - Pago
  - Pasto
  - Pasto de haya fruto
  - Regoldo
  - Urnija
  - Urniju
  - Zacarda
  - Zaya

- Asturiano:
  - Fauco
  - Faya
  - Fayeu
  - Fayuco
  - Habucu
  - Hae
  - Haya
  - Haúcu
  - Haya
  - Hayu
  - Hayucu
  - Jaya
  - Jayuco

Fallopia baldschuanica

Fallopia japonica

- Español:
  - Hierba nudosa japonesa

Fraxinus angutifolia

- Español:
  - Fleja
  - Fragino
  - Freno
  - Fresnera
  - Fresno
  - Fresno común
  - Fresno de castilla
  - Fresno de hoja estrecha
  - Fresno de hoja pequeña
  - Fresno de la tierra
  - Fresno silvestre
  - Friznu
  - Frédenu
  - Lubios
  - Lupas
  - Lupias
  - Ramón
  - Soto
  - Ujulobu
  - Varas de san blas

- Asturiano:
  - Fleino
  - Fleisno

- Freisnu
- Frenu
- Frisnu

Fraxinus excelsior

- Español:
  - Fleja
  - Fragino
  - Frejú
  - Fresno
  - Fresno común
  - Fresno común verdadero
  - Fresno de vizcaya
  - Fresno de hoja ancha
  - Fresno elevado
  - Fresno europeo
  - Fresno norteño
  - Frexno
  - Frexo
  - Frágino
- Asturiano:
  - Freisnu
  - Fresna
  - Fresniu
  - Fresno
  - Fresnu

Hedychium gardnerianum

- Español:
  - Jengibre hawaiano
  - Kahili

Ilex aquifolium

- Español:
  - Aceba
  - Acebeo
  - Acebiño
  - Acebo
  - Acebo hembra
  - Acebo mallorquín
  - Acebo que lleva como cerezas
  - Acebo-cerezo
  - Acebro
  - Acebu
  - Aceo
  - Acibu
  - Acibuche
  - Adebo
  - Agrifolio
  - Alcebo
  - Alebro
  - Aquifolia
  - Aquifolio
  - Arcebo
  - Azabuche
  - Bollitera
  - Bolostios
  - Cardo blanco
  - Cardon
  - Cardonera
  - Cardón
  - Carrasca
  - Carrasco
  - Carrascu
  - Cebor
  - Cebro
  - Cebro que suelen llamar cedro
  - Chaparro
  - Colostios
  - Crebol
  - Crébol
  - Crévol
  - Escardamudos
  - Grebolé
  - Grévole
  - Hereu
  - Muérdago
  - Sardón
  - Xardón
- Asturiano:
  - Acea
  - Aceba
  - Acebo
  - Acebro
  - Acebu
  - Acebuche
  - Acegu
  - Aceo
  - Aceu
  - Acibu
  - Acéu
  - Briscu
  - Carrasco
  - Carrascu
  - Carrescu
  - Cébranu
  - Escayu
  - Espino
  - Espinu
  - Sardón
  - Xardona
  - Xardonera
  - Xardón

Ipomoea indica

- Español:
  - Campanita morada
  - Batatilla de indias
  - Maravilla

Juglans regia

- Español:
  - Anoales
  - Anogal
  - Carriona
  - Cascarol
  - Cascarón
  - Coca
  - Cocón
  - Conchais
  - Conchal
  - Concheiro
  - Conchiro
  - Concho
  - Conjo
  - Corcón
  - Coscol

- Cuca
- Cucal
- Cuco
- Cuculina
- Cucón
- Cusial
- Cáscara
- La nogal
- La nogal
- La nogal blanca
- La nogal negra
- Matamadres
- Noal
- Nocedo
- Noceo
- Noces
- Noga
- Nogal
- Nogal blanca
- Nogal común
- Nogal español
- Nogal europeo
- Nogal negra
- Nogal picado
- Nogal real
- Nogal rinconero
- Nogala
- Nogar
- Nogueira
- Nogueiro
- Noguera
- Noguera rinconera
- Noguero
- Nozal
- Nuecero
- Nueces
- Nueces fruto
- Nueces machos
- Nueces mollares
- Nueces ocales
- Nueces rinconeras
- Nuez
- Nuez encarcelá
- Nuezal
- Nugal
- Nugueiro
- Nuguera
- Nuquera
- Nuzal
- Nuzeiro
- Pajarera
- Ñogal
- Ñozal
- Ñuez

- Asturiano:
  - Conchal
  - Concho
  - Conchu
  - Cunchal
  - Cunsial
  - Muergu
  - Noceira
  - Noceiro
  - Noceo
  - Noceu
  - Nogal
  - Nogalera
  - Nozal
  - Nuaz
  - Nuceiro
  - Nuecero
  - Nuez
  - Nuez habanera
  - Nuoz
  - Perote
  - Zozal
  - Ñozal
  - Ñuez

Juniperus thurifera

- Español:
  - Agallaras
  - Agalluja
  - Agallujas
  - Agayuga
  - Agayujas
  - Albarra
  - Alborra
  - Cagurrias
  - Cedro
  - Cedro de españa
  - Cedro de españa con fruto negro
  - Enebra
  - Enebro
  - Enebro de incienso
  - Enebro de la bardera
  - Gallaritas
  - Gayuba
  - Sabina
  - Sabina albal
  - Sabina albar
  - Sabina albarra
  - Sabina blanca
  - Sabina de españa
  - Sabina española
  - Sabina negral
  - Sabina real
  - Sabina roma
  - Sabina romana
  - Sabina trabina
  - Sabina turífera
  - Sabino
  - Tejo
  - Trabina
  - Trabino

- Asturiano:
  - Cairuetu

Larix deciudua

- Español:
  - Alerce
  - Alerce europeo
  - Lárice

Opuntia ficus-indica

- Español:
  - Nopal
  - Higuera de pala
  - Palera
  - Penca
  - Tuna
  - Chumbera

Oxalis pes-caprae

- Español:
  - Agrillo
  - Agrio
  - Agrios
  - Agritos
  - Aleluya
  - Canario
  - Canarios
  - Cebolla
  - Cebollica
  - Dormilones
  - Flor de sueño
  - Flor del sueño
  - Matacañas
  - Matapán
  - Talle
  - Trebina
  - Trebo
  - Trébol
  - Villanetas
  - Vinagrera
  - Vinagreras
  - Vinagreta
  - Vinagrillo

Pinus nigra

- Español:
  - Ampudio
  - Carrasco
  - Empudio
  - Laricio
  - Pin blanco
  - Pincarrasco
  - Pino
  - Pino albar
  - Pino ampudio
  - Pino austriaco
  - Pino blanco
  - Pino blanco laricio
  - Pino carrasco
  - Pino carrax
  - Pino cascalbo
  - Pino cascallo
  - Pino falgareño
  - Pino gargallo
  - Pino laricio
  - Pino maderero
  - Pino nasarre
  - Pino nassarre
  - Pino nassarro
  - Pino nazaron
  - Pino nazarro
  - Pino nazarrón
  - Pino nazarón
  - Pino negral
  - Pino negral español
  - Pino negro
  - Pino pudio
  - Pino real
  - Pino salgareño
  - Pino sargareño
  - Pino silvestre
  - Pudio
  - Sargareño

Pinus pinaster

- Español:
  - Pin rueno
  - Pino
  - Pino bermejo
  - Pino borde
  - Pino bravo
  - Pino carrasco
  - Pino de burdeos
  - Pino de flandes
  - Pino de aire
  - Pino de la resina
  - Pino de las landas
  - Pino de las landas
  - Pino de madera
  - Pino de resina
  - Pino gallego
  - Pino laricio
  - Pino marítimo
  - Pino morisco
  - Pino negral
  - Pino negrillo
  - Pino negro
  - Pino nevral
  - Pino oficinal
  - Pino pinagral
  - Pino resinero
  - Pino rodeno

- Pino rodero
- Pino rodesno
- Pino rodezno
- Pino royal
- Pino rubial
- Pino ruezno
- Pino ródeno
- Pino volador
- Pinos
- Piñón
- Rodeno
- Royal

- Asturiano:
  - Pino
  - Pinu

Pinus pinea

- Español:
  - Cañamón
  - Doncel
  - Doncell
  - Pin doncel
  - Pino
  - Pino albal
  - Pino albar
  - Pino alvar
  - Pino arbal
  - Pino blanquillo
  - Pino bueno
  - Pino de comer
  - Pino de la sierra
  - Pino de la tierra
  - Pino de madera
  - Pino de piñones
  - Pino de piñón gordo
  - Pino doncel
  - Pino manso
  - Pino parasol
  - Pino piñero
  - Pino piñonero
  - Pino piñorero
  - Pino real
  - Pino redondo
  - Pino romano
  - Pino rubial
  - Pino uñal
  - Pino vero
  - Pinos
  - Pinu
  - Pinón
  - Piña
  - Piña albar
  - Piñal
  - Piñón

Pinus radiata

- Español:
  - Pino
  - Pino de monterrey
  - Pino insigne
  - Pino insignis
  - Pino silvestre
  - Pino volador

- Asturiano:
  - Pino
  - Pinu

Pinus sylvestris

- Español:
  - Valsaín
  - Aznacho
  - Aznallo
  - Escupineiro
  - Pinabete
  - Pineda de carboneros
  - Pinete
  - Pino
  - Pino valsaín
  - Pino albal
  - Pino albar
  - Pino albar silvestre
  - Pino balsain
  - Pino balsaín
  - Pino bermejo
  - Pino blancal
  - Pino blanco
  - Pino blanquillo
  - Pino borde
  - Pino bravo
  - Pino bravío
  - Pino brozno
  - Pino carrasco
  - Pino chopo
  - Pino común
  - Pino cortezudo
  - Pino cortezuelo
  - Pino de balsain
  - Pino de escocia
  - Pino de valsaín
  - Pino de balsaín
  - Pino enebral
  - Pino gallego
  - Pino maderable
  - Pino maderero

- Pino nebral
- Pino negral
- Pino negro
- Pino real
- Pino rojal
- Pino rojo
- Pino royal
- Pino royano
- Pino royo
- Pino serrano
- Pino silvestre
- Piñal
- Teda

Platanus orientalis

- Español:
  - Platanera
  - Platano de levante
  - Plátano
  - Plátano de levante
  - Plátano de oriente
  - Plátano de paseo
  - Plátano de sombra
  - Plátano oriental
  - Plátanos de sombra

Populus alba

- Español:
  - Albar
  - Alemo
  - Blancón
  - Chopo
  - Chopo blanco
  - Chopo silvestre
  - Hierba de la ruda
  - Peralejo
  - Pobo
  - Pobo albar
  - Tiemblo
  - Triamoleta
  - Trémol
  - Álamo
  - Álamo alvar
  - Álamo blanco
  - Álamo blanco
  - Álamo blanquillo
  - Álamo común
  - Álamo plateado
  - Álamo temblón
  - Álamo-blanco
  - Álamu
  - Árbol blanco
- Asturiano:
  - Carolín
  - Chopo
  - Palero
  - Álamo

Populus nigra

- Español:
  - Chopa
  - Chopera
  - Chopo
  - Chopo blanquillo
  - Chopo canadiense
  - Chopo castellano
  - Chopo común
  - Chopo de lombardía
  - Chopo del país
  - Chopo lombardo
  - Chopo mosquitero
  - Chopo negral
  - Chopo negro
  - Chopo piramidal
  - Chopola
  - Chopro
  - Chopru
  - Chopu
  - Choupo
  - Guelguero
  - Leopardo
  - Lombardo
  - Lombardu
  - Morra
  - Negral
  - Negrillo
  - Palero
  - Pobo
  - Populo
  - Povero
  - Povisa
  - Povo
  - Puba
  - Pópulo
  - Sopo
  - Soupo
  - Temblero
  - Temblu
  - Tembo
  - Templu
  - Tiemblo
  - Viga
  - Álamo
  - Álamo de italia
  - Álamo de lombardía
  - Álamo negral
  - Álamo negrillo
  - Álamo negro
  - Álamos peralejos
- Asturiano:
  - Chopo
  - Chopu
  - Choupo

- Álamu

Populus tremula

- Español:
  - Alamillo
  - Aliso
  - Chopo
  - Chopo canariense
  - Chopo falso
  - Chopo temblón
  - Chopo tiemblo
  - Lamparilla
  - Llamo
  - Pobo
  - Temblón
  - Tiemblo
  - Tremol
  - Tremoleta
  - Tremolin
  - Tremolina
  - Tremolín
  - Álamo
  - Álamo bastardo
  - Álamo blanco
  - Álamo líbico
  - Álamo temblón altoaragonés: chopo temblón
  - Escaleral
  - Temblero
  - Temblón
  - Termoleta
  - Tiemblo
  - Treamoleta
  - Trembolera
  - Tremoleta
  - Tremolín
  - Triámbol
  - Triámol
  - Trémol
  - Trémul
- Asturiano:
  - Chopu
  - Chopu silvestre

Prunus avium

- Español:
  - Adoña
  - Albar
  - Albaruco
  - Carne de vaca
  - Cerdeiro
  - Cerecera
  - Cerecero
  - Cerecino
  - Cereisal
  - Cereixal
  - Cereizal
  - Ceresera
  - Cereza
  - Cereza ampollar
  - Cereza blanca
  - Cereza brava
  - Cereza castellana
  - Cereza de jaraguy
  - Cereza de sierra nevada
  - Cereza de agua
  - Cereza de corazón de cabrito
  - Cereza de costal
  - Cereza de la piedra
  - Cereza encarnada
  - Cereza garrafal
  - Cereza mollar
  - Cereza negra
  - Cereza pitorrera
  - Cereza tardía
  - Cerezal
  - Cerezal garrafal
  - Cerezar
  - Cerezas
  - Cerezas adoñas
  - Cerezas corazón de cabrito fruto
  - Cerezas de costal fruto
  - Cerezas mollares fruto
  - Cerezas morrinas
  - Cerezo
  - Cerezo borde
  - Cerezo bravo
  - Cerezo bravío
  - Cerezo ciento en boca
  - Cerezo común
  - Cerezo de aves
  - Cerezo de cerezas costaleras
  - Cerezo de flor doble
  - Cerezo de monte
  - Cerezo durazno
  - Cerezo garrafal
  - Cerezo mollar
  - Cerezo morrino
  - Cerezo negro

- Cajigá
- Cajigón
- Carbizo
- Carbizu
- Carcoxa
- Carrasca
- Carrascal
- Carrascalejo
- Cascabillo
- Caxigo
- Caxigos
- Caxiquel
- Chaparra
- Corbizo
- Encina
- Enciniego
- Gaña agalla
- Gállara
- Llata
- Macharu
- Matoju
- Matorro
- Queixigo
- Quejido
- Quejiga
- Quejigal
- Quejigo
- Quejio

- Douglasia verde
- Falsa tsuga verde de las rocosas
- Pino oregón
- Pino de oregón
- Abeto douglas
- Abeto de douglas

Quercus faginea

- Español:
  - Albar
  - Barda
  - Barroscu
  - Bellota
  - Berroscu
  - Berrosu
  - Billota
  - Caixigo
  - Caixigu
  - Cajibo
  - Cajiga
  - Cajigo
  - Cajigona
  - Cajigu

- Cereza de a onza
- Cereza de bocáu
- Cereza gallega
- Cereza gallera
- Cereza montesa
- Cereza negra
- Cereza ronsoria
- Cerezal
- Cerezo
- Cerezu
- Cireiza
- Cireizal
- Creixa
- Mourán
- Zereizal
- Zreisal
- Zreixal
- Zreizal
- Zriza

Pseudotsuga menziessii

- Español:
  - Cerezo salvaje
  - Cerezo silvestre
  - Cerezu
  - Cerozal silvestre
  - Cirasera
  - Cireixal
  - Cireixeis
  - Cireixel
  - Cireixo
  - Ciresera
  - Cirezal
  - Creisal
  - Cresa
  - Freisal
  - Gorronudas
  - Guereciga
  - Guindo
  - Guindo garrafal
  - Guindo morrino
  - Guindo silvestre
  - Guindo zorrero
  - Maroviña
  - Mollar
  - Morrino
  - Morriñas
  - Picotas
  - Seresero
  - Sirera
  - Ziresera
  - Zreisal
  - Zreiza
  - Zrisal
  - Zrizal altoaragonés: sirerera
  - Siresera
  - Sirisera
  - Zerezera
  - Zirasera
  - Zirazera
  - Ziresera
- Asturiano:
  - Cereiceira
  - Cereixal
  - Cereixeira
  - Cereixeiro
  - Cereiza
  - Cerexo
  - Cereza
  - Cereza chinchona

- Bellotas
- Cajiga
- Carbayo
- Carbizo
- Carrasca
- Carrasco
- Carrascu
- Carrasqueira
- Carvallo
- Carvallu
- Cassa
- Chaparrera
- Chaparro
- Follaco
- Jaro
- Llata
- Marfueyo
- Mata
- Matorro
- Potes
- Queixigo
- Rebola
- Rebollo
- Rebolo
- Relechu
- Roble
- Roble albar
- Roble albar
- Encinuche
- Enciá
- Incina
- Lancina
- Mata
- Mata negra
- Mata parda
- Matacán
- Matajarda
- Mataparda
- Matizo
- Matorra parda
- Mesto
- Pardo
- Resalbo
- Saparra
- Sardón
- Tramilla

Quercus petraea

- Español:
  - Albar
  - Albariza
  - Alcorque
  - Argayero
  - Barda
  - Bardión
  - Candelitas
  - Carrapito
  - Carrasca
  - Carrascas
  - Carrasco
  - Carrasqueira
  - Carrasquera
  - Carrasquero
  - Carrasquizo
  - Cascabillo
  - Chamorro
  - Chaparra
  - Chaparrera
  - Chaparrete
  - Chaparro
  - Chaparros
  - Chaparru
  - Curtío
  - Dul
  - Encina
  - Encina de bellotas dulces
  - Encina dulce
  - Encineta
  - Encino
  - Tallar
  - Zatorro
- Asturiano:
  - Caxiga
  - Caxigo
  - Caxigu
  - Caxigueiro

Quercus ilex

- Español:
  - Abeyoto
  - Albarrán
  - Alcina
  - Ancina
  - Anciá
  - Ancía
  - Arcina
  - Barda
  - Bardión
  - Bellota
  - Bellota dulce
  - Bellotas
  - Bellotas alvellanás
  - Bellotas avellenás
  - Beyotero
  - Candelillas
  - Quejío
  - Ramallo
  - Rebolla
  - Rebollo
  - Roble
  - Roble ancina
  - Roble andaluz
  - Roble carraspo
  - Roble carrasqueño
  - Roble encimego
  - Roble encina
  - Roble enciniego
  - Roble matorrizo
  - Roble negral
  - Roble quejido
  - Roble quejigo
  - Roble quejío
  - Roble tejido
  - Robre

- Roble albar-almezo
- Roble albero
- Roble carballo
- Roble común
- Roble de fruto sentado
- Roble de fruto sentado y axilar
- Roble de montaña
- Roble enciniego
- Roble macho
- Roble matiego
- Roble matizo
- Roble negral
- Roble propio
- Robolo
- Robre
- Robri
- Roure
- Tocorno
- Trampa
- Zatorro
- Zufreiro
- Asturiano:
  - Abeyota
  - Bellota
  - Beyota
  - Carbayu turcu
  - Casomera curcutsu
  - Caxigu
  - Rebutsu cocoxo
  - Roble
  - Roble albar
  - Robre caxigu
  - Robri
  - Tsandre

Quercus pyrenaica

- Español:
  - Abocayero
  - Abogalla
  - Agallarones
  - Agallones
  - Agayero
  - Barda
  - Bardal
  - Bardión
  - Bellota
  - Bellotas
  - Cajiga
  - Cajigo
  - Carballal
  - Carballo
  - Carrasco
  - Carvalho negral
  - Carvallo
  - Carvallo negro
  - Cerqueiro
  - Cerquiño
  - Chaparro
  - Chaporro
  - Corcabo
  - Corco
  - Cuerco
  - Curco villano
  - Gallaras
  - Malojo
  - Manoplas blancas
  - Marojo
  - Mata de robles
  - Matorra
  - Matorro
  - Melojo
  - Mesto
  - Negral
  - Potes
  - Quejigo
  - Rachizo
  - Reboll
  - Rebolla
  - Rebollar
  - Rebollera
  - Rebolleta
  - Rebollo
  - Rebollo gordo
  - Rebollón
  - Rebolo
  - Rebotsu
  - Roble
  - Roble albar
  - Roble borne
  - Roble cepillo
  - Roble común de castilla
  - Roble corchizo
  - Roble dulce
  - Roble escorche
  - Roble matorrizo
  - Roble melojo
  - Roble montaraz
  - Roble negral
  - Roble negro
  - Roble que lleva las nueces de agallas
  - Roble sapiego
  - Roble serrano
  - Roble tocio
  - Roble tocorno
  - Roble tozo
  - Roble turco
  - Roble villano

- Robleda
- Robleda doble
- Robre
- Robre
- Sapiego
- Talaya
- Tocio
- Tociu
- Tocorno
- Tocío
- Tozo
- Turco

- Asturiano:
  - Caxigu
  - Corquiu
  - Rebochu
  - Rebollada
  - Rebollu
  - Rebotsa
  - Rebutsu
  - Reutsu
  - Roble
  - Roble argañuzu
  - Roble carbayu
  - Roble tociu
  - Sapiego

Quercus robur

- Español:
  - Abellota
  - Abillote
  - Albar
  - Ariza
  - Bellota
  - Cagiga
  - Cajico
  - Cajiga
  - Cajigo
  - Cajigona
  - Calvallo
  - Carba
  - Carbajo
  - Carballo
  - Carbalo
  - Carbayo
  - Carbayu curcu
  - Carvallo
  - Carvayu
  - Cil
  - Grana
  - Llandera
  - Marfueyo
  - Marojo
  - Mata
  - Quejibo
  - Rebola
  - Rebollo albar
  - Roble
  - Roble albar
  - Roble carballo
  - Roble común
  - Roble de fruto con pezón largo
  - Roble de fruto en racimos
  - Roble de manoplas
  - Roble europeo
  - Roble fresnal
  - Roble marfueyo
  - Roble paisano
  - Roble pedunculado
  - Roble penduculado
  - Roble pie de marro
  - Roble piramidal
  - Robre
  - Robre albal

- Asturiano:
  - Abechota
  - Abeyota
  - Bellota
  - Cabayo
  - Cahíu
  - Camayo
  - Carbachal
  - Carbacho
  - Carbachu alvar
  - Carballo
  - Carbayo
  - Carbayu
  - Carvayo
  - Caxigu
  - Pipa
  - Potsisco
  - Potsiscu curcu
  - Rebosu
  - Rebotso
  - Rebotso albar
  - Rebotso cuartial
  - Rebotso curcu
  - Rebotso turcu
  - Roble
  - Roble albar
  - Roble carcoxu
  - Robre
  - Robre villandigu
  - Sandre
  - Sapiego
  - Tsandre

Quercus rotundifolia

- Español:
  - Aladierno
  - Ancina
  - Ancino
  - Bellota
  - Bellota dulce
  - Bellotero
  - Billota
  - Carrapito
  - Carrasca
  - Carrasca de bellotas dulces

- Carrasca dulce
- Carrasco
- Carrascu
- Carrasquera
- Carrasquero
- Chabasco
- Chabasquera
- Chaparra
- Chaparra blanca
- Chaparrera
- Chaparreta
- Chaparro
- Chavasco
- Coscoja
- Encina
- Encina avellanada
- Encina de salamanca
- Encina de bellotas dulces
- Encina de fruto dulce
- Encina joven
- Encina nueva
- Encina silvestre
- Encineta
- Encineto
- Encino
- Gla
- Incina
- Jardonal
- Jardón
- Maraña
- Mata parda
- Matacán
- Mataparda
- Mesto de bellotas como avellanas
- Ramasco
- Sarda
- Sardonal
- Sardonera
- Sardón
- Xardonal
- Xardón
- Asturiano:
  - Ancina

Quercus rubra

- Español:
  - Roble americano
- Asturiano:
  - Roble

Quercus suber

- Español:
  - Acornoque
  - Alcornoque
  - Alcornoque
  - Alcornoque extremeño
  - Alcornoque morisco
  - Alcornoquera
  - Alcornoqui
  - Alcorque
  - Alsina surera
  - Arconoque
  - Arcornoque
  - Azufreiro
  - Bellota
  - Bohnizo
  - Bornizo
  - Chaparreta
  - Chaparro
  - Corcheo
  - Corchero
  - Corchero
  - Corchizo
  - Corco
  - Gebrera
  - Gebrero
  - Machero
  - Merino
  - Moheda
  - Pimpollo
  - Roble sobrero
  - Sobreiro
  - Sobrero
  - Sobrero
  - Sofrero
  - Sufra
  - Sufreira
  - Sufreiro
  - Suro
  - Tornadizo
  - Trepoyo
  - Zafrero
  - Zufreiro
  - Zufrero
  - Árbol del corcho
- Asturiano:
  - Corchegu
  - Rebustsu
  - Sufreira

Salix salviifolia

- Español:
  - Ahoz
  - Barda
  - Barda negra
  - Bardagera
  - Bardaguera
  - Bardaguera blanca
  - Mimbrera
  - Sacera blanca
  - Sacera negra
  - Saceras
  - Salguera
  - Salguera de plata
  - Salguera sapiega

- Salguero
- Salguero sapiego
- Saoce
- Saoz
- Sarga
- Sauce
- Sauce blanco de portugal
- Saz
- Sazga
- Verguera
- Vimbre
- Zaoz
- Zaragato
- Zargatera
- Zargatillo
- Zauz
- Zaz

Senecio inaequidens

- Español:
  - Botón de oro

Spartina patens

- Español:
  - Espartillo de cangrejal
  - Borraza

Taxus baccata

- Español:
  - If
  - Iphi
  - Mataburros
  - Sabina
  - Sabino
  - Tajo
  - Taxo
  - Teijo
  - Teiso
  - Teixeda
  - Teixo
  - Teixu
  - Tejo
  - Tejo
  - Tejo blanco
  - Tejo común
  - Tejo encarnado
  - Tejo negro
  - Tejuelo
  - Tejón
  - Teo
  - Tesio
  - Texeira
  - Texio
  - Texo
  - Tiju
  - Toxo
  - Té hormiguero
- Asturiano:
  - Chochín
  - Mocu
  - Tehu
  - Teixo
  - Teixu
  - Tejo
  - Texo
  - Texu
  - Tisu

Tradescantia fluminensis

- Español:
  - Oreja de gato

Ulex europaeus

- Español:
  - Abolaga
  - Abulaga
  - Albar
  - Albolaga
  - Albolaga basta
  - Albolaguera basta
  - Albulaga
  - Aliaga
  - Aliaga de europa
  - Alisapa
  - Arbolaga
  - Arbolaguín
  - Arbulaga
  - Ardevilla
  - Ardivieja
  - Argoma
  - Argoma de santander
  - Argoma gallega
  - Argulaga
  - Arma
  - Arnaz
  - Arnello
  - Arnelo
  - Aulaga
  - Aulaga común
  - Bulaga
  - Carqueja
  - Cotolla
  - Cádava
  - Escajo
  - Escaju
  - Espino
  - Gateño
  - Guiri
  - Gáraba
  - Iscaju
  - Jabulaga
  - Olaga
  - Ollaga
  - Otaca
  - Pitus mourus
  - Rebolla
  - Rozo
  - Tocho
  - Tocio
  - Toco
  - Toho
  - Toiso
  - Toitso
  - Toixo
  - Tojo
  - Tojo de galicia
  - Toso
  - Toxo
  - Toya

- Toyo
- Ulaga
- Árgoma
- Árguma bable: arguimas
- Cacadavas
- Cadavas
- Cadaves
- Cotolla
- Cotoxa
- Cotoya
- Cádava
- Cádaves
- Gándaras
- Navia de toxo
- Rebolla
- Rozu
- Toxo
- Toxo bravo
- Toxo manso
- Toxu
- Árgoma
- Árguma
- Asturiano:
- Toxo

B.4. Contenido BBDD completo

Contenido completo de la base de datos, descrita en 4: Base de datos.

N_Cientifico	Scientific_Name	Endemic	Genus	Species	Class	Order	Family	cota_min	cota_max	op_cota_min	op_cota_max	PMA_min	PMA_max	op_PMA_min	op_PMA_max	PV_min	PV_max	TMA_min	TMA_max	op_TMA_min	op_TMA_max	TMC_min	TMC_max	TMF_min	TMF_max	op_TMF_min	op_TMF_max	HT	SE	Prof	pH_min	pH_max	procedencia	invasora	uicn	catalogo_ast
Castanea sativa	Castanea sativa Mill.	FALSE	Castanea	sativa	Magnoliopsida	Fagales	Fagaceae	0	1200	200	800	450	2000	600	1100	75	150	6	14	9.8	13.4	15	24	1	5	1	5	si	si	60	4.5	6.5	autóctona	no	LC	no
Abies alba	Abies alba Mill.	FALSE	Abies	alba	Pinopsida	Pinales	Pinaceae	700	2200	1400	1800	800	2150	900	1700	205	385	3.3	11	6	9.4	10	18	-6.4	0	-3	-2.4	si	si	60	3.5	7	alóctona	no	LC	no
Pinus pinaster	Pinus pinaster Aiton	FALSE	Pinus	pinaster	Pinopsida	Pinales	Pinaceae	0	1100	100	400	900	1600	400	0	20	125	10.4	14.1	12	13	20.6	28.9	1	7	0.5	6.2	no	si	60	5.5	8	alóctona	no	LC	no
Acer pseudoplatanus	Acer pseudoplatanus L.	FALSE	Acer	pseudoplatanus	Magnoliopsida	Sapindales	Sapindaceae	300	1800	600	1000	900	1770	900	1770	125	260	6	14	6	13.1	17	26.4	-4	4	-0.6	3.9	no	no	40	4.5	7	autóctona	no	LC	no
Pinus sylvestris	Pinus sylvestris L.	FALSE	Pinus	sylvestris	Pinopsida	Pinales	Pinaceae	800	2000	1600	1700	600	1200	680	900	100	2555	6	12	3.4	11	15	20	-1	3	-6.4	-1	no	si	0	4	7	autóctona	no	LC	no
Pinus nigra	Pinus nigra Aiton	FALSE	Pinus	nigra	Pinopsida	Pinales	Pinaceae	800	2000	1000	1500	600	1200	480	980	60	130	9	12	7.4	13.3	20	23	1	4			no	no	0	4.5	7.6	alóctona	no	LC	no
Pinus pinea	Pinus pinea L.	FALSE	Pinus	pinea	Pinopsida	Pinales	Pinaceae	0	1000	30	800	430	800	460	600	15	125	11	18	11.7	16	16	21	-1.4	11	3	7.8	no	no	10	4	9	alóctona	no	LC	no
Pinus radiata	Pinus radiata D.Don	FALSE	Pinus	radiata	Pinopsida	Pinales	Pinaceae	100	500	100	500	1000	2000	1200	1600	100	290	10	13	11	12	16	20	4	8	4	8	si	no	10	4.1	5.7	alóctona	no	EN	no
Pseudotsuga menziesii	Pseudotsuga menziesii (Mirb.) Franco	FALSE	Pseudotsuga	menziessii	Pinopsida	Pinales	Pinaceae	300	1800	800	1200	300	1800	800	1200	230	300	8	10	7	9	14	15	0	5	0	5	no	no	30	5	6	alóctona	no	no	no
Larix decidua	Larix decidua Mill.	FALSE	Larix	deciudua	Pinopsida	Pinales	Pinaceae	250	2200	1100	1900	500	2555	600	2000	100	200	4	10	5	8	22	30	0	5	0	5	no	si	0	5.5	7	alóctona	no	no	no
Fagus sylvatica	Fagus sylvatica L.	FALSE	Fagus	sylvatica	Magnoliopsida	Fagales	Fagaceae	0	2000	900	1600	600	1000	700	900	110	200	3	10	8	9	7.3	18	0	7.3	0	5	si	si	40	4	8.5	autóctona	no	LC	no
Quercus robur	Quercus robur L.	FALSE	Quercus	robur	Magnoliopsida	Fagales	Fagaceae	0	1500	0	1000	600	2555	1050	2160	200	2555	10	13.3	10	13.3	14	25	-10	4.9	-10	-0.6	si	no	20	5	7.5	autóctona	no	LC	no
Acer campestre	Acer campestre L.	FALSE	Acer	campestre	Magnoliopsida	Sapindales	Sapindaceae	600	1600	1200	1500	710	1550	800	910	110	350	6.8	11.9	7.2	9.6	-4.1	0.9	21.6	27.9	21.6	27.9	no	no	20	5.5	8.5	autóctona	no	LC	no
Fraxinus angustifolia	Fraxinus angustifolia Vahl	FALSE	Fraxinus	angustifolia	Magnoliopsida	Lamiales	Oleaceae	0	1500	200	500	500	2555	600	2000	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	6	8	autóctona	no	LC	Interés Especial
Ilex aquifolium	Ilex aquifolium L.	FALSE	Ilex	aquifolium	Magnoliopsida	Aquifoliales	Aquifoliaceae	500	1500	800	1400	600	1500	sd	sd	sd	sd	sd	sd	sd	sd	12	20	-1	10	sd	sd	si	si	sd	4.5	7.5	autóctona	no	LC	Interés Especial
Quercus faginea	Quercus faginea broteroi (Cout.) A. Camus	FALSE	Quercus	faginea	Magnoliopsida	Fagales	Fagaceae	200	1500	250	500	350	1400	590	970	135	270	8	16	11	15.1	26.9	31.2	-3.6	5	-3	-0.1	no	no	30	4	9	autóctona	no	LC	Interés Especial
Quercus ilex	Quercus ilex L.	FALSE	Quercus	ilex	Magnoliopsida	Fagales	Fagaceae	0	1600	0	1000	500	2555	500	2000	150	2555	10	18.1	11.4	18	25	45	-3	11	-2.2	2.3	no	no	10	5.5	9	autóctona	no	LC	Interés Especial
Quercus rotundifolia	Quercus rotundifolia Lam.	FALSE	Quercus	rotundifolia	Magnoliopsida	Fagales	Fagaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	5.5	9	autóctona	no	LC	Interés Especial
Quercus suber	Quercus suber L.	FALSE	Quercus	suber	Magnoliopsida	Fagales	Fagaceae	0	1300	300	600	570	1100	594	1060	15	165	13	18.1	13.7	16	20	36.9	0.9	7.8	4	5	no	no	50	3.5	5.5	autóctona	no	LC	Interés Especial
Salix salvifolia	Salix salvifolia Brot.	TRUE	Salix	salvifolia	Magnoliopsida	Malpighiales	Salicaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	5.5	8	autóctona	no	LC	Interés Especial
Taxus baccata	Taxus baccata L.	FALSE	Taxus	baccata	Pinopsida	Cupressales	Taxaceae	100	2000	300	1400	600	2600	sd	sd	sd	sd	10	13.3	sd	sd	sd	sd	sd	sd	sd	sd	si	si	sd	4	9.6	autóctona	no	LC	Interés Especial
Ulex europaeus	Ulex europaeus L.	FALSE	Ulex	europaeus	Magnoliopsida	Fabales	Fabaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	autóctona	no	LC	no
Quercus petraea	Quercus petraea (Matt.) Liebl.	FALSE	Quercus	petraea	Magnoliopsida	Fagales	Fagaceae	0	1500	100	1000	600	2555	600	1970	150	2555	4.4	15	5	10.4	15	25	-6.1	7	-3	-1	no	no	20	4.7	8	autóctona	no	LC	no
Quercus pyrenaica	Quercus pyrenaica Willd	FALSE	Quercus	pyrenaica	Magnoliopsida	Fagales	Fagaceae	0	2000	800	1200	600	2555	660	1970	125	2555	6	16	6.4	11.8	12	22	-5	7	-4.5	0	no	no	40	3.5	5.5	autóctona	no	LC	no
Abies pinsapo	Abies pinsapo Boiss.	FALSE	Abies	pinsapo	Pinopsida	Pinales	Pinaceae	450	1000	1650	1800	700	2080	800	2000	20	45	12	17.2	13	15	17	30.8	2.5	6.4	3	5	no	si	20	5	7	alóctona	no	EN	no
Betula alba	Betula pendula Roth	FALSE	Betula	alba	Magnoliopsida	Fagales	Betulaceae	0	2000	800	1800	350	2150	970	1800	120	300	3.2	18	6	12.7	10	18	-25	10	-0.3	7.5	no	no	20	5	7.5	autóctona	no	LC	no
Juniperus thurifera	Juniperus thurifera L.	FALSE	Juniperus	thurifera	Pinopsida	Cupressales	Cupressaceae	250	2000	800	1600	300	2000	450	740	50	130	7.5	14	8	12	24	34	-4.5	0.5	-4.2	-0.5	no	no	10	4	9	autóctona	no	LC	no
Quercus rubra	Quercus rubra L.	FALSE	Quercus	rubra	Magnoliopsida	Fagales	Fagaceae	0	600	100	480	600	1300	sd	sd	sd	sd	3	18	sd	sd	23	30	-18	15	sd	sd	si	si	30	4.5	8	alóctona	no	LC	no
Juglans regia	Juglans regia L.	FALSE	Juglans	regia	Magnoliopsida	Fagales	Juglandaceae	400	1500	460	800	700	2555	sd	sd	100	2555	15	30	sd	sd	sd	sd	sd	sd	sd	sd	si	si	60	4	7.5	alóctona	no	LC	no
Fraxinus excelsior	Fraxinus excelsior L.	FALSE	Fraxinus	excelsior	Magnoliopsida	Lamiales	Oleaceae	0	1800	500	1000	600	2090	820	1000	135	375	3.3	13	7	12	20	30	-6.4	6	-5	0.1	si	si	40	5.5	8	autóctona	no	NT	no
Platanus orientalis	Platanus orientalis L.	FALSE	Platanus	orientalis	Magnoliopsida	Proteales	Platanaceae	0	800	300	600	700	2000	sd	sd	sd	sd	4	21	6	12	20	32	-12	-1	sd	sd	no	no	sd	5.5	7.9	alóctona	no	VU	no
Prunus avium	Prunus avium L.	FALSE	Prunus	avium	Magnoliopsida	Rosales	Rosaceae	0	2000	400	700	650	1800	750	1550	84	144	4	14	8	12.1	16	28	-20	0	-6.4	-1	si	si	60	4.5	7.5	autóctona	no	LC	no
Alnus glutinosa	Alnus glutinosa (L.) Gaertn.	FALSE	Alnus	glutinosa	Magnoliopsida	Fagales	Betulaceae	0	1500	700	800	400	1820	800	1300	110	305	1	18	10.3	14.6	15	25	-30	8	0.6	6.4	si	no	40	5.5	7.2	autóctona	no	LC	no
Populus nigra	Populus nigra L.	FALSE	Populus	nigra	Magnoliopsida	Malpighiales	Salicaceae	0	1800	300	1000	300	1000	480	980	45	125	7.4	17	9	13.3	18	31	-5	12	-4.1	0.4	no	si	40	4.5	8.5	autóctona	no	DD	no
Populus alba	Populus alba L.	FALSE	Populus	alba	Magnoliopsida	Malpighiales	Salicaceae	0	1200	300	1000	300	1000	310	630	10	90	12.6	18	sd	sd	29.1	36.9	-1.5	6.3	sd	sd	si	no	20	6	7	alóctona	no	LC	no
Populus tremula	Populus tremula L.	FALSE	Populus	tremula	Magnoliopsida	Malpighiales	Salicaceae	0	2000	300	1400	300	1360	720	1000	140	375	3.9	15	7	11.1	16	28	-7	10	-6.4	-1	sd	sd	sd	sd	sd	autóctona	no	LC	no
Eucalyptus globulus	Eucalyptus globulus Labill.	FALSE	Eucalyptus	globulus	Magnoliopsida	Myrtales	Myrtoideae	0	400	0	400	550	1500	600	1400	140	400	9	23	18	21	19	31	0	15	1	10	si	si	10	5.5	8.5	alóctona	no	LC	no
Eucalyptus camaldulensis	Eucalyptus camaldulensis Dehnh.	FALSE	Eucalyptus	camaldulensis	Magnoliopsida	Myrtales	Myrtoideae	0	600	50	500	500	2555	sd	sd	sd	sd	5	16	sd	sd	21	41	0	22	sd	sd	si	no	10	sd	sd	alóctona	no	LC	no
Acacia dealbata	Acacia dealbata Link	FALSE	Acacia	dealbata	Magnoliopsida	Fabales	Mimosoideae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Agave americana	Agave americana L.	FALSE	Agave	americana	Liliopsida	Asparagales	Asparagaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	LC	no
Ailanthus altissima	Ailanthus altissima (Mill.) Swingle	FALSE	Ailantus	altissima	Magnoliopsida	Sapindales	Simaroubaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Ambrosia artemisiifolia	Ambrosia artemisiifolia L.	FALSE	Ambrosia	artemisiifolia	Magnoliopsida	Asterales	Asteraceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Araujia sericifera	Araujia sericifera Brot.	FALSE	Araujia	sericifera	Magnoliopsida	Gentianales	Apocynaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Azolla filiculoides	Azolla filiculoides Lam.(Sub Azolla ssp.)	FALSE	Azolla	filiculoides	Pteridopsida	Salviniales	Azollaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Baccharis halimifolia	Baccharis halimifolia L.	FALSE	Baccharis	halimifolia	Magnoliopsida	Asterales	Asteraceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Buddleja davidii	Buddleja davidii Franch.	FALSE	Buddleja	davidii	Magnoliopsida	Lamiales	Scrophulariaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Carpobrotus acinaciformis	Carpobrotus acinaciformis (L.) L.Bolus	FALSE	Carpobrotus	acinaciformis	Magnoliopsida	Caryophyllales	Aizoaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Carpobrotus edulis	Carpobrotus edulis (L.) N.E.Br.	FALSE	Carpobrotus	edulis	Magnoliopsida	Caryophyllales	Aizoaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Cortaderia selloana	Cortaderia selloana (Schult. & Schult.f.) Asch.	FALSE	Cortaderia	selloana	Liliopsida	Commelinales	Poaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Eichhornia crassipes	Eichhornia crassipes (Mart.) Solms & Graebn. (Sub Cortaderia ssp.)	FALSE	Eichhornia	crassipes	Liliopsida	Commelinales	Pontederiaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Fallopia baldschuanica	Eichhornia crassipes (Mart.) Solms	FALSE	Fallopia	baldschuanica	Magnoliopsida	Caryophyllales	Polygonaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Fallopia japonica	Fallopia baldschuanica (Regel) Holub	FALSE	Fallopia	japonica	Magnoliopsida	Caryophyllales	Polygonaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Hedychium gardnerianum	Fallopia japonica (Houtt.) Ronse Decr.	FALSE	Hedychium	gardnerianum	Magnoliopsida	Zingiberales	Zingiberaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Ipomoea indica	Ipomoea indica (Burm.) Merr.	FALSE	Ipomoea	indica	Magnoliopsida	Sonaiales	Convolvulaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Opuntia ficus-indica	Opuntia ficus-indica (L.) Mill.	FALSE	Opuntia	indica	Magnoliopsida	Caryophyllales	Cactaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	DD	no
Oxalis pes-caprae	Oxalis pes-caprae L.	FALSE	Oxalis	caprae	Magnoliopsida	Oxalidales	Oxalidaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Senecio inaequidens	Senecio inaequidens DC.	FALSE	Senecio	inaequidens	Dycotyledoneae	Asterales	Asteraceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Spartina patens	Spartina patens (Aiton) Muhl.	FALSE	Spartina	patens	Liliopsida	Cyperales	Poaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no
Tradescantia fluminensis	Tradescantia fluminensis Vell.	FALSE	Tradescantia	fluminensis	Liliopsida	Commelinales	Commelinaceae	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	sd	alóctona	si	no	no

Tabla B.1: Contenido de la BBDD completo, a excepción de las dos columnas que contienen los nombres vernáculos. NOTA: solo se puede apreciar la tabla en formato digital haciendo zoom, no en físico

C | Código del SGBD

C.1. Main

```
import funciones as f
main=True
f.bienvenida()
while main:
    f.opciones()
    opcion=input("                                          ")
    if opcion=="1":
        continuar=True
        while continuar:
            f.buscar()
            print("")
            print("                                [enter] - Realizar otra búsqueda")
            print("                                [s]     - Salir al menu principal")
            sel=input("                                        ")
            if sel=="s":
                continuar=False
    if opcion=="2":
        continuar=True
        while continuar:
            f.consulta()
            print("")
            print("                                [enter] - Realizar otra consulta")
            print("                                [s]     - Salir al menu principal")
            sel=input("                                        ")
            if sel=="s":
                continuar=False

    if opcion=="3":
        break
```

C.2. Funciones

```
import csv
import unicodedata
from openpyxl.utils.cell import get_column_letter #etiqueta columna .csv

def remove_accents(input_str): #quitar tildes de una cadena
    '''quita tildes de una cadena'''
    nfkd_form = unicodedata.normalize('NFKD', input_str)
    return u"".join([c for c in nfkd_form if not unicodedata.combining(c)])

def remove_accents_lista(lista):
    '''quita las tildes de las cadenas de una lista y lo deja en minuscula'''
    l=[]
    for i in lista:
        l.append((remove_accents(i)).lower())
    return l

def llaves_a_lista(dicc):
    '''Saca las llaves de un diccionario y las mete en una lista'''
    r=[]
    for i in dicc:
        r.append(i)
    return r

def ordenar_dicc_valores(x):
    return dict(sorted(x.items(), key=lambda item: item[1],reverse=True))

def ordenar_dicc_keys(x):
    return sorted(x.keys(), key=lambda x:x.lower())

def copiar_dic(dicc):
    clon={}
    for i in dicc:
        clon[i]=dicc[i]
    return clon

def div_(st): #divide cadenas divididas por "_" en una lista
    lista=[]
    palabra=""
    count=1
    for i in st:
        if i=="_":
            #palabra=remove_accents(palabra)
```

```
43              #palabra.lower()
44              lista.append(palabra)
45              palabra=""
46          elif count==len(st):
47              palabra=palabra+i
48              lista.append(palabra)
49          else:
50              palabra=palabra+i
51          count+=1
52      return lista
53
54  def cient_lista(cient): #cient tiene que ser un nombre cientifico ej
55                          #"Castanea sativa"
56      '''devuelve lista [genero, epiteto específico] a partir del nombre
57      cientifico como cadena'''
58      palabra=""
59      r=[]
60      count=0
61      for i in cient:
62          if i==" ":
63              r.append(palabra)
64              palabra=""
65          else:
66              palabra=palabra+i
67          count+=1
68          if count==len(cient):
69              r.append(palabra)
70
71      return r
72
73  def import_csv(): #importar y generar lista
74      with open('Especies.csv') as File:
75          reader = csv.reader(File, delimiter=';', quotechar='.',
76                              quoting=csv.QUOTE_MINIMAL)
77          count=0 #contador de la fila del csv
78          lista=[]
79          for row in reader:
80              if count==0:
81                  header=row #header es una lista que contiene los nombres de
82                             #cada columna
83                  count+=1
84              else:
85                  lista.append(row)
86          return header,lista
```

```
87
88  def import_consulta(): #importar y generar lista
89      with open('consulta.csv') as File:
90          reader = csv.reader(File, delimiter=';', quotechar='.',
91                              quoting=csv.QUOTE_MINIMAL)
92          lista=[]
93          for row in reader:
94              lista.append(row)
95          return lista
96
97  def get_header():
98      '''header (nombres parámetros, 1a fila del .csv)'''
99      return csv_[0]
100
101 def get_list(): #lista que incluye sublistas con cada valor
102     '''lista (de listas) con cada valor de cada especie
103     Su longitud "len(getlist())" coincide con el numero de especies'''
104     return csv_[1]
105
106 def get_cient(): # lista nombres cientificos
107     '''lista de todos los nombres cientificos'''
108     lista={}
109     dlista={}
110     cont=0
111     for i in l: #linea CSV
112         for j in range(len(i)):
113             if j==0:
114                 if i[j] in dlista:
115                     print(24*"-")
116                     print("¡La especie", i[j],
117                           "está duplicada! Posiciones CSV:",
118                           lista.index(i[j])+2, "y", cont+2)
119                     print(24*"-")
120                     lista[i[j]]=cont+2
121                     dlista[i[j]]=cont
122                 else:
123                     lista[i[j]]=cont+2
124                     dlista[i[j]]=cont
125                 cont+=1
126             else:
127                 break
128     return lista
129
130 def get_posiciones(lista_nombres):
131     '''busca posiciones (numero de columna) de los nombres de los parámetros
```

```
132         en el header y las devuelve en una lista'''
133         l=[0] #se incluye 0 porque asi mas adelante no ignora el nombre
134               #cientifico. que está en la pos. 0
135         for i in lista_nombres:
136             l.append(header.index(i))
137         return l
138
139     def buscar_nv_header():
140         '''busca nombres vernaculos en el header y devuelve una posoicion con el
141         idioma y su posición'''
142         a={"ESP":1,"AST":2}
143         count=3
144         for i in header[3:]:
145             if i[0:5]=="N_Ver":
146                 a[i[7:len(i)-1]]=count
147             count+=1
148         return a
149
150     def get_nver():
151         '''Diccionario de nombres vernáculos (lista) a cada nombre científico
152         (key) dic={nombre cientifico: [ [lista esp], [lista ast] ]'''
153         nvheader=buscar_nv_header()
154         posnv=list(nvheader.values())
155         a={}
156         for i in l: #linea csv
157             for j in range(len(i)): #posición dentro de la linea; 0=nombre
158                                     #científico y 1=nombres vernáculos
159                 if j==0:  #añade asturiano y español
160                     if i[j] not in a:
161                         lt=[]
162                         lt.append(div_(i[j+1])) #0 = castellano
163                         lt.append(div_(i[j+2])) #1 = asturianu
164                         for m in posnv[2:]:
165                             lt.append(div_(i[m])) #el resto de idiomas
166                         a[i[j]]=lt
167
168         return a
169
170
171     def cotas(): #todas las cotas tienen que estar cubiertas para que funcione
172         umbral_min="cota_min"
173         umbral_max="cota_max"
174         optimo_min="op_cota_min"
175         optimo_max="op_cota_max"        #los nombres de los parámetros en el
176                                         #CSV!! Tienen que coincidir.
```

```
177
178         posiciones=get_posiciones([umbral_min,umbral_max,optimo_min,optimo_max])
179         cota_umbral={}
180         cota_optima={}
181         for i in l:
182             umbral=[]
183             optimo=[]
184             try:
185                 umbral.append(float(i[posiciones[1]])) #umbral min
186                 umbral.append(float(i[posiciones[2]])) #umbral max
187                 optimo.append(float(i[posiciones[3]])) #optimo min
188                 optimo.append(float(i[posiciones[4]])) #optimo max
189                 if i[0] not in cota_umbral:
190                     cota_umbral[i[0]]=umbral
191                     cota_optima[i[0]]=optimo
192             except:
193                 cota_umbral[i[0]]="error"
194                 cota_optima[i[0]]="error"
195                 print("")
196                 print("Error en lectura de cotas de", i[0])
197                 print("Comprobar en fila", ncient[i[0]])
198             else:
199                 for j in posiciones[1:]:
200                     if float(i[j])<0:
201                         print(" ")
202                         print("Error de cota:",header[j],"en",i[0],"fila",
203                               ncient[i[0]])  #comprueba valores negativos
204
205         return cota_umbral, cota_optima
206
207
208     def comprobar_4valor(valor,min_,max_,op1,op2): #comprueba si un valor
209                                                    #es compatible
210         try:
211             if op1<=valor and op2>=valor:
212                 return ["optimo"]
213             elif min_<=valor and max_>=valor:
214                 return ["compatible"]
215             else:
216                 if valor<min_:
217                     return ["incompatible", -round(min_-valor,2)]
218                 else:
219                     return ["incompatible", round((-(max_-valor)),2)]
220         except:
221             try:
```

```
222             return comprobar_2valor(valor, min_, max_)
223         except:
224             return "error"
225 
226 def comprobar_2valor(valor,v_inf,v_sup):
227     '''Comprueba si un valor entra dentro de un rango definido por valor
228     superior e inferior'''
229     try:
230         if v_inf<=valor and v_sup>=valor:
231             return ["apto"]
232         else:
233             if valor<v_inf:
234                 return ["incompatible", -round(v_inf-valor,2)]
235             else:
236                 return ["incompatible", round(-(v_sup-valor),2)]
237     except:
238         return "error"
239 
240 def get_columna(nombre_columna):
241     valor={}
242     for i in l:
243         if i[get_posiciones([nombre_columna])[1]]=='':
244             valor[i[0]]="error"
245         else:
246             valor[i[0]]=i[get_posiciones([nombre_columna])[1]]
247     return valor
248 
249 
250 def comprobar_cota(valor):
251     '''Comprueba si el valor está dentro del rango optimo o compatible'''
252     comp_cota={}
253     for i in cotas_umbral:
254         a=comprobar_4valor(valor, cotas_umbral[i][0], cotas_umbral[i][1],
255                            cotas_optimo[i][0], cotas_optimo[i][1])
256         comp_cota[i]=a
257     return comp_cota
258 
259 def buscar():
260     '''busca:
261         - Especies
262         - Generos
263         - Nombres comunes'''
264     print(" ")
265     #print("-",len(ncient), "especies forestales en la BBDD")
```

```
266     ncomun=input("                                                  Búsqueda: ")
267     #ncomun=remove_accents(ncomun)
268     print("")
269     def buscar_lista_nv(ncomun,lista):
270         '''busca si un nombre comun está en una lista (que contiene listas
271         con ncs, como en el diccionario nver'''
272         f=False
273         cont=0
274         while f==False:
275             for i in lista:
276                 if ncomun in i or remove_accents(ncomun.lower()) in\
277                     remove_accents_lista(i): #si coincide con o sin tilde
278                     f=True
279                 cont+=1
280             if cont==len(lista):
281                 break
282         return f
283     nv=[]
284     gens=[]
285     genero=""
286     #buscador de especies y generos
287     for i in llaves_a_lista(ordenar_dicc_keys(ncient)):
288         if ncomun.lower()==i.lower():
289             print("Especie:",n_cientificos[i]) #si encuentra una especie,
290                                                #la devuelve.
291             print(" ")
292             print("Nombres vernáculos:")
293             print("     Castellano:",end=' ')
294             print(*nver[i][0], sep=', ')
295             print(" ")
296             print("     Asturiano:",end=' ')
297             print(*nver[i][1], sep=', ')
298             print(" ")
299             print("Clase   -",clases[i])
300             print("Orden   -",ordenes[i])
301             print("Familia -",familias[i])
302             print(" ")
303             print("Procendencia:", procedencia[i])
304             print(" ")
305             if i in catalogo_ast:
306                 print("Presente en el Catálogo Regionalde Flora Amenazada de Asturias:")
307                 print("    Categoría:", catalogo_ast[i])
308                 print(" ")
309             if i in uicn:
```

```
                print("Presente en la Lista Roja de Especies Amenazadas de la UICN:")
                print("    Categoría:", uicn[i])
                print(" ")
            if invasora[i]=="si":
                print("- Invasora!")
            if endemicos[i]=="True":
                print("- Endémica!")

            print("")
            print("Altitud (m)")
            print("     Compatible :",cotas_umbral[i][0],
                  "-",cotas_umbral[i][1])
            print("     Optimo     :",cotas_optimo[i][0],
                  "-",cotas_optimo[i][1])
            print("")
            print("PMA (mm)")
            print("     Compatible :",precipitacion_umbral[i][0],
                  "-",precipitacion_umbral[i][1])
            print("     Optimo     :",precipitacion_optima[i][0],
                  "-",precipitacion_optima[i][1])
            print("")
            print("Precipitación estival (mm)")
            print("     Compatible :",precipitacion_estival[i][0],
                  "-",precipitacion_estival[i][1])
            print("")
            print("TMA (žC)")
            print("     Compatible :",tma_umbral[i][0], "-",tma_umbral[i][1])
            print("     Optimo     :",tma_optimo[i][0], "-",tma_optimo[i][1])
            print("")
            print("TM Mes Calido (žC)")
            print("     Compatible :",tm_calido[i][0], "-",tm_calido[i][1])
            print("")
            print("TM Mes Frio (žC)")
            print("     Compatible :",tmf_umbral[i][0], "-",tmf_umbral[i][1])
            print("     Optimo     :",tmf_optimo[i][0], "-",tmf_optimo[i][1])
            print("")
            print("Sensibilidades:")
            print("     Helada tardia :", sensible_ht[i])
            print("     Encharcamiento:", sensible_encharcamiento[i])
            print("")
            print("Profundidad minima (cm):", profundidades[i])
            print("")
            print("pH del suelo")
            print("     Min :", ph[i][0])
```

```
354             print("     Max::", ph[i][1])
355 
356             return
357         elif ncomun.lower()==cient_lista(i)[0].lower():
358             gens.append(cient_lista(i)[1])
359             genero=cient_lista(i)[0]
360             nombre_genero=cient_lista(i)[0]
361 
362     if ncomun.upper() in uicn.values(): #UICN
363         print("Categoría UICN, especies:")
364 
365         for i in ordenar_dicc_keys(uicn):
366             if uicn[i].lower()==ncomun.lower():
367                 print("  -",i)
368         return
369 
370     if ncomun.upper() in catalogo_ast.values(): #CATALOGO AST
371         print("Categoría Catálogo Regional de Flora Amenazada de Asturias , especies:")
372 
373         for i in ordenar_dicc_keys(catalogo_ast):
374             if catalogo_ast[i].lower()==ncomun.lower():
375                 print("  -",i)
376         return
377 
378     if ncomun.lower() in ["invasora","invasoras","especies invasoras"]:
379         print(contar_valores_dicc(invasora,"si"),"Especies invasoras: ")
380         for i in ordenar_dicc_keys(invasora):
381             if invasora[i]=="si":
382                 print("  -",i)
383         return
384 
385     if remove_accents(ncomun.lower()) in ["autoctona","autoctonas","especies autoctonas"]:
386         print(contar_valores_dicc(procedencia,"autóctona"),"especies autóctonas: ")
387         for i in ordenar_dicc_keys(procedencia):
388             if remove_accents(procedencia[i])=="autoctona":
389                 print("  -",i)
390         return
391 
392     if remove_accents(ncomun.lower()) in ["aloctona","aloctonas"]:
393         print(contar_valores_dicc(procedencia,"alóctona"),"Especies alóctonas: ")
394         for i in ordenar_dicc_keys(procedencia):
395             if remove_accents(procedencia[i])=="aloctona":
396                 print("  -",i)
397         return
```

```
398     if remove_accents(ncomun.lower()) in ["generos","gens"]:
399         print(len(set(generos.values()))-1, "géneros en la BBDD:")
400         for i in sorted(set((generos).values())):
401             if i!="error":
402                 print("  -", i)
403         return
404 
405     if remove_accents(ncomun.lower()) in ["clase","clases"]:
406         print(len(set(clases.values()))-1, "clases en la BBDD:")
407         for i in sorted(set((clases).values())):
408             if i!="error":
409                 print("  -", i)
410         return
411 
412     if remove_accents(ncomun.lower()) in ["familia","familias"]:
413         print(len(set(familias.values()))-1, "familias en la BBDD:")
414         for i in sorted(set((familias).values())):
415             if i!="error":
416                 print("  -", i)
417         return
418     if remove_accents(ncomun.lower()) in ["orden","ordenes"]:
419         print(len(set(ordenes.values()))-1, "órdenes en la BBDD:")
420         for i in sorted(set((ordenes).values())):
421             if i!="error":
422                 print("  -", i)
423         return
424     if remove_accents(ncomun.lower()) in ["especies","sspp","especie"]:
425         print(len(set(especies.keys()))-1, "especies en la BBDD:")
426         for i in sorted(set((especies).keys())):
427             if i!="error":
428                 print("  -", i)
429         return
430     if genero!="":
431         print("Genero:",genero)
432         print(" ")
433 
434 
435     for i in nver:
436         if buscar_lista_nv(ncomun.lower(),nver[i])==True:
437             nv.append(i)
438     #if len(nv)==0:
439     #     nv=False
440     result=[gens,nv] #trabaja con generos y nombres vernáculos
441     #print(result)
```

```
442     for i in result:
443         if genero!="" and result[0]!="":
444             if i!=False:
445                 #if i==nv:
446                 print("Especie(s):", end = ' ')
447                 cnt=1
448                 for m in gens:
449                     if cnt==1:
450                         print(nombre_genero, m)
451                     else:
452                         print("           ",nombre_genero, m)
453                     cnt+=1
454                 break
455         else:
456             print("Especies:", end=' ')
457             sp_count=1
458             for especie in sorted(nv):
459                 if sp_count==1:
460                     print(especie)
461                 else:
462                     print("         ",especie)
463                 sp_count+=1
464             break
465 
466 
467     if not any(result): #si no hay resultados (todos los elementos de la
468                         #lista son  falsos)
469         print("No hay resultados.")
470 
471 def sistematica():
472     n_cientifico="Scientific_Name"
473     endemico="Endemic"
474     genero="Genus"
475     especie="Species"
476     clase="Class"
477     orden="Order"
478     familia="Family"
479     n_cientificos=get_columna(n_cientifico)
480     endemicos=get_columna(endemico)
481     generos=get_columna(genero)
482     especies=get_columna(especie)
483     clases=get_columna(clase)
484     ordenes=get_columna(orden)
485     familias=get_columna(familia)
486     return n_cientificos,endemicos,generos,especies,clases,ordenes,familias
```

```
487
488 def libro_catalogo_cires():
489     invasora="invasora"
490     procedencia="procedencia"
491     inv=get_columna(invasora)
492     proc=get_columna(procedencia)
493     return inv,proc
494
495 def amenazas():
496     uicn="uicn"
497     cat="catalogo_ast"
498     uic=get_columna(uicn)
499     cata=get_columna(cat)
500     uic = {k:v for k,v in uic.items() if v != 'error'}
501     cata = {k:v for k,v in cata.items() if v != 'error'}
502     cata = {k:v for k,v in cata.items() if v != 'no'}
503     return uic, cata
504
505 def precipitaciones():
506
507     pma_min="PMA_min"           #precipitación media minima
508     pma_max="PMA_max"           #precipitación media máxima
509     op_PMA_min="op_PMA_min"     #optima minima
510     op_PMA_max="op_PMA_max"
511     PV_min="PV_min"             #precipitación estival min/max
512     PV_max="PV_max"
513
514     posiciones=get_posiciones([pma_min,pma_max,op_PMA_min,op_PMA_max,PV_min,
515                                PV_max])
516     precipitacion_umbral={}
517     precipitacion_optima={}
518     precipitacion_estival={}
519     for i in l:
520         umbral=[]
521         optimo=[]
522         estival=[]
523         lista=[umbral,optimo,estival]
524         lista_dicc=[precipitacion_umbral,precipitacion_optima,
525                     precipitacion_estival]
526         numero_pos=1
527         for n in range(len(lista)):
528             try:
529                 lista[n].append(float(i[posiciones[numero_pos]]))
530                 numero_pos+=1
531                 suma=False
```

```
                lista[n].append(float(i[posiciones[numero_pos]]))
                numero_pos+=1
                suma=True
                if i[0] not in lista_dicc[n]:
                    lista_dicc[n][i[0]]=lista[n]
            except:
                lista_dicc[n][i[0]]="error"
                print("")
                print("Error en lectura de precipitaciones de", i[0], "en",
                      header[posiciones[numero_pos]])
                print("Comprobar en fila", ncient[i[0]],
                      get_column_letter(posiciones[numero_pos]+1))
                if suma==False:
                    numero_pos+=1
                else:
                    numero_pos+=2
    return precipitacion_umbral, precipitacion_optima, precipitacion_estival

def temperaturas():
    tma_min="TMA min (žC)"
    tma_max="TMA max (žC)"
    op_tma_min="op_TMA min (žC)"
    op_tma_max="op_TMA max (žC)"

    tm_calido_min="TMC min (žC)"
    tm_calido_max="TMC max (žC)"

    tm_frio_min="TMF min(žC)"
    tm_frio_max="TMF max (žC)"
    op_frio_min="op_TMF min(žC)"
    op_frio_max="op_TMF max (žC)"

    sensible_helada_tard="Sen Hel T"

    posiciones=get_posiciones([tma_min,tma_max,op_tma_min,op_tma_max,
                               tm_calido_min,tm_calido_max,
                               tm_frio_min,tm_frio_max,op_frio_min,
                               op_frio_max, sensible_helada_tard])
    (tma_umbral,tma_optimo,tm_calido,tmf_umbral,tmf_optimo,
     sensible_helada_tardia)=({},{},{},{},{},{})

    for i in l:
        _tma_umbral=[]
        _tma_optimo=[]
```

```
576         _tm_calido=[] #solo 2 valores
577         _tmf_umbral=[]
578         _tmf_optimo=[]
579         lista_prec=[_tma_umbral,_tma_optimo,_tm_calido,_tmf_umbral,_tmf_optimo]
580         lista_dicc=[tma_umbral,tma_optimo,tm_calido,tmf_umbral,tmf_optimo,
581                     sensible_helada_tardia]
582         numero_pos=1
583         for n in range(len(lista_prec)):
584             try:
585                 lista_prec[n].append(float(i[posiciones[numero_pos]]))
586                 numero_pos+=1
587                 suma=False
588                 lista_prec[n].append(float(i[posiciones[numero_pos]]))
589                 numero_pos+=1
590                 suma=True
591                 if i[0] not in lista_dicc[n]:
592                     lista_dicc[n][i[0]]=lista_prec[n]
593             except:
594                 lista_dicc[n][i[0]]="error"
595                 print("")
596                 print("Error en lectura de temperaturas de", i[0], "en",
597                       header[posiciones[numero_pos]])
598                 print("Comprobar en fila", ncient[i[0]],
599                       get_column_letter(posiciones[numero_pos]+1))
600                 if suma==False:
601                     numero_pos+=1
602                 else:
603                     numero_pos+=2
604 
605     nombre="Sen Hel T"
606     en=get_columna(nombre)
607     fila=1
608     for i in en: #comprueba que no haya valores distintos a "si" y "no"
609         fila+=1
610         if en[i]!="si" and en[i]!="no":
611             print("")
612             print("Error de lectura de", i,"en", nombre)
613             print("Comprobar en fila", fila,
614                   get_column_letter(get_posiciones([nombre])[1]+1))
615             en[i]="error"
616 
617     return tma_umbral, tma_optimo, tm_calido, tmf_umbral, tmf_optimo, en
618 
619 def encharcamiento():
```

```
620         nombre="Sen Encharcamiento"
621         en=get_columna(nombre)
622         fila=1
623         for i in en: #comprueba que no haya valores distintos a "si" y "no"
624             fila+=1
625             if en[i]!="si" and en[i]!="no":
626                 print("")
627                 print("Error de lectura de", i,"en", nombre)
628                 print("Comprobar en fila", fila,
629                       get_column_letter(get_posiciones([nombre])[1]+1))
630                 en[i]="error"
631         return en
632 
633     def profundidades():
634         nombre="Prof_min (cm)"
635         prof=get_columna(nombre)
636         fila=1
637         for i in prof:
638             fila+=1
639             try:
640                 prof[i]=float(prof[i])
641                 if prof[i]<0 or prof[i]>200:
642                     prof[i]="error"
643                     print("")
644                     print("Error de lectura de", i,"en", nombre)
645                     print("Comprobar en fila", fila,
646                           get_column_letter(get_posiciones([nombre])[1]+1))
647 
648             except:
649                 print("")
650                 print("Error de lectura de", i,"en", nombre)
651                 print("Comprobar en fila",
652                       fila, get_column_letter(get_posiciones([nombre])[1]+1))
653                 prof[i]="error"
654         return prof
655 
656     def ph():
657         _ph={}
658         ph_min="pH_min"
659         ph_max="pH_max"
660         posiciones=get_posiciones([ph_min,ph_max])
661         for i in l:
662             ph=[]
663             try:
664                 a=ph_min
```

```
665             ph.append(float(i[posiciones[1]]))
666             a=ph_max
667             ph.append(float(i[posiciones[2]]))
668             if i[0] not in _ph:
669                 _ph[i[0]]=ph
670         except:
671             _ph[i[0]]="error"
672             print("")
673             print("Error en lectura de pH de", i[0], "en",
674                   header[get_posiciones([a])[1]])
675             print("Comprobar en fila", ncient[i[0]],
676                   get_column_letter(get_posiciones([a])[1]+1))
677     return _ph
678
679
680
681 def comprobar_pma(valor):
682     comp_prec={}
683     for i in precipitacion_umbral:
684         a=comprobar_4valor(valor, precipitacion_umbral[i][0],
685                            precipitacion_umbral[i][1],
686                            precipitacion_optima[i][0],
687                            precipitacion_optima[i][1] )
688         comp_prec[i]=a
689     return comp_prec
690
691 def comprobar_precest(valor): #precipitacion estival
692     comp_est={}
693     for i in precipitacion_estival:
694         a=comprobar_2valor(valor, precipitacion_estival[i][0],
695                            precipitacion_estival[i][1])
696         comp_est[i]=a
697     return comp_est
698
699 def comprobar_tma(valor):
700     comp_tma={}
701     for i in tma_umbral:
702         a=comprobar_4valor(valor, tma_umbral[i][0], tma_umbral[i][1],
703                            tma_optimo[i][0], tma_optimo[i][1])
704         comp_tma[i]=a
705     return comp_tma
706
707 def comprobar_tm_calido(valor):
708     comp_tmc={}
```

```
709     for i in tm_calido:
710         a=comprobar_2valor(valor, tm_calido[i][0], tm_calido[i][1])
711         comp_tmc[i]=a
712     return comp_tmc
713 
714 def comprobar_tm_frio(valor):
715     comp_tmf={}
716     for i in tmf_umbral:
717         a=comprobar_4valor(valor, tmf_umbral[i][0], tmf_umbral[i][0],
718                            tmf_optimo[i][0],  tmf_optimo[i][1])
719         comp_tmf[i]=a
720     return comp_tmf
721 
722 def comprobar_sen_ht(valor): #si   // helada tardía
723     comp_ht={}
724     temporal=comp_si_no(comp_ht, sensible_ht, valor)
725     for i in temporal:
726         if temporal[i]=="incompatible":
727             temporal[i]="apto"
728         if temporal[i]=="compatible":
729             temporal[i]="incompatible"
730     return temporal # doble
731                     #negacion de "sensible a heladas",
732                     #lo mismo con encgarcamiento"
733 
734 def comp_si_no(comp_dict, dict_, valor):
735     '''Funcion intermedia para usar dentro de funciones de comprobar
736     si/no (comprobar_sen_ht)'''
737     for i in dict_:
738         if dict_[i]=="error":
739             comp_dict[i]="error"
740         elif valor=="si":
741             if dict_[i]==True or dict_[i]=="si":
742                 comp_dict[i]=["incompatible"]
743             else:
744                 comp_dict[i]=["apto"]
745         else:
746             comp_dict[i]=["apto"]
747     return comp_dict
748 
749 def comprobar_sen_encharcamiento(valor): #si  // encharcamiento
750     comp_encharcamiento={}
751     temporal = comp_si_no(comp_encharcamiento, sensible_encharcamiento, valor)
752     for i in temporal:
753         if temporal[i]=="incompatible":
```

```
754                 temporal[i]="apto"
755             if temporal[i]=="compatible":
756                 temporal[i]="incompatible"
757         return temporal
758
759     def comprobar_profundidades(valor):
760         comp_prof={}
761         for i in profundidades:
762             try:
763                 if profundidades[i]>valor:
764                     comp_prof[i]=["incompatible"]
765                 else:
766                     comp_prof[i]=["compatible"]
767             except:
768                 comp_prof[i]=["error"]
769         return comp_prof
770
771
772     def comprobar_ph(valor):
773         comp_ph={}
774         for i in ph:
775             a=comprobar_2valor(valor, ph[i][0], ph[i][1])
776             comp_ph[i]=a
777         return comp_ph
778
779
780
781
782
783
784
785     ######## CONSULTAS ########
786     def idoneidad_RCO(l_inf,l_sup,op_inf,op_sup,valor):
787         try:
788             if valor>=op_inf and valor<=op_sup:
789                 return 100
790             elif valor<op_inf:
791                 return (90*(valor-l_inf)/(op_inf-l_inf))+10
792             else:
793                 return (90*(l_sup-valor)/(l_sup-op_sup))+10
794         except:
795                 return idoneidad_RC(l_inf, l_sup, valor)
796
797     def idoneidad_RC(l_inf,l_sup,valor): #optimo es el 10% del rango central
```

```
798         try:
799             op_inf=0.45*(l_sup-l_inf)
800             op_sup=0.55*(l_sup-l_inf)
801             return idoneidad_RCO(l_inf,l_sup,op_inf,op_sup,valor)
802         except:
803             return 'error'
804 
805     def consulta():
806         arboles=copiar_dic(ncient)
807         compatible={}
808         no_compatible={}
809 
810         lista_consulta=import_consulta()
811         def vaciar_consulta():
812             lista_vaciada=[]
813             for i in lista_consulta:
814                 if i[1]!='':
815                     lista_vaciada.append(i)
816             return lista_vaciada
817 
818 
819         def ponderacion():
820             pesos={}
821             for i in vaciar_consulta():
822                 pesos[i[0]]=float(i[2])
823             return pesos
824 
825         def comprobar_optimo(diccionario_comprobacion, arbol):
826             if diccionario_comprobacion[arbol][0]=='apto' or\
827                 diccionario_comprobacion[arbol][0]=='optimo':
828                 return True
829             elif diccionario_comprobacion[arbol][0]=='compatible':
830                 return True
831             else:
832                 return False
833         def añadir_a_compatibilidad(arbol,variable,diccionario):
834             '''variable es cadena; diccionario es compatible o no_compatible'''
835             if arbol not in diccionario:
836                 diccionario[arbol]=[variable]
837             else:
838                 diccionario[arbol].append(variable)
839         def añadir_definitivo(variable, cadena_variable):
840             for arbol in arboles:
841                     if comprobar_optimo(variable,arbol):
842                         añadir_a_compatibilidad(arbol, cadena_variable,
```

```
                                        compatible)
            else:
                añadir_a_compatibilidad(arbol, cadena_variable,
                                        no_compatible)


def consulta_estricta():

    for i in vaciar_consulta(): #nombres del CSV

        if i[0]=="Cota":
            cotas=comprobar_cota(float(i[1]))
            añadir_definitivo(cotas, 'Cota')

        elif i[0]=="PMA":
            pmas=comprobar_pma(float(i[1]))
            añadir_definitivo(pmas, 'PMA')

        elif i[0]=="P Estival":
            pest=comprobar_precest(float(i[1]))
            añadir_definitivo(pest, 'P Estival')

        elif i[0]=="TMA":
            tmas=comprobar_tma(float(i[1]))
            añadir_definitivo(tmas, 'TMA')

        elif i[0]=="TM (calido)":
            cal=comprobar_tm_calido(float(i[1]))
            añadir_definitivo(cal, 'TM (calido)')

        elif i[0]=="TM (frío)":
            fri=comprobar_tm_frio(float(i[1]))
            añadir_definitivo(fri, 'TM (frío)')

        elif i[0]=="Helada Tardia":
            ht=comprobar_sen_ht(i[1])
            añadir_definitivo(ht, 'Helada Tardia')

        elif i[0]=="Encharcamiento":
            en=comprobar_sen_encharcamiento(i[1])
            añadir_definitivo(en, 'Encharcamiento')

        elif i[0]=="Profundidad":
            prof=comprobar_profundidades(float(i[1]))
```

```
887                 añadir_definitivo(prof, 'Profundidad')
888 
889             elif i[0]=="pH":
890                 phs=comprobar_ph(float(i[1]))
891                 añadir_definitivo(phs, 'pH')
892 
893         return compatible,no_compatible
894 
895     (compatible,incompatible)=consulta_estricta()
896 
897 
898     print("")
899     print(5*"########### ")
900     print("")
901     for parametro in vaciar_consulta():
902         print (parametro[0], ":", parametro[1], parametro[3])
903     print("")
904     print("Resultados de compatibilidad según consulta:")
905 
906     compatibilidad={}
907     for especie in compatible:
908         puntuacion=0
909         for parametro in compatible[especie]:
910             puntuacion+=ponderacion()[parametro]
911         compatibilidad[especie]=round\
912             (puntuacion/sum(ponderacion().values())*100,2)
913 
914     compatible_100={}
915     no_compatible={}
916     for i in compatibilidad:
917         if compatibilidad[i]==100:
918             compatible_100[i]=compatibilidad[i]
919         else:
920             no_compatible[i]=compatibilidad[i]
921 
922     print("")
923     if len(compatible_100)>0:
924         idoneidades={}
925         ido_media=0
926         c=0
927         for especie in ordenar_dicc_keys(compatible_100):
928             for p in vaciar_consulta(): #comprobar idoneidad haciendo la media
929                 try:
930                     if p[0]=="PMA":
```

```
                ido_media=ido_media+idoneidad_RCO\
                    (precipitacion_umbral[especie][0],
                     precipitacion_umbral[especie][1],
                              precipitacion_optima[especie][0],
                              precipitacion_optima[especie][1],
                              float(p[1]))
                c+=1
            elif p[0]=="Cota":
                ido_media=ido_media+idoneidad_RCO\
                    (cotas_umbral[especie][0],
                     cotas_umbral[especie][1],
                              cotas_optimo[especie][0],
                              cotas_optimo[especie][1],
                              float(p[1]))
                c+=1
            elif p[0]=="P estival":
                ido_media=ido_media+idoneidad_RC\
                    (precipitacion_estival[especie][0],
                                              precipitacion_estival\
                                                  [especie][1],
                                              float(p[1]))
                c+=1
            elif p[0]=="TMA":
                ido_media=ido_media+idoneidad_RCO\
                    (tma_umbral[especie][0], tma_umbral[especie][1],
                              tma_optimo[especie][0],
                              tma_optimo[especie][1],
                              float(p[1]))
                c+=1
            elif p[0]=="TM (calido)":
                ido_media=ido_media+idoneidad_RC\
                    (tm_calido[especie][0], tm_calido[especie][1],
                                              float(p[1]))
                c+=1
            elif p[0]=="TM (frío)":
                ido_media=ido_media+idoneidad_RCO\
                    (tmf_umbral[especie][0], tmf_umbral[especie][1],
                              tmf_optimo[especie][0],
                              tmf_optimo[especie][1],
                              float(p[1]))
                c+=1
            elif p[0]=="Helada Tardia":
                ido_media+=100
                c+=1
```

```
                    elif p[0]=="Encharcamiento":
                        ido_media+=100
                        c+=1
                    elif p[0]=="Profundidad":
                        ido_media+=100
                        c+=1
                    elif p[0]=="pH":
                        ido_media=ido_media+idoneidad_RC(ph[especie][0],
                                                         ph[especie][1],
                                                         float(p[1]))
                        c+=1

                except:
                    print("errrorrrrr")
                    idoneidades[especie]=0
                    ido_media=0
                    c=0
            try:
                idoneidades[especie]=round(ido_media/c,2)
                ido_media=0
                c=0
            except:
                print("errrorrrrr")
                idoneidades[especie]=0
                ido_media=0
                c=0
        print(len(idoneidades), "especie(s) compatibles:")
        for i in (ordenar_dicc_valores(idoneidades)):
            if idoneidades[i]<=0:
                print("")
                print(" -", i,"- datos insuficientes en BBDD para calcular idoneidad")
            else:
                print("")
                print(" -", i,"-",idoneidades[i], "% de idoneidad")
    else:
        print("Ninguna especie compatible.")



    if len(no_compatible)>0:
        print("")
        print("")
        print(len(no_compatible), "especie(s) no compatibles:")
        contador=0
```

```
        for i in (ordenar_dicc_valores(no_compatible)):
            if contador==3:
               print(" ")
               print(" ")
               print("                                                    [enter] - Terminar")
               print("                                                    [t]     - Mostrar todos los resultad
               resp=input("                                                    ")
               if resp=="":
                   return
               elif resp=="t":
                   contador=-500
               else:
                   contador=0
            print("")
            print(" ",i, ordenar_dicc_valores(no_compatible)[i],
                  "% de parámetros compatibles" )
            if len(compatible[i])<len(vaciar_consulta()):
                print("     Parámetros incompatibles:")
                for parametro_no_compatible in incompatible[i]:
                    print("        -",parametro_no_compatible)
            contador+=1

print("Leyendo BBDD")

csv_=import_csv()    #contiene todo el .csv
l=get_list()         #CSV ahora está contenido en este parámetro l
header=get_header()  #Lista nombres de los parámetros (1a fila del CSV)
ncient=get_cient()   #diccionario de los nombres científicos
nver=get_nver()      #diccionario de nombres vernáculos: nombre_cientifico:
                        #[lista castellano, lista ]

(cotas_umbral,cotas_optimo)=cotas() #diccionarios de cotas umbral y óptimas
(precipitacion_umbral, precipitacion_optima, precipitacion_estival) =precipitaciones() #diccionarios
(tma_umbral, tma_optimo, tm_calido, tmf_umbral, tmf_optimo, sensible_ht)=temperaturas()
sensible_encharcamiento=encharcamiento()
profundidades=profundidades()
ph=ph()

(n_cientificos,endemicos,generos,especies,clases,ordenes,familias)=sistematica()
(invasora, procedencia)=libro_catalogo_cires()
(uicn,catalogo_ast)=amenazas()
del(csv_) #esto genera un error que en realidad no afecta,
           #borro la variable porque luego no es necesaria
```

```
1063
1064  def bienvenida():
1065      print(" ")
1066      print(5*"########### ")
1067      print(" ")
1068      print(" ")
1069      print(" ")
1070      print(" ")
1071      print("""\
1072                          # #### ####
1073                        ### \/#|### |/####
1074                       ##\/#/ \||/##/_/##/_#
1075                     ###  \/###|/ \/ # ###
1076                   ##_\_#\_\## | #/###_/_####
1077                  ## #### # \ #| /  #### ##/##
1078                   __#_-###`  |{,###--###-~
1079                             \ }{
1080                              }}{
1081                              }}{
1082                         ejm  {{}
1083                        , -=-~{ .-^- _          CF1.0
1084                              `}                Daniel Pfitzer 2022
1085                               { """)
1086
1087  def opciones():
1088      print(" ")
1089      print(" ")
1090      print("                                                  [1] - Búsqueda")
1091      print("                                                  [2] - Consulta")
1092      print(" ")
1093      print(" ")
1094
1095  def contar_valores_dicc(dicc,valor):
1096      cont=0
1097      for i in dicc:
1098          if dicc[i]==valor:
1099              cont+=1
1100      return cont
1101
1102
1103
1104  def contar_nvers():
1105      contador=0
1106      nombres={}
1107      for i in nver:
```

```
1108            for listas in nver[i]:
1109                if len(listas)>0:
1110                    for nombre in listas:
1111                        contador+=1
1112                        #print(contador,nombre)
1113                        if nombre not in nombres:
1114                            nombres[nombre]=1
1115                        else:
1116                            nombres[nombre]+=1
1117        return contador,ordenar_dicc_valores(nombres)
1118
1119
1120    def resumen_clases():
1121        rclases={}
1122        for i in clases:
1123            if clases[i] not in rclases:
1124                rclases[clases[i]]=1
1125            else:
1126                rclases[clases[i]]+=1
1127        return rclases
1128
1129    def resumen_ordenes():
1130        rordenes={}
1131        for i in ordenes:
1132            if ordenes[i] not in rordenes:
1133                rordenes[ordenes[i]]=1
1134            else:
1135                rordenes[ordenes[i]]+=1
1136        return  ordenar_dicc_valores(rordenes)
1137
1138    def resumen_familias():
1139        rordenes={}
1140        for i in familias:
1141            if familias[i] not in rordenes:
1142                rordenes[familias[i]]=1
1143            else:
1144                rordenes[familias[i]]+=1
1145        return  ordenar_dicc_valores(rordenes)
1146
1147
1148    def resumen_generos():
1149        rordenes={}
1150        for i in generos:
1151            if generos[i] not in rordenes:
```

```
            rordenes[generos[i]]=1
        else:
            rordenes[generos[i]]+=1
    return  ordenar_dicc_valores(rordenes)


(contador_nvers, nvers_cont_dic)=contar_nvers()

# ================================================================================
# for i in ordenar_dicc_keys(resumen_generos()):
#     print("\item \\textit{", i, "} -",resumen_generos()[i])
# ================================================================================


# ================================================================================
# with open('readme.txt', 'w') as f: ####genrador latex nombres vernaculos
#
#
#
#     contador=0
#     for i in nver:
#             f.write("\\begin{center} "+"\\textbf{\\textit{"+i+"}}\\end{center}")
#             f.write('\n')
#             for m in nver[i]:
#                 try:
#                     if m[0]:
#                         f.write("\\begin{itemize}\\setlength\\itemsep{0.1mm}")
#                         f.write('\n')
#                         f.write("\\item Idioma")
#                         f.write('\n')
#                         f.write("\\begin{itemize}\\setlength\\itemsep{0.1mm}")
#                         f.write('\n')
#                         for nombre in m:
#                             f.write("\\item "+nombre)
#                             f.write('\n')
#                             contador+=1
#                         f.write("\\end{itemize}")
#                         f.write('\n')
#                         f.write("\\end{itemize}")
#                         f.write('\n')
#                 except:
#                     f.write("")
#                     f.write('\n')
# ================================================================================
```

C.3. Conversor a nombres vernáculos

```
s=input("in: ")
l=[]
for i in s:
    if i==",":
        l.append("_")
    else:
        l.append(i.lower())
d=[]
c=True
for i in range(len(l)):
    if l[i]=="_" and i==len(l):
        break
    if l[i]=="_" and l[i+1]==" ":
        d.append("_")
        c=False
    elif c==False:
        c=True
    else:
        d.append(l[i])
f=""
for i in d:
    if i in "()1234567890.":
        f=f
    else:
        f=f+i
g=""
for i in range(len(f)):
    if f[i]==" " and f[i+1]=="_":
        g=g
    else:
        g=g+f[i]
g=g+"_"
print("-")
print(g)
```

C.4. Simulaciones para validación

Código empleado para realizar las simulaciones en 7 Validación del programa.

```
import funciones as f
cc=0
def contador(x):
    sp3={}
    if type(x[es])==list and x[es][0]!="incompatible":
        if es not in sp3:
            sp3[es]=1
            print(cc, "simulaciones")
        else:
            sp3[es]+=1

            print(cc, "simulaciones")
    else:

        print(cc, "simulaciones")
    return sp3

especies={}

for i in range (105,2405,5):
    for es in f.ncient:
        l1=contador(f.comprobar_cota(i))
        l2=contador(f.comprobar_pma(861))
        l3=contador(f.comprobar_precest(136))
        l4=contador(f.comprobar_tma(8.3))
        l5=contador(f.comprobar_tm_frio(2))
        l6=contador(f.comprobar_tm_calido(17.7))
        if es in (l1 and l2 and l3 and l4 and l5 and l6):
            if es not in especies:
                especies[es]=1
            else:
                especies[es]+=1
    cc+=1

print(" ")
cd=0
for i in f.ordenar_dicc_valores(especies):
    print(i, especies[i],"de", cc, especies[i]/cc*100, "%")
    cd+1
    if cd>5:
        break
```

C.5. Análisis de la BBDD

Código empleado para estudiar la variación de los parámetros a lo largo de las distintas especies en la BBDD.Los resultados se pueden ver en la sección 4.1.2: Requerimientos climáticos y en el anexo B.2: Frecuencias de los parámetros (CC).

```
import funciones as f
import numpy as np
from matplotlib.ticker import MaxNLocator


def estudio_cotas(v):
    cotas={}
    for i in v:
        #print(i)
        for n in np.arange(0, 12, 0.5):

            if v[i]!="error":
                #print(n)
                if n>=v[i][0] and n<=v[i][1]:
                    #print("ta dentro!")
                    if n in cotas:
                        cotas[n]+=1
                    else:
                        cotas[n]=1
                else:
                    #print("nope")
                    if n not in cotas:
                        cotas[n]=0

    return (cotas)


D=estudio_cotas(f.ph)

import matplotlib.pyplot as plt

plt.bar(range(len(D)), list(D.values()), align='center')
plt.xticks(range(len(D)), list(D.keys()))
plt.xticks(rotation=45)
plt.locator_params(nbins=10)
```

```
# # for python 2.x:
# plt.bar(range(len(D)), D.values(), align='center')  # python 2.x
# plt.xticks(range(len(D)), D.keys())  # in python 2.x

plt.show()


x=[]
y=[]
lista=[]
for i in D:
    x.append(i)
    y.append(int(D[i]))
    for n in range(D[i]):
        lista.append(i)

a = np.array(lista)
p50=np.percentile(a, 50)
p25 = np.percentile(a, 25) # return 50th percentile, e.g median.
p75 = np.percentile(a, 75)
print("Percentil 25:", p25)
print("Percentil 50:", p50)
print("Percentil 25:", p75)

ax = plt.figure().gca()
ax.yaxis.set_major_locator(MaxNLocator(integer=True))
plt.locator_params(nbins=10)
plt.axvline(x=p25, ymin=0,ymax=0.91, color='red', linestyle='dotted', linewidth=1)
plt.axvline(x=p75, ymin=0,ymax=0.65, color='red', linestyle='dotted', linewidth=1,label= 'Percentiles 25
plt.legend(loc = 'upper right')
plt.plot(x, y)
plt.grid()
plt.savefig("tma.svg")
plt.xlabel('tma', fontsize=12)
plt.ylabel('Registros', fontsize=12)


plt.show()
```

Referencias

AA, A. (2016). Catálogo Regional de Flora Amenazada de Asturias. *Normativa sobre Flora Amenazada de Asturias. Especies catalogadas en Peligro de Extinción en Asturias*.

Alonso, R., López, E., Gómez, V., Sánchez-Palomares, O., Ruiz-Peinado, R. & Montero, G. (2022). ModERFoRest. Consultado: 3/04/2022. https://www.inia.es/serviciosyrecursos/recursosinformaticos/modernforest/Paginas/ModERFoRest.aspx

Anthos. (2011). Sistema de Información sobre las plantas de españa. Consultado: 3/04/2022. http://www.anthos.es/

Bravo, A. & Montero, G. (2008). Descripción de los caracteres culturales de las principales especies forestales de España. *INIA-Ministerio de Educación y Ciencia, Madrid*, *1178*.

Castaño-Santamara, J., Barrio-Anta, M. & Álvarez-Álvarez, P. (2013). Potential above ground biomass production and total tree carbon sequestration in the major forest species in NW Spain. *International Forestry Review*, *15*(3), 273-289.

Cisneros. (2002). *Autoecología del cerezo de monte (Prunus avium L.) en Castilla y León* (Tesis doctoral). Universidad Politécnica de Madrid,

de la Torre, J. R., Viñas, J. I. G., del Castillo y de Navascués, J. R. & Cardo, Ó. G. (2006). *Flora mayor*. Organismo Autónomo Parques Nacionales.

Díaz-González, T. (2015). Guía para la identificación de los bosques, matorrales y series de vegetación (vegetación potencial) de Asturias (España) mediante bioindicadores fitocenológicos. *Bol. Cien. Nat. RIDEA*, *53*(9).

Díaz-Maroto, Vila-Lameiro, P. & Diaz-Maroto, M. C. (2006). Autecology of sessile oak (*Quercus petraea*) in the north-west Iberian Peninsula. *Scandinavian Journal of Forest Research*, *21*(6), 458-469.

Elena, Sánchez, F., Sánchez, A. R., Sanz, V. G., Aunós, A., Andray, A. B. & Palomares, O. S. (2001). Autoecología de los hayedos catalanes. *Investigación Agraria: Sistemas y Recursos Forestales*, *10*(1), 21-42. https://oa.upm.es/48479/

Elena Rossello, R., Sanchez Palomares, O. & Carretero Carrero, P. (1985). Physiographic and climatic study of Spanish autochthonous pine woods of *Pinus nigra* Arn. *Comunicaciones-Instituto Nacional de Investigaciones Agrarias. Serie Recursos Naturales (Spain)*.

Fernández Díaz-Formentí, J. (2004). Árboles y arbustos naturales de Asturias. *Cajastur. Oviedo*.

Gandullo, J. M. (1972). Ecología de los pinares españoles. III. *Pinus halepensis* Mill.

Gandullo, J. M., González Alonso, S. & Sánchez Palomares, O. (1974). Ecología de los pinares españoles. IV. *Pinus radiata* D. Don.

Garcia, J. P., Cerrillo, R. N., Nicolás-Peragón, J., Saez, M. P. & Hierro, R. S. (2014). Producción y manejo de semillas y plantas forestales. *Ecosistemas*, *22*(1), 92.

González-Vázquez. (1938). *Fundamentos Naturales de la Selvicultura. Los bosques ibéricos*. Libro Primero. Instituto Forestal de Investigaciones y Experiencias. Valencia.

Gutiérrez, J. M. G. & Palomares, O. S. (1994). *Estaciones ecológicas de los pinares españoles*. Instituto Nacional para la Conservación de la Naturaleza.

Hughes, L. (2000). Biological consequences of global warming: is the signal already apparent? *Trends in ecology & evolution*, *15*(2), 56-61.

Lopez-Senespleda, E., Montero, G., Ruiz-Peinado, R., Alonso Ponce, R., Serrada, R. & Sánchez-Palomares, O. (2018). Cincuenta años de autoecología forestal paramétrica en España. *70*.

López-Senespleda & P, S. (2006). Tipificación ecológico-selvícola de *Quercus faginea* Lamk. en España, Informe convenio INIA-DGB.

Nicolas, A., Gandullo, J. y col. (1969). The ecology of the Spanish Pine forests. II. *Pinus sylvestris*.

Nicolás, A. d. & Gandullo, J. M. (1967). Ecología de los pinares españoles. I. *Pinus pinaster* Ait.

Olaya, V. (2009). Sistemas de información geográfica. *Cuadernos internacionales de tecnología para el desarrollo humano*, (8), 15.

Prieto, J. A. F., Rodríguez, E. C., Sánchez, Á. B., Vázquez, V. M. & Fernández, H. S. N. (2014). *Catálogo de las plantas vasculares del Principado de Asturias*. Jardín Botánico Atlántico.

Ramos, I., Lima, H., Pajarón, S., Romero-Zarco, C., Sáez, L., Pataro, L., Molina-Venegas, R., Rodrguez, M. Á. & Moreno-Saiz, J. C. (2021). Atlas of the vascular flora of the Iberian Peninsula biodiversity hotspot (AFLIBER). Consultado: 7/04/2022 (J. Lenoir, Ed.). *Global Ecology and Biogeography*, *30*(10), 1951-1957. https://doi.org/10.1111/geb.13363

Raybaut, P. (2009). Spyder-documentation.

Roces-Díaz, J. V., Jiménez-Alfaro, B., Chytr, M., Díaz-Varela, E. R. & Álvarez-Álvarez, P. (2018). Glacial refugia and mid-Holocene expansion delineate the current distribution of Castanea sativa in Europe. *Palaeogeography, Palaeoclimatology, Palaeoecology*, *491*, 152-160.

Rubio. (1992). *Estudio ecológico de los castañares de Extremadura* (Tesis doctoral). Universidad Politécnica de Madrid.

Sánchez-Palomares & Carretero. Caracterización de hábitats en los alcornocales andaluces. En: *Irati*. II Congreso Forestal Español. 1997, 575-580.

Sánchez-Palomares, J, G. & S, R. (2006). Convenio entre el INIA y la Dirección General de Biodiversidad (DGB) parala realización de trabajos en materia de investigación de tipificaciones ecológico-selvícolas: *Quercus pyrenaica* Willd. Informe final.

Sánchez-Palomares, López-Senespleda, S, R., A, V., S, S. & J, G. (2006). Tipificación ecológico selvícola de la encina (*Quercus ilex*) en su área de distribución meseteña (Castilla y León, Castilla La Mancha, Madrid y Extremadura). Informe final.

Sánchez-Palomares, O., López-Senespleda, E., Calama, R., Ruiz-Peinado, R. & MONTERO, G. (2013). *Autoecología paramétrica de Pinus pinea L. en la España peninsular*. INIA-Instituto Nacional de Investigación y Tecnología Agraria y Alimentaria.

Serrada, R. (1993). *Apuntes de repoblaciones forestales* (inf. téc.).

Trespaderne, F. M. & Echevarría, R. M. (2019). Fundamentos de Programación en python. Consultado: 4/05/2022. https://www2.eii.uva.es/fund_inf/python/notebooks/00_Introduccion/Introduccion.html#

UICN. (2021). The IUCN Red List of Threatened Species. Consultado: 3/04/2022. https://www.iucnredlist.org/

Van Rossum, G. & Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Zazo, J., Calderón, C. & Cornejo, L. (2000). Apuntes y notas de los caracteres culturales y otras caractersticas de interés de algunas frondosas forestales españolas. Escuela Ingeniería Técnica Forestal, Madrid.