
CONAHCYT
CONSEJO NACIONAL DE HUMANIDADES CIENCIAS Y TECNOLOGÍAS
CICY
MÉTODOS de interpolación espacial para el mapeo de la riqueza de especies usando R
José Luis Hernández-Stefanoni ▪ Fernando Tun Dzul
Juan Andrés Mauricio ▪ Luis Ángel Hernández Martínez

D.R. 2023. *Métodos de interpolación espacial para el mapeo de la riqueza de especies usando R*. José Luis Hernández-Stefanoni, Fernando Tun Dzul, Juan Andrés Mauricio, Luis Ángel Hernández Martínez, Centro de Investigación Científica de Yucatán, A.C.

Esta obra debe citarse de la siguiente forma:

Hernández-Stefanoni, J. L., Tun Dzul, F., Andrés-Mauricio, J. & Hernández Martínez, L. Á. (2023). M*étodos de interpolación espacial para el mapeo de la riqueza de especies usando R.* Centro de Investigación Científica de Yucatán, A.C.

Calle 43 #130 x 32 y 34, Col. Chuburná de Hidalgo.
C.P. 97205. Mérida, Yucatán, México.
Tel. (999) 942-8330
Centro Público de Investigación del Conahcyt

ISBN: 978-607-7823-54-4
Primera edición: agosto del 2023.

Coordinador editorial: Julio César Domínguez Orta.
Cuidado editorial: Miguel Gibrán Román Canto.
Diseño editorial: Norma Marmolejo Quintero.
Portada: Juan Andrés Mauricio.

Esta obra fue sometida a un estricto proceso de arbitraje por pares como parte del proceso editorial del Centro de Investigación Científica de Yucatán A.C.

Hecho en México

MÉTODOS de **interpolación espacial** para el mapeo de la riqueza de especies usando R

José Luis Hernández-Stefanoni
Fernando Tun Dzul
Juan Andrés Mauricio
Luis Ángel Hernández Martínez

CONAHCYT
CONSEJO NACIONAL DE HUMANIDADES CIENCIAS Y TECNOLOGÍAS
CICY

Contenido

Presentación

A continuación, se describe la forma como está estructurado el manual; se explica el propósito, la organización y la estructura, así como las convenciones de escritura utilizadas.

Propósito del manual

El propósito de este libro es servir como una guía para las y los usuarios que requieran estimar la distribución espacial de la riqueza de especies o cualquier otra variable de interés en un área de estudio, a partir de datos medidos en sitios ubicados en el espacio donde se conoce el valor del atributo estudiado. Para obtener mapas con una superficie continua del atributo de interés, en este caso, la riqueza de especies o, dicho de otra forma, para estimar la riqueza de especies en sitios no muestreados, se utilizan diferentes métodos de interpolación espacial. Algunos métodos solamente usan los datos con la ubicación del atributo de interés medido en campo, otros combinan datos de campo, información de imágenes de satélite y diferentes procedimientos de interpolación espacial. Para ello se utiliza el *software* libre R, con el propósito de que investigadores e investigadoras, estudiantes, y técnicos y técnicas forestales adscritas a organizaciones no gubernamentales y del gobierno, así como las y los usuarios en general que posean conocimientos básicos y habilidades mínimas de programación para implementar los *scripts* de procesamiento de la información de campo y utilizar información de imágenes de satélite y relacionar estos dos conjuntos de información, puedan obtener mapas con la distribución espacial de la riqueza de especies en sus áreas de interés.

Los métodos de interpolación espacial presentados en este libro comprenden desde aquellos que se aplican a todo el conjunto de datos (métodos globales), pasando por los que se aplican a un subconjunto de datos de manera repetida (métodos locales), hasta aquellos que combinan las técnicas de interpolación con datos auxiliares derivados de las imágenes de satélite. Esta metodología utiliza *scripts* desarrollados en el lenguaje R, en los cuales se describe paso a paso cómo se realiza la estimación de la riqueza de especies con los diferentes métodos de interpolación. Por otro lado, se presenta una guía para validar cada uno de los métodos de interpolación utilizados en este documento.

Este manual debe usarse en conjunto con los datos de campo de una selva o bosque tropical seco de la península de Yucatán, el cual comprende un área de 64 km^2 y se ubica en el municipio de Bacalar, en el estado de Quintana Roo, México. Los datos de esta área se utilizan como ejemplo durante todo el proceso. Además, se requieren los *scripts* empleados en las diferentes etapas de la estimación de la distribución espacial de la riqueza de especies de la selva. Ambos pueden ser descargados libremente como se describe en el Apéndice.

Estructura del manual

Este manual está compuesto por esta presentación, una introducción, nueve capítulos, un apartado de referencias y un apéndice. Siguiendo a esta presentación, en la introducción se señala la importancia de la riqueza de especies de plantas en los bosques tropicales secos, así como la relevancia de la estimación de la distribución espacial de la riqueza para su conservación. Se mencionan algunos de los estudios más recientes sobre el mapeo de la riqueza de especies, tanto a través de datos de campo, como del uso de métodos de interpolación espacial y de procedimientos mixtos que involucran el uso de métodos de interpolación combinado con datos auxiliares como los derivados de sensores remotos. El Capítulo 1 guía al usuario o usuaria en los pasos necesarios para la instalación y configuración del entorno de R y su interfaz RStudio, se describe el procedimiento básico de instalación, su puesta en marcha y una descripción general de los paquetes utilizados en los *scripts* de este manual. En el Capítulo 2 se realiza una introducción general al uso y

manejo del lenguaje R. En el Capítulo 3 se describe el área de estudio, se presentan los procedimientos para el muestreo de campo y el procesamiento de los datos de sensores remotos. En el Capítulo 4 se describe la forma de preparar los datos para la obtención del mapa de coberturas del suelo utilizado para delimitar el área de estudio, así como los objetos de datos necesarios para realizar las interpolaciones. En el Capítulo 5 se presentan los métodos globales de interpolación: polinomios de primer y segundo orden, además de la predicción usando métodos de clasificación. Los métodos locales de interpolación son abordados en el Capítulo 6 y se presenta la estimación utilizando polígonos de Thiessen, así como la interpolación usando distancia inversa y aquella que utiliza Kriging. En el Capítulo 7 se presentan algunos métodos de interpolación espacial que utilizan datos auxiliares derivados de imágenes de satélite, entre los que se incluye: la interpolación dentro de estratos con distancia inversa o con Kriging, regresión con Kriging y Random Forest con Kriging. Por último, en el Capítulo 8 se hace un análisis comparativo de los diferentes métodos de interpolación y se presentan algunas conclusiones de los resultados; mientras que en el Capítulo 9 se comparte una serie de consideraciones finales.

Adicionalmente, en el Apéndice se detallan los pasos para descargar los datos de campo y los *scripts* que se han utilizado para el procesamiento llevado a cabo en este manual, así como la organización de estos.

Convenciones de escritura

Para facilitar la lectura de este documento, el texto en general está escrito con un tipo de letra serif (Cambria) de tamaño 11 pt, las líneas de comando están escritas con una tipografía sans serif (**Barlow**) semibold y sobre un fondo gris. Los resultados arrojados por el *software* R tienen el mismo tipo de letra y tamaño que las líneas de comando (Barlow) en versión *light*, también sobre fondo gris. Por último, las gráficas y figuras que no son resultado de líneas de comando están enumeradas y contienen títulos, mientras que el resto se presenta tal como se obtiene de R.

Introducción

Los bosques tropicales no solo cuentan con una gran diversidad de plantas y animales, sino también suministran varios bienes, incluidos alimentos y materias primas. Además, estos bosques proporcionan diferentes servicios ambientales, entre los que se encuentra la regulación del clima (Portillo-Quintero et al., 2015; Powers et al., 2018). Sin embargo, los bosques tropicales están siendo destruidos por la degradación descontrolada y la conversión a otros usos del suelo. La deforestación es uno de los principales procesos que afectan a estos bosques, siendo la segunda mayor fuente de emisiones de CO_2 a la atmósfera (Le Quéré et al., 2015). El cambio de uso de la tierra contribuye a la pérdida de la diversidad biológica, al daño a los hábitats, a la erosión del suelo y a las perturbaciones infligidas a los ciclos hidrológicos y de nutrientes, entre otros (Swamy et al., 2018). Para preservar la diversidad biológica de estos bosques se requiere información precisa sobre su distribución espacial. Desarrollar y utilizar este tipo de información es, por lo tanto, una parte esencial de los programas de conservación. Por ejemplo, la distribución espacial de especies en un área ayuda en la identificación de zonas de alta prioridad para la conservación (Lechner et al., 2020), que pueden usarse para ubicar reservas, refugios u otras áreas protegidas. Estos mapas son particularmente importantes para los bosques tropicales, que están cada vez más fragmentados, degradados y son eliminados para dar paso a otras formas de uso de la tierra (Fahrig, 2003).

La distribución espacial de la riqueza de especies no puede obtenerse de manera exclusiva a través del uso de técnicas tradicionales de levantamientos de información en campo, debido a las dificultades logísticas y los costos elevados que esto representa. Uno de los procedimientos más comunes para estimar la diversidad de especies a través del espacio, se basa en los valores promedio de la riqueza de especies dentro de clases de vegetación o estratos. En este método, las clases de vegetación se ven como hábitats y la diversidad dentro de esas clases se evalúa a través de muestras de campo. Sin embargo, a pesar de ser fácil de aplicar, generalmente ignora la varia-

bilidad dentro del hábitat, mediante el uso de un único valor medio para predecir la diversidad dentro de cada clase de cobertura terrestre (Hernández-Stefanoni et al., 2012). Además, este enfoque asume independencia de las muestras, es decir, no tiene en cuenta la autocorrelación espacial (Dormann et al., 2007).

Una alternativa a este método es la utilización de procedimientos de interpolación espacial continua**.** Uno de estos procedimientos es el conocido como Kriging, siempre que los datos sean espacialmente dependientes (Webster et al., 2001). Estos métodos se basan en el conocimiento de la estructura espacial del fenómeno, que se obtiene a través de funciones de autocorrelación espacial como los semivariogramas. La estimación de los valores en sitios no muestreados se realiza tomando en cuenta el grado de autocorrelación espacial encontrada en los datos (Webster et al., 2001). Sin embargo, en muchos casos, a pesar de que las muestras de campo están autocorrelacionadas espacialmente, el tamaño de muestra es muy bajo y las muestras están tan dispersas que existe un aumento en la incertidumbre de la estimación de la riqueza de especies cuando se utiliza interpolación con Kriging ordinario (Hernández-Stefanoni et al., 2011).

Se han sugerido procedimientos híbridos, una combinación de Kriging con variables auxiliares para mejorar la precisión de la estimación (Webster et al., 2001). Las y los investigadores han utilizado variables auxiliares que están correlacionadas con la variable principal para mejorar el proceso de estimación. La variable auxiliar se muestrea más intensamente que la variable original porque es más fácil o barata de medir. Esto reduce el costo del muestreo y aumenta la precisión de la predicción de la variable objetivo que tiene menor número de muestras. La percepción remota ofrece un medio económico de obtener una cobertura espacial completa de información ambiental para grandes áreas, a intervalos de tiempo regulares y, por lo tanto, puede ser extremadamente útil para proporcionar variables auxiliares para estimar la riqueza de especies (Gillespie et al., 2008).

Las estimaciones de la riqueza de especies en los bosques han tenido cierto éxito cuando se utilizan datos de sensores remotos. Se utilizan parámetros o indicadores obtenidos de las imágenes de satélite, como son la variabilidad de los valores de reflectancia y los índices de vegetación calculados a partir de bandas de las imágenes, los cuales funcionan como sustitutos de factores asociados con la diversidad de especies (Viedma et al.,

2012). Uno de estos factores es la heterogeneidad ambiental o de hábitat, la cual está relacionada con la distribución y diversidad de especies a través de asociaciones especie-hábitat y disponibilidad de diferentes nichos (Balvanera et al., 2002). Los avances recientes en la investigación de percepción remota han establecido un vínculo entre la variabilidad espectral de las señales de sensores remotos, usados como una aproximación de la heterogeneidad ambiental y la diversidad de especies (Rocchini et al., 2009).

Varios estudios han tratado de combinar datos auxiliares con estimaciones de Kriging para examinar si es posible mejorar la precisión de las predicciones de diferentes variables ecológicas, como la distribución de la abundancia de especies pelágicas pequeñas (Georgakarakos et al., 2008) utilizando datos de percepción remota como variables auxiliares. Del mismo modo, Sales et al. (2007) utilizaron la elevación, el tipo de vegetación y la textura del suelo para mejorar la predicción espacial de la biomasa forestal empleando regresión con Kriging. Además, Hernández-Stefanoni et al. (2012) evaluaron la distribución espacial de la diversidad alpha y beta de especies de árboles mediante la combinación de diferentes métodos de regresión con Kriging y el uso de datos de sensores remotos. Recientemente, Chen et al. (2019) estimaron la biomasa aérea de los bosques utilizando diferentes datos, tales como el radar de apertura sintética (ALOS PALSAR), imágenes ópticas (Sentinel 2), así como modelos de elevación digital y el algoritmo de Random Forest con Kriging.

En este manual se describen las fuentes de información para realizar la estimación de superficies continuas con el uso de interpolación espacial. Por otro lado, se presentan los métodos de interpolación espacial más utilizados y se mencionan sus ventajas y desventajas. Se describen las técnicas de interpolación espacial tomando en consideración desde aquellas que se aplican a todo el conjunto de datos (métodos globales), hasta las que se aplican a un subconjunto de datos de manera repetida (métodos locales). De igual forma, las que proporcionan estimaciones de las variables de interés combinando métodos de interpolación con Kriging y datos auxiliares. Se obtuvieron *scripts* del lenguaje R para implementar los diferentes métodos de interpolación; a pesar de que en este manual se toma como ejemplo la estimación de la riqueza de especies, los métodos y programas de *software* que se presentan son aplicables para mapear cualquier otra variable ecológica y utilizando cualquier otro tipo de imágenes de sensores remotos o de variables auxiliares.

1. Configuración del entorno de R

En este capítulo se pretende guiar al usuario y usuaria a través de los pasos necesarios para instalar y configurar R y la interfaz más popular: RStudio. Se presenta el procedimiento de instalación para el sistema operativo Microsoft Windows, sin embargo, estos pasos pueden trasladarse a otras plataformas con relativa facilidad.

Las ventajas de utilizar R son muchas y se presentarán de forma más detallada en el siguiente capítulo, sin embargo, para usuarias y usuarios inexpertos puede resultar complicado iniciarse en este lenguaje de programación. Por ello, en este apartado se describe el procedimiento básico de instalación, es decir, los primeros pasos para su puesta en marcha y una descripción general de los paquetes que son de interés para cumplir los objetivos de este manual.

1.1 Descargar e instalar R

La forma más fácil de acceder al instalador de R es a través del Comprehensive R Archive Network (CRAN; https://cran.r-project.org/). En esta web será necesario seleccionar el instalador más adecuado para el sistema operativo utilizado. Dado que la interfaz de R no resulta tan amigable para personas no programadoras o con poca experiencia, en esta guía se plantea el procesamiento de datos e ilustración de productos asumiendo la utilización de RStudio.

1.2 Descargar e instalar RStudio

Para instalar RStudio se puede acceder directamente a su página web (https://www.rstudio.com/products/rstudio/download/), donde es posible disponer de diversas versiones de este programa. Para los fines del manual, la versión gratuita de escritorio (RStudio Desktop) resulta adecuada y no se entrará en detalles sobre las otras versiones disponibles. Es importante señalar que la versión más reciente de RStudio (versión 2023.03.1 al momento de la creación de este manual) solo es compatible con sistemas operativos de 64 bits, por lo tanto, para versiones de 32 bits deberá instalarse una versión anterior. En la página de descarga también es posible verificar la versión de R necesaria para utilizar RStudio. Los pasos generales para instalar R y RStudio se resumen a continuación:

a. Descargar e instalar R desde https://cran.r-project.org/, eligiendo la versión compatible con su sistema operativo.
b. Descargar RStudio (https://www.rstudio.com/products/rstudio/download/), seleccionando la versión libre de escritorio (Rstudio Desktop).
c. Descargar e instalar la versión más reciente de Rtools desde https://cran.r-project.org/bin/windows/Rtools/.

1.3 Cómo iniciar con R y RStudio

Desde la consola de R se puede acceder a la documentación más básica. Escribir **help.start()** en la consola permite iniciar una interfaz de ayuda general en el navegador, donde se encuentran disponibles diversos manuales; *An Introduction to R*, es la primera opción. Fuera del entorno de R es posible localizar estos manuales en https://cran.r-project.org/manuals.html. La función **demo()** permitirá ejecutar demostraciones para diferentes paquetes de uso general; mientras que **help()** permite obtener ayuda para algún paquete o función especificada, por ejemplo, **help(help)** para aprender a utilizar la función de ayuda. De igual forma, funciona ingresar **?help** en la consola, mientras que **??** se utiliza para realizar una búsqueda general en la web.

Afortunadamente, hoy en día es posible disponer de una cuantiosa documentación de R para diversas aplicaciones. Además de los libros, manuales y guías publicados, los foros resultan de suma utilidad para resolver cuestiones específicas de las y los usuarios. Stack Overflow es un ejemplo con gran afluencia de programadores y programadoras en R y otros lenguajes como Phyton y JavaScript, con la ventaja adicional de contar con una versión en español (https://es.stackoverflow.com/questions/tagged/r). De igual forma, los paquetes de R disponen de documentación sobre sus funcionalidades y, generalmente, ofrecen tutoriales paso a paso. Quizás esta última opción resulte la más adecuada para superar obstáculos. No obstante, la experiencia del usuario es un elemento que también le permitirá valorar las fuentes de ayuda más pertinentes.

Para iniciar con R y RStudio también es posible aprovechar la numerosa oferta de capacitación disponible en torno a este lenguaje. A reserva de parecer limitado el hacer una recomendación frente a la vasta disponibilidad de cursos en diferentes modalidades, se puede mencionar un curso en línea gratuito que puede resultar de utilidad para personas principiantes (https://es.coursera.org/learn/intro-data-science-programacion-estadistica-r). Dentro del mismo entorno de RStudio también es posible acceder a ofertas de capacitación dentro de la pestaña de *Help*, localizada en la ventana inferior derecha.

1.4 Paquetes de R

Para la mayoría de las y los usuarios, el uso de R sienta sus bases en la implementación de funcionalidades diseñadas o compiladas por otras personas usuarias y puestas a disposición a través de paquetes o librerías de uso libre. Las librerías son una colección de funciones diseñadas para atender una tarea específica. Esta es la principal fortaleza de R como *software* libre de código abierto: la facilidad de compartir funcionalidades y programas entre las y los usuarios. Al momento de elaboración de este manual existían casi 18 000 paquetes o librerías.

La instalación de paquetes se realiza con la función **install.packages()**, directamente en la consola de R valiéndose de una conexión a internet, o bien, dirigiendo esta fun-

ción a un repositorio que contenga el instalador del paquete almacenado en la PC. La forma más sencilla será utilizando esta función haciendo referencia al paquete, por ejemplo, **install.packages(ggplot2)**. Otra alternativa que puede resultar útil cuando se obstaculice la instalación por este método, es el uso de la pestaña *Packages* (localizada en la ventana inferior derecha de RStudio), en la cual se pueden visualizar los paquetes que ya han sido instalados, así como gestionar la instalación de paquetes nuevos utilizando el botón *Install*, localizado en el margen superior izquierdo dentro de esta misma pestaña. A continuación, se presenta una breve descripción de los paquetes más importantes implementados en este manual para la interpolación de la riqueza de especies:

sp. Permite la manipulación de datos espaciales en formatos 2D y 3D para la elaboración de mapas, selección, selección de subconjuntos de datos, entre otras cosas. Utiliza funciones de **rgdal**.

raster. Ofrece herramientas básicas para escritura, importación y manipulación de datos espaciales en formato *raster* (matriz). También incluye funciones más avanzadas de modelado y herramientas de interacción entre datos *raster* y vectorial.

sf. *Simple features* para R se basa en las librerías GDAL, GEOS y PROJ para escribir, manipular datos, realizar operaciones geométricas y proyectar. Ofrece una estandarización de esas tres librerías para la implementación de programas de forma más sencilla.

ggplot2. Quizás la librería para elaboración de gráficos más utilizada dentro del entorno de R. Además de la gran variedad de gráficos para representación de datos, ofrece herramientas que también son útiles para el diseño de mapas de buena calidad.

tmap. Con una sintaxis similar a **ggplot2**, facilita el diseño de mapas temáticos con buena calidad estética. Su diseño se centra en la representación de distribuciones de variables en el espacio.

tiff. Herramientas para leer, escribir y visualizar imágenes en formato ***.tiff**. Puede leer y escribir archivos y vectores sin procesar en memoria.

spatstat. Herramientas para el análisis de patrones espaciales representados con puntos. Trabaja con puntos que pueden representarse en dos y tres dimensiones. También admite patrones espaciotemporales, patrones de puntos en una red lineal y patrones de otros objetos geométricos. Admite datos de covariables espaciales, como imágenes en formato *raster*. Contiene funciones para el análisis de datos, ajuste de modelos, simulación, muestreo espacial, diagnóstico de modelos e inferencia formal.

MASS. Es un paquete estadístico que contiene múltiples funciones para ejecutar diferentes métodos estadísticos.

maptools. Conjunto de herramientas para la manipulación de datos geográficos. Incluye acceso binario a archivos **'GSHHG'**. El paquete también proporciona contenedores de interfaz para intercambiar objetos espaciales con paquetes como **'PBSmapping'**, **'spatstat.geom'**, **'maps'** y otros.

rgeos. Ofrece una interfaz en R para GEOS (Interface to Geometry Engine-Open Source), utilizado para operaciones de topología en geometrías.

olsrr. Es una librería diseñada para la fácil implementación de modelos de regresión, pruebas de heterocedasticidad y colinealidad, análisis de residuales y selección de variables en modelos ajustados por mínimos cuadrados ordinarios.

gstat. Este paquete implementa modelos geoestadísticos espaciales y espaciotemporales, de predicción y simulación. Contiene las rutinas para interpolar datos por Distancia Inversa Ponderada (IDW), variogramas y Kriging, entre otros.

randomForest. Permite implementar el algoritmo Random Forest en modelos de clasificación y regresión.

ModelMap. Se utiliza para construir y validar modelos con subconjuntos de datos, por validación cruzada o predicciones Out Of Bag (OOB). También permite la creación de gráficos y tablas con los resultados de la validación y la creación de mapas utilizando modelos. Para instalar las librerías aquí descritas, se puede utilizar el siguiente *script* en R.

#Instalación de los paquetes necesarios para el mapeo de la riqueza de especies

#con diferentes métodos de interpolación

```
install.packages("sp" , dependencies = TRUE)
install.packages("raster", dependencies = TRUE)
install.packages("ggplot2", dependencies = TRUE)
install.packages("tmap", dependencies = TRUE)
install.packages("tiff", dependencies = TRUE)
install.packages("spatstat", dependencies = TRUE)
install.packages("maptools", dependencies = TRUE)
install.packages("rgeos", dependencies = TRUE)
install.packages("MASS", dependencies = TRUE)
install.packages("olsrr", dependencies = TRUE)
install.packages("gstat", dependencies = TRUE)
install.packages("randomForest", dependencies = TRUE)
install.packages("ModelMap", dependencies = TRUE)
```

2. Introducción general a R

En este capítulo se presenta una breve introducción al uso de R, desde la ejecución de operaciones básicas, lectura de bases de datos, estadística descriptiva y representación gráfica básica. Las bases generales servirán de punto de partida. No obstante, el desarrollo de programas más complejos irá de la mano de los intereses del usuario y usuaria. A medida que se adquiere experiencia, se volverá más fácil adaptar y diseñar programas de análisis más robustos. Para los fines de este manual, la y el usuario descubrirá que es posible adaptar el código proporcionado para llevar a cabo análisis en otras áreas de estudio.

RStudio es un Ambiente Integrado de Desarrollo (IDE, por sus siglas en inglés) y ofrece un entorno que asiste al programador o programadora proporcionando múltiples elementos auxiliares **(Figura 1)**, como son el editor de código (1), la consola de R (2), la ventana de ambiente (3) y la ventana de gráficos (4). Es necesario señalar que las ventanas de ambiente y gráficos tienen más de una función. La ventana de ambiente permite visualizar el historial y las conexiones; mientras que la ventana de gráficos, a través de las diferentes pestañas, permite acceder a los archivos almacenados en el disco, ofrece asistencia para la gestión de paquetes y la búsqueda de documentación en la pestaña de *Help*, como se detalló en el capítulo anterior. La pestaña de *Viewer*, permite visualizar archivos html/pdf creados dentro de RStudio.

Como primer paso, siempre será recomendable definir el directorio de trabajo, es decir, la ruta que dirige a una carpeta en disco, a partir de la cual la o el usuario podrá importar archivos hacia RStudio (leer) o bien, guardar/exportar (escribir) diferentes tipos de archivos. El comando **setwd()** tiene esta función y se utiliza ingresando entre comillas **(“ ”)** la ruta donde se localiza la carpeta que será utilizada para alojar los archivos, por ejemplo: **setwd(“C:/directorio1/directorio2”)**. Otra forma de definir el directorio de trabajo es a través de la pestaña **Session>Set Working Directory>Choose Directory**.

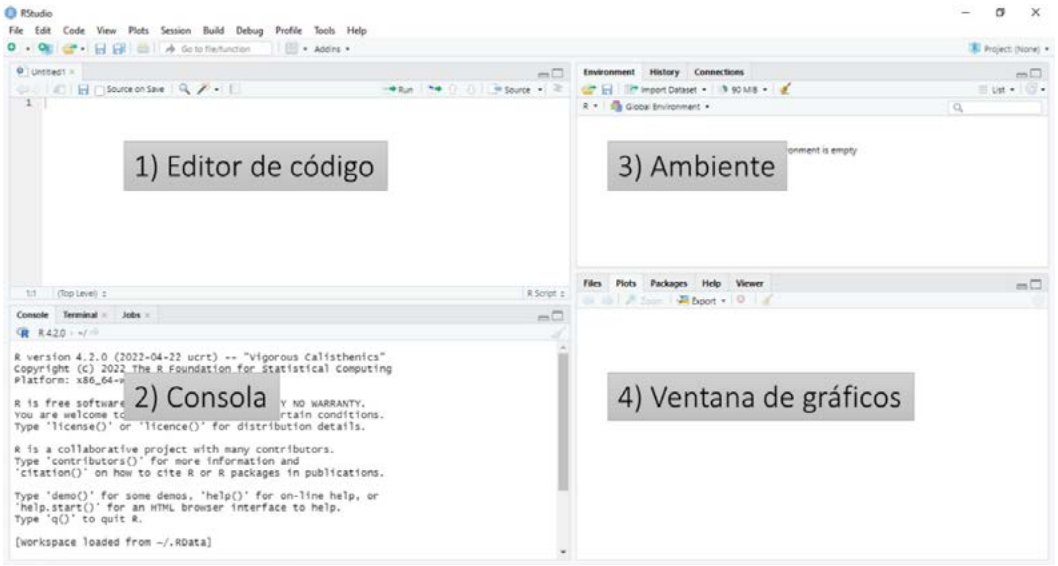


Figura 1. *Interfaz del usuario y usuaria de RStudio.*

R es un lenguaje dirigido a objetos. Un objeto puede ser un número, un carácter o un objeto lógico (TRUE o FALSE). Dentro de estos, el objeto más simple en R es un vector numérico y generalmente las operaciones están dirigidas a ellos. Otros objetos comunes son las tablas (*dataframe*), las matrices y las listas. En el editor de código de R es posible implementar operaciones matemáticas básicas de forma simple, como se muestra a continuación:

```
1+1
[1] 2
2*2
[1] 4
```

El resultado de las operaciones realizadas con este método se imprime en la consola de R de forma automática. Es posible realizar diversas operaciones al mismo tiempo, separándolas con un punto y coma "**;**".

```
5+5;8+3;6*2
[1] 10
[1] 11
[1] 12
```

La asignación de vectores o variables se puede realizar utilizando los símbolos de "menor que" y el guion medio (**<-**), simulando una flecha, o a través del signo de "igual" (**=**); la forma más común es utilizando el primero. Así, es posible almacenar valores o vectores empleando este método de asignación para después utilizarlos en operaciones aritméticas.

```
a<- 5
b<- 5
a+b
[1] 10
```

El siguiente ejemplo contiene la expresión **a+b**, es decir, la suma de los dos objetos que tienen un valor igual a 10.

```
suma<-a+b
suma
[1] 10
```

Con la función **c()** es posible combinar valores para crear un vector numérico al que pueden dirigirse otras operaciones. Las operaciones entre vectores en R siempre se realizarán de forma ordenada, como en el siguiente ejemplo, donde se crean dos vectores numéricos con la misma longitud y luego se multiplican. El resultado de esta operación también puede alojarse en un objeto que después podemos "**imprimir**" con el comando **print()**. De igual forma podemos imprimir en la consola el vector si es necesario visualizar los valores que contiene.

```
valores<-c(2,4,6,8,10)
otrosvalores<-c(3,6,9,12,15)
valores*otrosvalores
[1]  6  24  54  96 150
```

```
resultado<- valores*otrosvalores
print(resultado)
[1]  6  24  54  96 150
print(valores)
[1]  2  4  6  8 10
```

Los estadísticos descriptivos pueden obtenerse con las funciones **max()**, **min()**, **mean()** y **sum()**, dirigidos siempre al vector de interés. Un resumen de estadísticos se obtiene mediante **summary()**.

```
max(valores)
[1] 10
min(valores)
[1] 2
mean(valores)
[1] 6
sum(valores)
[1] 30
summary(valores)

#Min. 1st Qu.  Median   Mean 3rd Qu.   Max.
#   2     4     6     6     8    10
```

Una de las formas más habituales de importar datos hacia R es a través de tablas, llamadas *dataframe* dentro de este entorno. El comando por utilizar dependerá del formato en el que se encuentre el *dataframe*. Para leer uno en formato ***.txt**, se utiliza la función **read.table()**, que puede operar de dos formas: 1) definiendo el directorio **read.table("C:/directorio1/directorio2", header=TRUE)**, **header=TRUE** que indica que los nombres de las variables se encuentran en la primer línea; y 2) abriendo el explorador de archivos para localizar el *dataframe* de forma manual: **read.table(file.choose(), header=TRUE)**. Cuando los datos se encuentran en formato ***.csv,** delimitado por comas, su importación se realiza mediante **read.csv("C:/directorio1/directorio2/nombre_del_archivo")**; en este segundo caso no es necesario especificar la presencia de encabezados (**header=TRUE**).

La función **data.frame()** permite crear *dataframes* nombrando cada columna. En el siguiente ejemplo, se crea uno con tres columnas: ID, color y valor, y se asigna al objeto "**datos**". Existen diferentes comandos para explorar elementos de un *dataframe*: **dim()** regresa la dimensión en número de filas y columnas, **length()** arroja el largo de un *dataframe* que se define de acuerdo con su número de columnas, mientras que **names()** regresa los nombres de cada columna. Cuando se trabaja con datos en forma de *dataframe*, es posible generar estadísticos descriptivos de alguna variable particular; la forma más sencilla de referirse a una sola variable dentro de un *dataframe* es mediante "**$**".

```
datos <- data.frame(
 "ID" = 1:4,
 "color" = c("azul", "blanco", "negro", "rojo"),
 "valor" = c(5.5, 4.8, 4.5, 5.9)
)
datos
ID  color valor
 1  azul  5.5
 2 blanco  4.8
 3  negro  4.5
 4  rojo  5.9
dim(datos)
[1] 4 3
length(datos)
[1] 3
names(datos)
[1] "ID"    "color" "valor"
mean(datos$valor)
[1] 5.175
```

Finalmente, para la manipulación de *dataframes* resulta conveniente familiarizarse con la función **subset()**, que permite crear subconjuntos de datos a partir de los operadores **==** (igual), **!=** (diferente), **<** (menor que), **>** (mayor que), **<=** (menor o igual) y **>=** (mayor o igual).

```
subset(datos, color==”azul”)
 ID color valor
1    1 azul   5.5
subset(datos, valor>=4.8)
  ID  color valor
1    1  azul   5.5
2    2 blanco   4.8
4    4   rojo   5.9
subset(datos, valor>5.5)
 ID color valor
4    4  rojo   5.9
```

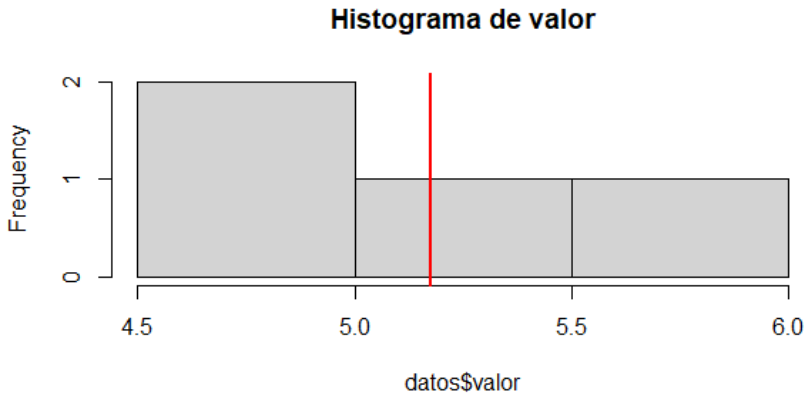
El uso de corchetes también es útil para llamar una parte de los datos utilizando atributos de su posición. En esta sección se agregarán notas dentro del código empleando el sigo de número (#), de igual forma puede emplearse este método en el editor de código; todo lo que esté después de un “#”, será ignorado al ejecutar el programa.

```
datos[ , 3] #Sólo la tercera columna
[1] 5.5 4.8 4.5 5.9
datos[ , 1] #Sólo la primera columna
[1] 1 2 3 4
datos[1, ] #Sólo la primera fila
 ID color valor
1    1 azul   5.5
datos[1 , 2] #Sólo la primera fila, columna 2
[1] “azul”
> datos[1:2, ] #Filas del 1 al 2
 ID  color valor
1    1  azul   5.5
2    2 blanco   4.8
```

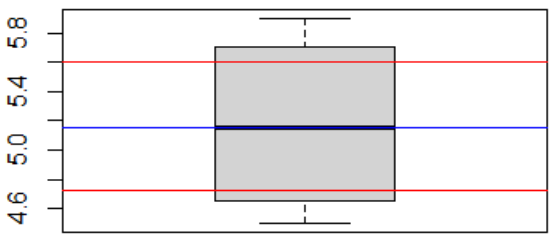
A partir del sencillo *dataframe* generado anteriormente, es posible construir algunas representaciones que ilustran el potencial de R para el diseño de gráficos. Los siguientes ejemplos parten de la funcionalidad de gráficos que está integrada con el lenguaje.

En los siguientes capítulos, los mapas y otros tipos de gráficos más elaborados se desarrollarán utilizando los paquetes **ggplot2** y **tmap**. Como regla general, para la representación gráfica en R se utilizan "**capas**" que añaden elementos a un gráfico "**base**". La misma lógica se sigue en las funciones de la librería **ggplot2** y **tmap**. En los siguientes ejemplos, se emplea la función **abline()** para agregar líneas de referencia.

```
hist(datos$valor, main="Histograma de valor")
abline(v= mean(datos$valor), col="red", lwd=2)
```

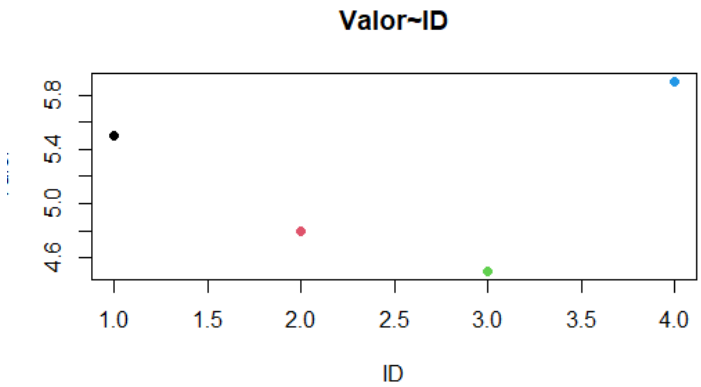



```
boxplot(datos$valor)
q25 <- quantile(datos$valor, .25)
abline(h=q25, col="red")
q50 <- quantile(datos$valor, .5)
abline(h=q50, col="blue")
q75 <- quantile(datos$valor, .75)
abline(h=q75, col="red")
```



En la representación de una variable en función de otra, es posible definir una condición para la propia representación de los datos. En el siguiente ejemplo de un gráfico de puntos de la variable "**valor**" en función de la variable "**ID**", también se designa una diferenciación de color (**col=**), en relación con esta segunda variable. Aquí, también se define un nombre para los ejes.

```
plot(datos$valor ~ datos$ID,pch=16, col=datos$ID,main="Valor~ID",
xlab="ID", ylab="Valor")
```



Finalmente, es posible acceder a la ayuda para conocer cómo utilizar las diferentes funciones de R. Para obtener una descripción de las funciones, se utilizan las funciones **?**, **??**, y **help()**. Por ejemplo, para conocer qué se puede hacer en la función **plot()** se escribe: **?plot**, **??plot** o **help(plot)**.

3. Datos para la estimación de la riqueza de especies

3.1 Área de estudio

El área de estudio de donde se tomaron los datos de campo para mapear la riqueza de especies del bosque, comprende una extensión de 64 km^2 y se ubica en el estado de Quintana Roo, México (18°53'54"-18°58'14"N, y 88°10'04"-88°14'37"O) **(Figura 2)**. El área tiene una topografía plana y el clima es cálido tropical subhúmedo con un período marcado de sequía. La temperatura media anual es de 26 °C, el mes más frío es enero con 23 °C, y los meses más cálidos son julio y agosto con 29 °C. La precipitación media anual está entre 1000 y 1300 mm, concentrada en un período entre junio y octubre, y la estación seca es durante los meses de diciembre a abril (Cabrera Cano et al., 1982). El área de estudio está cubierta por tres tipos de vegetación: selva mediana subperennifolia, sabana y selva baja inundable. La selva media subperennifolia es el tipo de vegetación que domina en el área estudiada. Los árboles alcanzan alturas de hasta 25 m y la comunidad vegetal es estructuralmente compleja con dos o tres capas de dosel que consisten principalmente en árboles, arbustos y lianas. En este tipo de vegetación se pueden encontrar diferentes edades de sucesión: Etapa 1 (vegetación secundaria ≤ 3 años); Etapa 2 (4-10 años); Etapa 3 (11-19 años) y Etapa 4 (selva de más de 20 años, incluidos rodales maduros). Los otros dos tipos de vegetación están asociados a áreas que se inundan durante la época de lluvias. La sabana es una comunidad dominada por hierbas y carece de una cubierta arbórea continua. En contraste, la selva baja inundable tiene árboles con alturas que pueden llegar a los 10 m (Cabrera Cano et al., 1982). En total, dentro del área de estudio se pueden distinguir seis clases de cobertura vegetal.

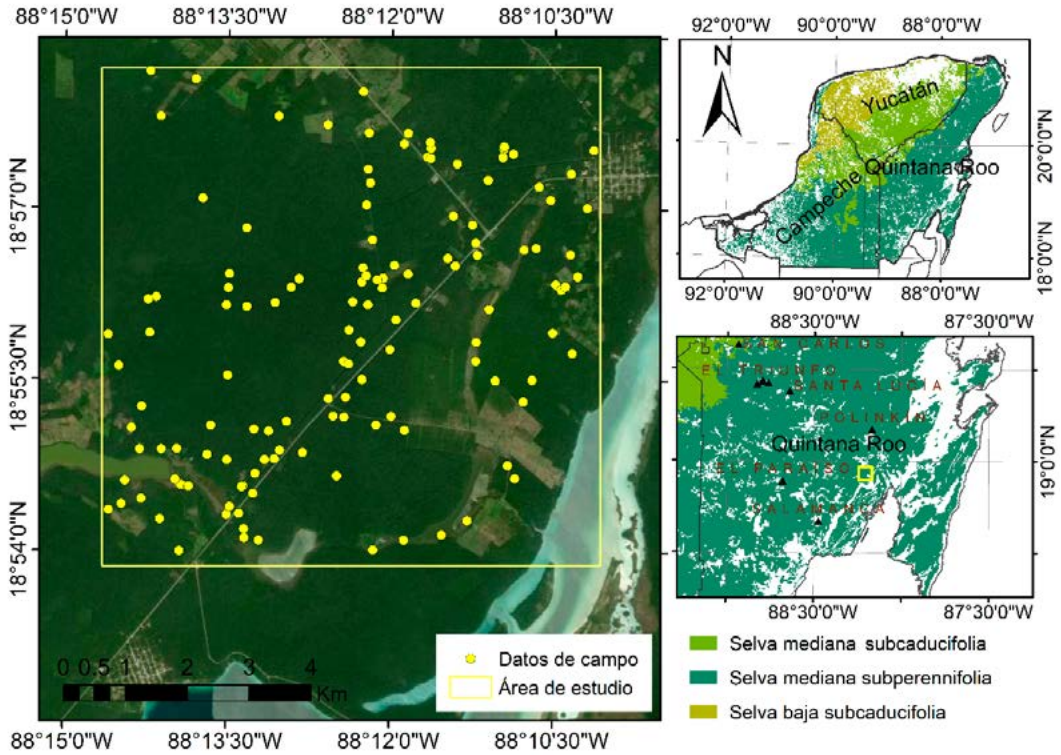


Figura 2. *Ubicación del área de estudio y de los sitios de muestreo.*

3.2 Muestreo de campo y procesamiento de datos de la vegetación

Los datos de riqueza de especies se registraron en dos levantamientos de información de campo realizados durante la época de lluvias de los años 2000 y 2001. Estos levantamientos de información tuvieron un diseño de muestreo estratificado, en el que cada estrato (tipo de vegetación) se muestreó en proporción a su área. El tamaño final de la muestra fue de 130 parcelas de campo o unidades de muestreo.

En todas las parcelas de campo se obtuvo su localización geográfica utilizando una unidad de GPS convencional. Cada unidad de muestreo consistió en un cuadrante de 10 x 10 m, que se utilizó para medir todos los árboles de más de 3 m de altura. En cada parcela se calculó el número de especies de árboles por sitio de muestreo (es decir, la densidad de especies *sensu* Gotelli et al., 2001), como medida de la diversidad local o diversidad α.

Debido a que la mayoría de las especies de plantas tropicales tienen densidades muy bajas, la evaluación de la riqueza de especies a nivel comunitario es difícil y requiere el muestreo de grandes áreas para capturar la rareza de la mayoría de las especies en los bosques tropicales. Esto se puede lograr utilizando pocas unidades de muestro muy grandes o colocando una gran cantidad de parcelas pequeñas distribuidas en toda el área estudiada. En este estudio se utilizaron parcelas pequeñas (100 m^2), no solo por ser más fáciles de medir, sino también porque su uso permite una adecuada estimación de la diversidad vegetal a nivel de paisaje (Plotkin et al., 2000), especialmente en paisajes irregulares y fragmentados.

3.3 Datos de sensores remotos y procesamiento de imágenes

Se utilizó una imagen Landsat 7 Thematic Mapper (ETM+) adquirida en abril de 2000, la cual fue georreferenciada a la Proyección Universal Transversa de Mercator (WGS 84) y recortada al área de estudio. El error estándar cuadrático promedio (RMSE, por sus siglas en inglés) para la georreferenciación fue de 0.42 píxeles, correspondiente a 12.6 m en el suelo. La imagen fue corregida radiométrica y atmosféricamente para minimizar el efecto de la dispersión atmosférica. Los valores DN (números digitales) de las bandas ETM+ se convirtieron a radiancia y se transformaron a unidades de reflectancia exoatmosférica, utilizando las ecuaciones sugeridas por el Landsat 7 Handbook (NASA, 2019). A continuación, se identificaron los píxeles que contenían cada uno de los 130 cuadrantes de muestreo y la reflectancia en tres longitudes de onda (5-infrarrojo de onda corta: 1.55-1.75 hm; 4-infrarrojo cercano: 0.76-0.90 hm; y 3-rojo: 0.63-0.69 hm) y fueron extraídos. El NDVI se calculó a partir de las bandas del Infrarrojo Cercano (NIR, banda 4) y Rojo (R, banda 3) y se extrajo.

Se calcularon dos medidas de textura de primer orden y seis de segundo orden en cada uno de los 130 cuadrantes de muestreo para tres bandas Landsat ETM+ (ETM+3, ETM+4 y ETM+5), y para los valores calculados de NDVI. Las medidas de segundo orden se calcularon en cuatro direcciones (0°, 45°, 90°, 135°) y se promediaron para obtener un único valor de textura espacialmente invariable como lo sugieren Haralick et al. (1973). Las medidas de textura de primer y segundo orden se calcularon utilizando una ventana de 5 × 5 píxeles para capturar la variabilidad dentro de un fragmento de la misma clase de vegetación y evitar efectos de borde, considerando que el tamaño mínimo de un fragmento es de aproximadamente 2.25 ha (25 píxeles). Las medidas de textura se calcularon utilizando funciones de R, como se indica en Hernández-Stefanoni et al. (2021).

Las medidas de textura de primer orden son propiedades estadísticas que no consideran las relaciones de píxeles vecinos y que se derivan de los valores de la imagen original dentro de un tamaño de ventana determinado. Aquí se utilizaron la media (**med**) y la varianza (**var**) como medidas de la variabilidad espectral dentro de una ventana de 3 x 3 píxeles. Las medidas de textura de segundo orden consideran todas las relaciones espaciales entre grupos de dos píxeles vecinos dentro de la ventana (Haralick et al., 1973). Estas medidas de textura se calculan a partir de una matriz de coocurrencia de nivel de gris (GLCM) que contiene las probabilidades de coocurrencia de valores de píxeles para pares de píxeles en una dirección y distancia dadas. Los seis estadísticos de segundo orden considerados en este estudio corresponden a tres categorías. Uno de ellos se basa en el grado de contraste entre píxeles e incluye homogeneidad (**hom**), contraste (**cont**) y disimilitud (**dis**); otro se basa en la organización de los píxeles dentro de una ventana (entropía –**ent**–, segundo momento angular –**asm**–), mientras que el restante se basa en estadísticas (correlación –**cor**–). Para obtener una descripción y fórmulas para calcular las medidas de textura de primer y segundo orden, consulte Anys et al. (1994) y Haralick et al. (1973), respectivamente.

La descripción de los datos derivados de las imágenes de satélite utilizados en este manual se encuentra en el Apéndice.

4. Preparación de datos

En este capítulo se describen los procedimientos para leer el mapa de coberturas terrestres, los cuales permitirán delimitar aquellas áreas donde se harán las estimaciones de riqueza de especies por medio de la interpolación espacial. Por otro lado, se crea una base de datos con la información del número de especies de cada parcela de muestreo, la ubicación en el espacio de estas parcelas e información auxiliar que será utilizada para ejecutar los diferentes procedimientos de interpolación. Por último, se crea un objeto SpatialGrid que es necesario para llevar a cabo las interpolaciones.

4.1 Mapa de coberturas

Para tener una estimación más realista de la distribución espacial de la riqueza de especies en el área de estudio, se utiliza un mapa que identifique las áreas de bosque de aquellas que están desprovistas del mismo, es decir, áreas deforestadas, cuerpos de aguas y terrenos agrícolas. Con este mapa se excluyen las áreas que no tienen bosque cuando se hace la estimación del atributo estudiado. De igual manera, el mapa de coberturas terrestres permite aplicar algunos métodos de interpolación que se llevan a cabo en tipos de vegetación específicos.

El mapa de cobertura terrestre se generó a partir de imágenes Landsat 7 Thematic Mapper (TM). En este mapa se identificaron cuatro clases de selva mediana subperennifolia en diferentes etapas de sucesión, además de otros dos tipos de vegetación: la sabana y la selva baja inundable, así como áreas deforestadas, pastizales y áreas de cultivo. Para obtener este mapa se utilizó una clasificación supervisada de las bandas 5 (infrarrojo de onda corta: 1.55 - 1.75 hm), 4 (infrarrojo cercano: 0.76 - 0.90 hm) y

3 (rojo: 0.63 - 0.69 hm), con el algoritmo de máxima verosimilitud. Las parcelas de muestreo se utilizaron para la evaluación de la precisión del mapa, el cual tuvo una precisión general de 82.3 % (Hernández-Stefanoni et al., 2006).

Para facilitar la ejecución de los *scripts* de R que se utilizan en este manual, se recomienda establecer un espacio de trabajo específico, es decir, la ruta de una carpeta para leer y escribir archivos por *default*. Para poder definir este espacio de trabajo se utiliza la función **setwd()**, y para saber cuál es el espacio de trabajo se utiliza la función **getwd()**. Se guardó el directorio de trabajo en la variable "**folder"** para usarlo posteriormente. En este manual, la ruta de trabajo para mapear la riqueza de especies usando diferentes métodos de interpolación es la siguiente: "**c:/met_int**".

```
#Definir directorio de trabajo, este debe contener los archivos de datos y #el mapa con el área de estudio delimitada

setwd("C:/met_int")
getwd()

# [1]"c:/met_int"

folder <- getwd()
```

Se abren los paquetes que se utilizarán en este módulo y que fueron descritos en la sección 1.3.

```
#Abrir los siguientes paquetes

library(sp)
library(raster)
library(ggplot2)
library(tmap)
library(tiff)
```

El mapa de coberturas terrestres que se utiliza para construir diferentes máscaras está almacenado como un archivo de tipo TIF. Se utiliza la función **raster()** para dar lectura a la imagen "**sup_class_filter.tif**". Esta imagen es asignada en el objeto llamado "**clases**". Después de leer la imagen se asignan las coordenadas y la proyección geográfica que este mapa tiene (WSG84), utilizando la función **crs()** y se imprime. La impresión del objeto "**clases**", nos indica que tenemos una imagen *raster* con una resolución de 30 m, la proyección geográfica que tiene, sus dimensiones, así como el nombre del archivo en donde estaba guardado.

```
#Leer mapa de coberturas y reclasificarlo para crear una máscara

clases <- raster("sup_class_filter.tif")
```

```
#Asignar las coordenadas del mapa de cobertura con UTM n16 WGS84

clases.proj <-
 "+proj=utm +zone=16 +units=m +ellps=WGS84 +datum=WGS84 +no_defs"
crs(clases) <- crs(clases.proj)
clases

# class       : RasterLayer
# dimensions  : 268, 268, 71824  (nrow, ncol, ncell)
# resolution  : 30, 30  (x, y)
# extent      : 368985, 377025, 2089965, 2098005  (xmin, xmax, ymin, ymax)
# crs         : +proj=utm +zone=16 +datum=WGS84 +units=m +no_defs
# source      : sup_class_filter.tif
# names       : sup_class_filter
```

Una vez leído el mapa de coberturas y asignada la proyección geográfica adecuada, se puede graficar este mapa utilizando diferentes opciones. La primera es emplear la función **plot()**. Esta tiene varios parámetros, aquí se utilizaron 2 de ellos. El primero es un objeto de tipo *raster,* y el segundo la variable **my_col**, en la cual se asignan los colores de cada categoría para el mapa de cobertura. Otra alternativa es por medio del paquete **tmap**, con el que se pueden generar mapas temáticos de manera sencilla

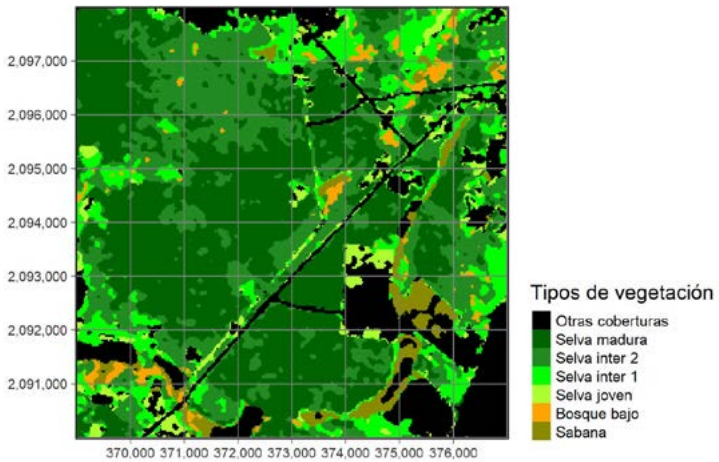
y flexible. La sintaxis para crear mapas de salida es similar a la que utiliza el paquete **ggplot2**, pero adaptada a los mapas. Para obtener la gráfica con **tmap**, primero se realizó una reclasificación de las coberturas terrestres usando función **reclassify()**. Esta función tiene varios parámetros; el primero de ellos es un objeto de tipo *raster*, en este caso es el mapa de coberturas terrestres "**clases**". El segundo parámetro que se incluye en esta función es una matriz para la reclasificación. En esta se especifica el rango de valores y cómo serán reclasificados. La imagen reclasificada se guarda en el objeto *raster* "**rveg**".

```
#Opción sencilla para graficar el mapa

my_col <- c("black", "darkgreen","green", "greenyellow", "forestgreen", "orange",
"yellow4")
plot (clases,legend = TRUE, col = my_col)
```

```
#Reclasificar el mapa de categorías

m0 <- c(0,0,1,1,2,2,3,3,4,4,5,5,6,6)
mcmat <- matrix(m0, ncol=2, byrow=TRUE)

rveg <- reclassify(clases, mcmat)
```

Para graficar con el paquete **tmap** se utilizan varias funciones. Al principio se coloca la función **tm_shape()**, la cual tiene como parámetro el conjunto de datos del cual se creará el mapa; en este caso, es un objeto de tipo *raster* "**rveg**". Enseguida se colocan uno o más niveles de funciones que definen el tipo de visualización y están separadas por el signo "**+**". Las funciones que se utilizan en este mapa fueron: **tm_raster()**, en donde se especifica que es un mapa de diferentes categorías (en formato *raster*); **tm_grid()**, para imprimir la cuadrícula con coordenadas; y la función **tm_layout()**, en donde se especifica la ubicación de la leyenda. Este conjunto de funciones es asignado al objeto de tipo **tmap** llamado "**tm1**", que después se imprime.

```
#Impresión de mapa con los tipos de vegetación

tm1 <- tm_shape(rveg)+
 tm_raster(style= "cat",
          labels = c("Otras coberturas","Selva madura",
                     "Selva inter 2", "Selva inter 1",
                     "Selva joven", "Bosque bajo", "Sabana"),
          palette = c("black", "darkgreen", "forestgreen"
                     ,"green","greenyellow","orange", "yellow4"),
          title="Tipos de vegetación")+
 tm_grid()+
 tm_layout(legend.outside = TRUE,
          legend.position= c("right", "bottom"))

tm1
```



Para poder grabar en disco este mapa graficado con el paquete **tmap**, primero se define el directorio en donde se guardará dicho mapa; el nombre del directorio es "**Maps**" y está ubicado en "**C:/met_int/Maps/**". Posteriormente, se utiliza la función **tmap_save()** para grabar este mapa en disco. Esta función puede tener hasta 12 parámetros, aquí se utilizaron 5. En el primero se indica

el objeto que se debe guardar en disco, que es **tm1**; el segundo corresponde al nombre de archivo, incluida la extensión y la ruta. Después se especifican el largo y ancho de la imagen que se imprimirá; debido a que no se especifican las unidades, se utiliza por *default* píxeles. Finalmente "**asp**" se establece en 0 para indicar que el marco del mapa debe colocarse en los bordes de la imagen.

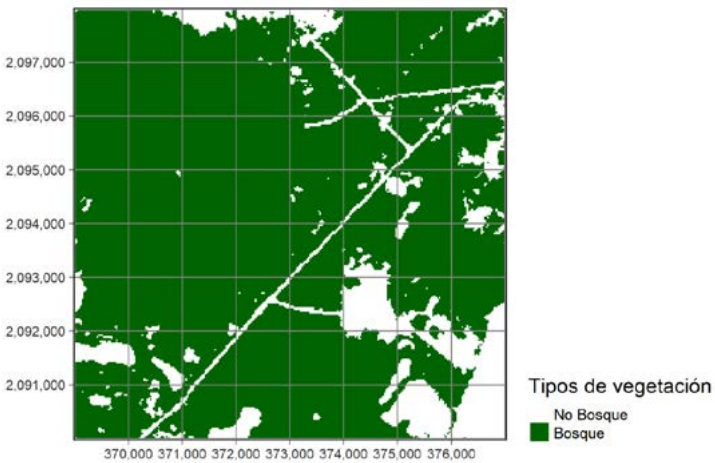
```
#Crear el directorio de salida para guardar los mapas

dir.MAPS <-  file.path(folder,"Maps") #Definir la carpeta a usar

#Evaluar si ya existe la carpeta nombrada
ifelse(dir.exists(dir.MAPS), dir.MAPS,
        #Si no existe se crea la carpeta
        dir.create(dir.MAPS))
```

```
#Grabar mapa en disco

tmap_save(tm1,
        filename= paste(dir.MAPS, "map_cober.tiff", sep="/"),
        width=1720, height=1070, asp= 0)
```

Para excluir de las estimaciones de la riqueza de especies las áreas donde no hay bosque, se utilizó la función **reclassify()** del paquete *raster*. Esta función crea un nuevo objeto *raster* con valor de 1 en aquellos tipos de vegetación que son de bosque, mientras que el resto de los píxeles del objeto *raster* tendrán un valor de 0. Esto es, la función **reclassify(x, rcl, filename='', right = FALSE...)** se utilizó con 3 parámetros: **x** es el objeto *raster* a ser reclasificado; **rcl** es la matriz para reclasificar, en donde se especifica que las coberturas con bosque serán clasificadas con un valor 1, y las áreas de no bosque serán actualizadas en el objeto *raster* con un valor 0; y **right** es un argumento lógico que indica si los intervalos definidos deben cerrarse por la derecha o viceversa. En este caso se define como **FALSE**, los intervalos son cerrados por la izquierda. Como resultado de aplicar la función **reclassify()**, se crea el objeto *raster* llamado "**rc**", que contiene la imagen reclasificada. Por último, se grafica el objeto *raster* en donde se aplicó la reclasificación para su visualización usando **tmap** y se guarda en disco usando la función **tmap_save()**.

```
#Creación de una máscara bosque no bosque

m <- c(0, 1, 0,1, 7, 1)
rclmat <- matrix(m, ncol=3, byrow=TRUE)

rc <- reclassify(rveg, rclmat, right = FALSE)
```

```
#Visualización de mapa de la máscara

tm1 <- tm_shape(rc)+
 tm_raster(style= "cat", labels = c("No Bosque", "Bosque"),
           palette = c("white", "darkgreen"),
           title="Tipos de vegetación")+
 tm_grid()+
 tm_layout(legend.outside = TRUE,
           legend.position= c("right", "bottom"))

tm1
```

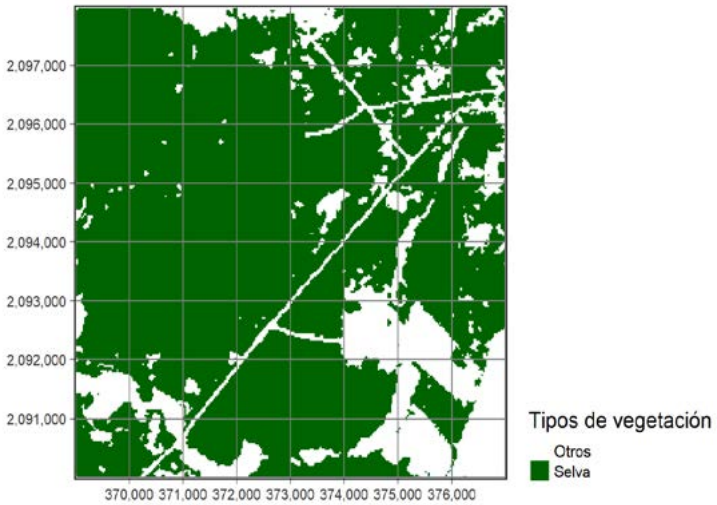


```
#Grabar mapa en disco

tmap_save(tm1,
          filename= paste(dir.MAPS,  "map_bosque.tiff", sep="/"),
          width=1720, height=1070, asp= 0)
```

Se crearon dos objetos nuevos de tipo *raster* con la función **reclassify()** del paquete *raster*. El primero de ellos, llamado "**rs**", tendrá valor de 1 en aquellas áreas cubiertas con selva mediana subperennifolia, incluyendo las 4 etapas de sucesión, y 0 en el resto de los píxeles. El segundo objeto *raster,* llamado "**rb**", tendrá valor de 1 en áreas cubiertas con sabana y bosque bajo inundable; en el resto de los píxeles se asigna el valor de 0. Esto permitirá obtener dos máscaras, una para la selva y la otra para los otros dos tipos de vegetación. Con estas dos máscaras es posible obtener interpolaciones para esas dos grandes categorías. La función **reclassify()** se aplicó de manera similar a como se hizo antes. Por último, se graficaron los objetos *raster* para su visualización usando **tmap** y se guardaron en disco usando la función **tmap_save()**.

```
#Reclasificar las categorías de vegetación de la clase selva

ms <- c(0,1,0,1,5,1,5,7,0)
rclmats <- matrix(ms, ncol=3, byrow=TRUE)

rs <- reclassify(rveg, rclmats, right = FALSE)
```

```
#Mapa de la máscara de selva

tm1 <- tm_shape(rs)+
 tm_raster(style= "cat",labels = c("Otros", "Selva"),
          palette = c("white", "darkgreen"),
          title="Tipos de vegetación")+
 tm_grid()+
 tm_layout(legend.outside = TRUE,
          legend.position= c("right", "bottom"))

tm1
```

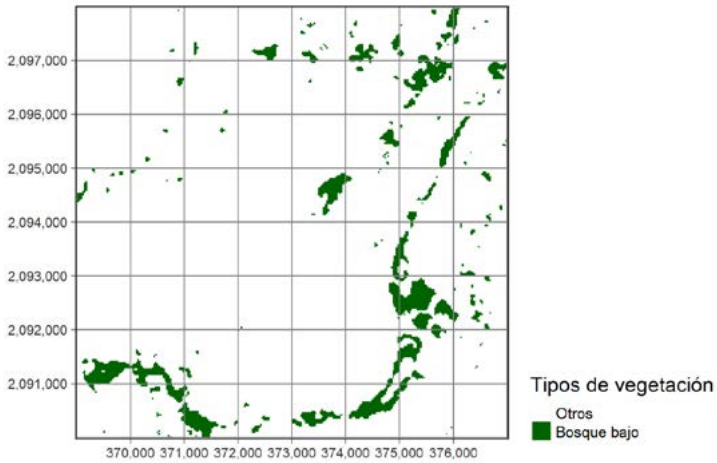



```
#Grabar mapa en disco

tmap_save(tm1,
          filename= paste(dir.MAPS, "map_selvas.tiff", sep="/"),
          width=1720, height=1070, asp= 0)
```

```
#Reclasificar las categorías de vegetación de la clase selva baja
#inundable

mb <- c(0,5,0,5,7,1)
rclmatb <- matrix(mb, ncol=3, byrow=TRUE)

rb <- reclassify(clases rveg, rclmatb, right = FALSE)
```

```
#Visualización del mapa de la máscara de la selva baja inundable

tm1 <- tm_shape(rb)+
 tm_raster(style= "cat",
        labels = c("Otros", "Bosque bajo"),
        palette = c("white", "darkgreen"),
```

```
        title="Tipos de vegetación")+
 tm_grid()+
 tm_layout(legend.outside = TRUE,legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
        filename= paste(dir.MAPS,"map_bajos.tiff", sep="/"),
        width=1720, height=1070, asp= 0)
```

4.2 Leer archivo de datos y crear objetos SpatialPointDataFrame

El archivo con los datos de cada una de las parcelas de campo es **dat.csv**, el cual se encuentra en la carpeta "**C:/met_int**". Este archivo contiene la información para cada parcela durante el muestreo en campo con relación al número de especies y tipo de vegetación, así como información de la unidad de muestreo (identificación, coordenadas geográficas del centro de la parcela). Adicionalmente contiene información espec-

tral y de textura de las bandas TM3, TM4 y el NDVI. Para cargar este archivo se utiliza la función **read.csv()**.

Durante este proceso, el archivo leído es asignado a un objeto *dataframe* con el nombre "**datos**". Se puede observar que el objeto tiene 130 registros y 32 variables. Al dar doble clic en "**datos**", aparecerá en la ventana superior derecha el archivo leído con las variables y los datos de cada registro. Sin embargo, la función **head(datos)** muestra todas las variables que contiene este *dataframe* y los primeros registros.

```
#Cargar el archivo que contenga los datos de campo de la variable de
#interés con coordenadas

library(spatstat)

datos <- read.csv("dat.csv")

head(datos)

#   ID     x      y Num_esp t_veg TM3 TM4  NDVI TM3_med TM3_var TM3_hom TM3_con
# 1  1 370997 2091679     27    2  40  95 0.407    4.02  2.7224  0.6217    3.40
# 2  2 372038 2094463     24    1  39 102 0.447    3.22  0.1704  0.8050    0.39
# 3  3 372218 2091797     32    2  38  96 0.433    3.21  0.1656  0.8250    0.35
# 4  4 372769 2091416     34    1  38 100 0.449    3.14  0.1792  0.8450    0.31
# 5  5 371962 2092300     25    2  39 100 0.439    3.39  0.4176  0.7364    1.07
# 6  6 375526 2091578     31    2  40 102 0.437    6.89 26.4104  0.3856   35.28
#   TM3_dis TM3_entr TM3_smo TM3_cor TM4_med TM4_var TM4_hom TM4_con TM4_dis
# 1    1.10  2.1098  0.1512 -0.6562   35.41  7.0096  0.4325    4.95    1.69
# 2    0.39  1.0280  0.4208  6.1968   36.76  2.2232  0.5656    1.97    1.05
# 3    0.35  1.0224  0.4464  4.8611   36.06  2.1752  0.4641    3.61    1.47
# 4    0.31  0.9468  0.5072  5.1831   37.85  0.9512  0.5960    1.36    0.90
# 5    0.61  1.4847  0.2744  1.5563   38.57  0.5248  0.6000    1.64    0.94
# 6    3.88  2.7386  0.0744 -0.8518   38.51  4.5712  0.3776    6.38    2.00
#   TM4_entr TM4_smo  TM4_cor NDVI_med NDVI_var NDVI_hom NDVI_con NDVI_dis
# 1  2.8566  0.0624  -3.7821   55.24  11.2224   0.4473    12.06    2.12
```

```
# 2  2.6732 0.0824 -1.4357  57.20 1.4208  0.5500   1.62   1.02
# 3  2.7577 0.0712 -1.6461  57.00 1.5000  0.4718   2.68   1.32
# 4  2.3245 0.1080 -0.0852  57.76 0.8032  0.6100   1.26   0.86
# 5  2.1076 0.1496 -0.0559  57.54 0.7872  0.6055   2.24   0.98
# 6  2.9589 0.0560 -11.0976  53.43 39.9168  0.2657   50.01   5.01
#   NDVI_entr NDVI_smo NDVI_cor
# 1   2.7907  0.0672 -0.6796
# 2   2.4959  0.0904 -0.1064
# 3   2.6461  0.0832 -0.3792
# 4   2.1938  0.1312 -0.0313
# 5   2.0733  0.1544 -0.0257
# 6   3.0559  0.0504 -14.2181
```

El *dataframe* "**datos**" que se acaba de crear contiene la base de datos para realizar las interpolaciones. Sin embargo, la mayoría de las funciones de los diferentes métodos de interpolación requieren que la información se encuentre en un objeto del tipo **SpatialPointDataFrame**, que contiene diferentes atributos como en un *dataframe*, pero que, además, contiene las ubicaciones de los puntos en el espacio. Por lo tanto, en esta parte del *script* se convirtió el *dataframe* "**datos**" a un objeto **SpatialPointDataFrame** de la siguiente manera. Primero se creó un nuevo *dataframe* que llamamos "**coor**", que contiene exclusivamente las coordenadas de cada registro en la base de datos (*dataframe* "**datos2**"). Después se utilizó la función **SpatialPoints()**, que tiene como parámetro un objeto *dataframe* para convertir este objeto en otro de la clase **SpatialPoints**, al cual nombramos como "**sp.point**". Enseguida se asignaron las coordenadas y la proyección geográfica de los puntos, utilizando la función **proj4string()**. Por último, se utilizó la función **SpatialPointsDataFrame()** con dos parámetros: "**sp.point**", un objeto de la clase **SpatialPoints**; y el *dataframe* "**datos**", para crear un objeto **SpatialPointDataFrame** que nombramos como "**spdf**".

```
#En el paquete spatstat todos los objetos se asumen que son planos
#Esto significa que spatstat no está diseñado para trabajar directamente con
#coordenadas geográficas (longitud y latitud)
#Eliminar del dataframe datos, las variables que no correspondan a las
#coordenadas
```

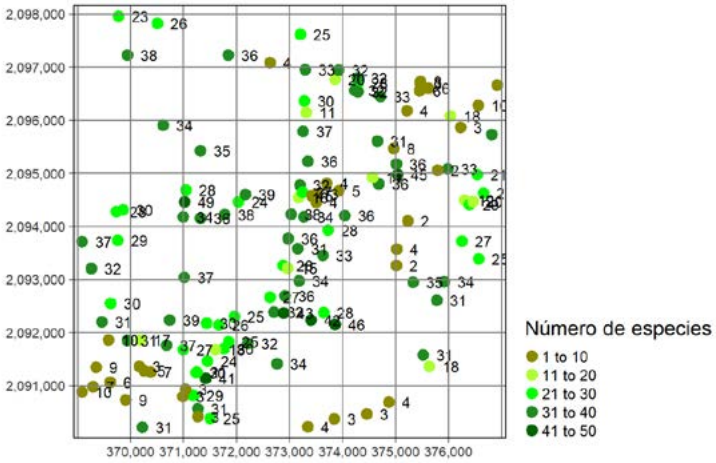
```
coor <- datos[,c(2,3)]
```

```
#Convertir puntos de un dataframe a SpatialPoints

sp.points <- SpatialPoints(coor)
proj4string(sp.points) <-
 "+proj=utm +zone=16 +units=m +ellps=WGS84 +datum=WGS84 +no_defs"
```

```
#Convertir Spatial points a SpatialpointDataFrame

spdf = SpatialPointsDataFrame(sp.points, datos)
```

Con la finalidad de visualizar la distribución espacial de las parcelas de muestreo y sus valores de riqueza de especies, se creó un mapa usando el objeto "**spdf**" y diferentes funciones del paquete **tmap**. Posteriormente se guardó este mapa en disco usando la función **tmap_save()**.

```
#Mapa de las parcelas de todos los tipos de vegetación

my_pal <- c("yellow4", "greenyellow", "green","forestgreen","darkgreen")

tm1 <- tm_shape (spdf) +
 tm_dots(col="Num_esp",
       palette =my_pal,
       title= "Número de especies", size= 0.3)+
 tm_text("Num_esp", just="left", xmod= .5, size = 0.7) +
 tm_grid()+
 tm_layout(legend.outside = TRUE,
        legend.position= c("right", "bottom"))

tm1
```

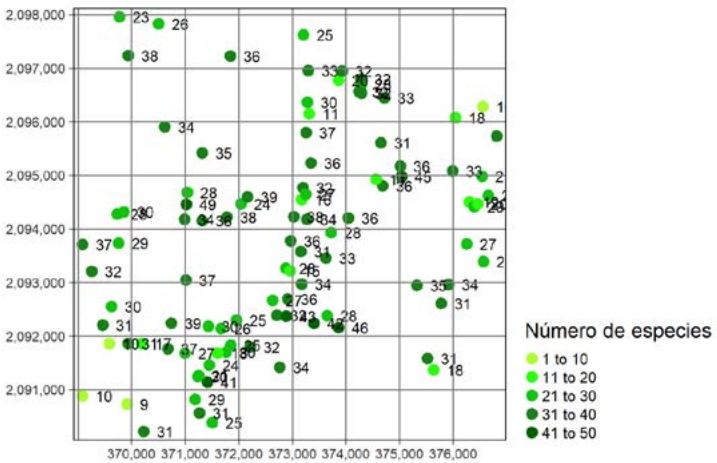



```
#Grabar mapa en disco

tmap_save(tm1,
          filename= paste(dir.MAPS,"map_parcelas.tiff", sep="/"),
          width=1720, height=1070, asp= 0)
```

En algunos de los procedimientos de interpolación que se revisarán en los siguientes capítulos, se aplica el análisis para un grupo particular de datos. Por lo tanto, se obtuvieron dos subconjuntos **SpatialPointDataFrame** llamados "**spdf_sel**" y "**spdf_baj**". El primero contiene la información con registros de la selva mediana subperennifolia, mientras que el segundo de los tipos de vegetación de sabana y selva baja inundable. Por último, para visualizar la distribución espacial de las parcelas de muestreo y sus valores de riqueza de especies de estos 2 grupos de datos, se crearon los mapas usando funciones del paquete **tmap**. Y después se guardaron en disco usando la función **tmap_save()**.

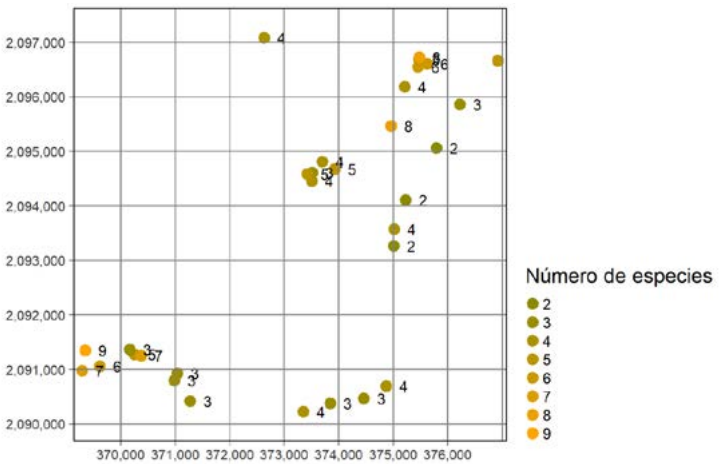
```
# SpatialpointDataFrame por clase de vegetación

spdf_sel = SpatialPointsDataFrame(sp_s.points, datos_selva)
spdf_baj = SpatialPointsDataFrame(sp_b.points, datos_bajo)
```

```
#Mapa de las parcelas de la selva

my_pal <- c("greenyellow", "green","forestgreen","darkgreen")

tm1 <- tm_shape (spdf_sel) +
 tm_dots(col="Num_esp", palette =my_pal,
      title= "Número de especies", size= 0.3)+
 tm_text("Num_esp", just="left", xmod= .5, size = 0.7) +
 tm_grid()+
 tm_layout(legend.outside = TRUE,
       legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
      filename= paste(dir.MAPS,"map_parc_sevas.tiff", sep="/"),
      width=1720, height=1070, asp= 0)
```

```
#Visualizar el mapa de las parcelas de la selva baja inundable

my_pal <- c("yellow4", "orange")

tm1 <- tm_shape (spdf_baj) +
 tm_dots(col="Num_esp", palette =my_pal,
      title= "Número de especies", size= 0.3)+
 tm_text("Num_esp", just="left", xmod= .5, size = 0.7) +
 tm_grid()+
 tm_layout(legend.outside = TRUE,
       legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
          filename= paste(dir.MAPS,"map_parc_bajos.tiff", sep="/"),
          width=1720, height=1070, asp= 0)
```

4.3 Definición del *grid* para realizar las interpolaciones

La mayoría de las funciones que se utilizan en este manual para aplicar los diferentes métodos de interpolación, tienen como salida un objeto *raster*. Esto requiere que primero creemos una cuadrícula *raster* vacía (un *grid*), y luego apliquemos el método de interpolación para calcular los valores en cada uno de los píxeles no muestreados. Por lo anterior, es necesario crear un objeto de tipo **SpatialGrid**, que no debe contener ninguna información. Este objeto se utilizará para almacenar los valores estimados con el método de interpolación particular.

Para crear un objeto **SpatialGrid**, primero se utiliza la función **expand.grid()**. Esta función crea como salida un objeto *dataframe* a partir de una serie de vectores que se proporcionan como parámetros. Se crearon los vectores *x1* y *y1*, de longitud 268, que es el número de renglones y columnas que tendrá el **SpatialGrid**. Para poder obtener estos vectores se utilizó la función **seq()**, la cual genera una secuencia regular de números a partir de valores mínimos y máximos, y utilizando un número de observaciones. Una vez obtenido el *dataframe* "***grid***", se creará un objeto espacial; para ello se utiliza la función **coordinates()**, en donde se definen las coordenadas *x*, *y*. Después se crea un objeto espacial de píxeles y de un objeto **SpatialGrid** usando las funciones **gridded()** y **fullgrid()**. Enseguida se asigna la proyección geográfica del objeto utilizando la función **proj4string()** y se imprime. El objeto "***grid***" es del tipo **SpatialGrid**, con tamaño de píxel de 30 m y con 268 columnas y renglones.

Para conocer los límites de nuestra área de interpolación basada en el archivo *raster* de los tipos de vegetación, podemos usar la función **extent()** para visualizar los valores máximos y mínimos de las coordenadas *x* y *y*.

```
#Visualizar coordenadas máximas y mínimas

extent(rveg)

#class    : Extent
#xmin     : 368985
#xmax     : 377025
```

```
#ymin      : 2089965
#ymax      : 2098005
```

```
#Ajustar límites para tener la parte central

minx1 <- 368985 + 15
maxx1 <- 377025 - 15
miny1 <- 2089965 + 15
maxy1 <- 2098005 - 15

x1 <- seq(minx1, maxx1, length.out = 268)
y1 <- seq(miny1, maxy1, length.out = 268)

grd <- expand.grid(x = x1, y = y1)
coordinates(grd) = ~ x + y

gridded(grd) <- TRUE # Crear un objeto SpatialPixel
fullgrid(grd) <- TRUE # Crear un objeto SpatialGrid
```

```
#Se asignan coordenadas a ese grid

proj4string(grd) <-
 "+proj=utm +zone=16 +units=m +ellps=WGS84 +datum=WGS84 +no_defs"

summary(grd)

# Object of class SpatialGrid
# Coordinates:
#      min    max
# x  368985  377025
# y 2089965 2098005
# Is projected: TRUE
# proj4string :
# [+proj=utm +zone=16 +datum=WGS84 +units=m +no_defs]
```

```
# Grid attributes:
#   cellcentre.offset cellsize cells.dim
# x         369000      30      268
# y        2089980      30      268
```

Para finalizar este capítulo, se creó la función **RMSE**, que permite calcular la raíz cuadrada del error cuadrático promedio.

```
#Función para calcular RMSE

RMSE <- function(observed, predicted){
 sqrt(mean((predicted - observed)^2,
           na.rm=TRUE))
 }
```

5. Métodos globales de interpolación

Los métodos globales de interpolación se utilizan para construir superficies continuas de la variable de interés, que incluyen todas las observaciones del atributo estudiado dentro del área de estudio. Es decir, usan todos los datos disponibles en el área de estudio para calcular la distribución espacial de la variable de interés en un mapa global. Esto significa que, ante un cambio de valor de algún sitio muestreado, se generarán cambios en todo el mapa. Existen diferentes métodos de interpolación espacial que se realizan con todos los datos disponibles, entre ellos se encuentran los análisis de tendencia de superficies y los métodos de clasificación global, que se presentan a continuación.

5.1 Análisis de tendencias de superficies

Este método de interpolación es utilizado cuando el atributo que nos interesa estimar tiene una variación continua en el área de estudio, la cual puede representarse mediante ecuaciones polinomiales. En esta técnica de interpolación, las estimaciones se obtienen a partir de la ubicación geográfica de cada sitio usando las coordenadas *x*, *y*, además de un análisis de regresión múltiple. En modelos polinomiales, los valores del atributo a estimar representan a la variable dependiente y los valores de las coordenadas geográficas son las variables explicativas. El ajuste del modelo de regresión debe cumplir con condiciones de validez, como el que la variable dependiente debe ser lineal y los errores de la regresión se distribuyan de manera normal. Adicionalmente, los datos deben ser independientes de la ubicación de los puntos de muestreo, es decir, no deben existir autocorrelación espacial (Zar, 1999).

El atributo de interés *z* puede ser calculado por un modelo de regresión lineal múltiple, como el siguiente:

$$Z_{xy} = b_0 + b_1x + b_2y + \varepsilon \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Donde b_0 es la ordenada al origen; b_i son los parámetros asociados a las variables explicativas, *x* y *y* son las coordenadas geográficas de ubicación del atributo; mientras que ε es el error aleatorio. Sin embargo, *z* puede representarse como función no lineal de *x* y *y*. Por consiguiente, la estimación de atributos en puntos no medidos se puede obtener mediante modelos polinomiales de segundo o tercer orden. Adicionalmente, la variabilidad de *z* puede depender de otros factores diferentes a la ubicación que se podrían agregar a los modelos, por ejemplo, el clima o la topografía.

En la medida en que se aumenta el número de términos en el modelo de regresión, se podrán encontrar mejores ajustes entre la variable dependiente y el grupo de variables explicativas, por consiguiente, se usa una curva de regresión con mayor complejidad, pero con menor error de estimación (Burrough et al., 1998).

Una de las principales desventajas de este método de interpolación es que las superficies creadas son muy susceptibles a los efectos de los límites. Es decir, al ajustar los datos con ecuaciones de segundo y tercer orden se obtienen superficies cuyos valores caen fuera del área que cubren los puntos, pueden tener valores muy altos o bajos (Burrough et al., 1998).

5.1.1 Polinomios de primer orden

Para obtener una superficie continua a partir de una muestra de datos por medio de un polinomio de primer orden, es necesario definir cuál es la ecuación con la que se obtendrá el modelo de regresión. Entonces se crea el objeto "***formula***" que llamaremos "**f.1**", utilizando la función **as.formula()**. Aquí se utiliza como argumento un objeto de la clase "***formula***", que es una descripción simbólica del modelo de regresión a ajustar con la variable de respuesta y las variables explicativas. Un modelo típico tiene la forma siguiente: variable de respuesta ~ términos. Términos se refiere a una serie de variables que se utilizarán como predictores de la variable de respuesta. En este caso,

el número de especies estará en función de la suma de las variables explicativas, que representan las coordenadas de cada parcela de muestreo (*x*, *y*).

```
#Definir la ecuación con el polinomio de primer orden
f.1 <- as.formula(Num_esp ~ x + y)
```

La función de regresión lineal se ajusta utilizando la función **lm(formula, data, subset, weights, na.action, ...)**, que puede tener una serie de argumentos; en este caso se utilizaron 2 argumentos. En el primero se especifica un objeto de la clase "***formula***", con la descripción del modelo lineal, y el segundo es un objeto del tipo *dataframe* que contiene los vectores de las variables de respuesta y explicativas para ajustar el modelo de regresión. Esta función regresa un objeto de la clase "**lm**"; en este caso se le nombró como "**lm.1**". Para observar el coeficiente de determinación del modelo de regresión ajustado (0.042), así como los coeficientes de los parámetros y la demás información del modelo de regresión, se imprime un resumen del objeto "**lm.1**", con la función **summary()**. Se puede observar que el parámetro asociado a variable *x* difiere de "0" (el signo * muestra que p=0.05, lo que significa que con un valor de alpha mayor a este valor se rechaza la hipótesis nula de que el parámetro sea igual a 0), mientras que, tanto la ordenada al origen, como el parámetro asociado a la variable *y*, no son significativamente diferentes de "0". Sin embargo, a pesar de ser un modelo significativo, el porcentaje de variabilidad explicado por este modelo es del 4.2 %, como lo indica el coeficiente de determinación. Con este modelo tan simple se explica una proporción muy baja de la variabilidad de la variable dependiente. En otras palabras, el modelo no explica una gran proporción de la variable dependiente. Y por lo tanto no es un buen modelo para hacer predicción.

```
#Ejecutar el modelo de regresión

lm.1 <- lm(f.1, data=datos)
summary(lm.1)

#
# Call:
# lm(formula = f.1, data = datos)
```

```
#
# Residuals:
#    Min     1Q  Median     3Q    Max
# -23.347 -14.716   3.937   9.033  25.314
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept) -1.701e+03  1.167e+03  -1.457   0.1476
# x           -1.234e-03  5.678e-04  -2.173   0.0316 *
# y            1.043e-03  5.915e-04   1.764   0.0802 .
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 12.73 on 127 degrees of freedom
# Multiple R-squared:  0.04248, Adjusted R-squared:  0.0274
# F-statistic: 2.817 on 2 and 127 DF,  p-value: 0.06351
```

Una vez que se ajusta el modelo de regresión lineal, se obtendrá una superficie continua estimando valores del número de especies al calcular el valor de la variable de interés con el modelo de regresión en sitios no muestreados. Para ello, se emplea la función **SpatialGridDataFrame()**, en la cual se define una cuadrícula espacial con datos del atributo que pasamos a dicha función. Esta función tiene como argumentos un objeto **SpatialGrid** que llamaremos "***grid***", el cual no contiene datos y fue creado en el Capítulo 4. Por otro lado, contiene como segundo argumento la variable predicha "**var1.pred**" utilizando el modelo lineal ajustado "**lm.1**". Esta función regresa un objeto **SpatialGrid** "**dat.1st**" con la superficie continua que contiene los valores estimados del número de especies en el área de estudio. El *grid* "**dat.1st**" se convierte a un objeto de tipo *raster* utilizando la función **raster()**.

Para excluir las estimaciones del número de especies en las áreas donde no hay vegetación se utilizará la función **mask()** del paquete *raster*. Esta función crea un nuevo objeto *raster* con los mismos valores del objeto *raster*, excepto que para los píxeles del objeto *raster* máscara que están marcados con un valor específico, se actualizarán con un valor nulo. Esto es, la función **mask(x, mask, maskvalue, ...)** se utilizará

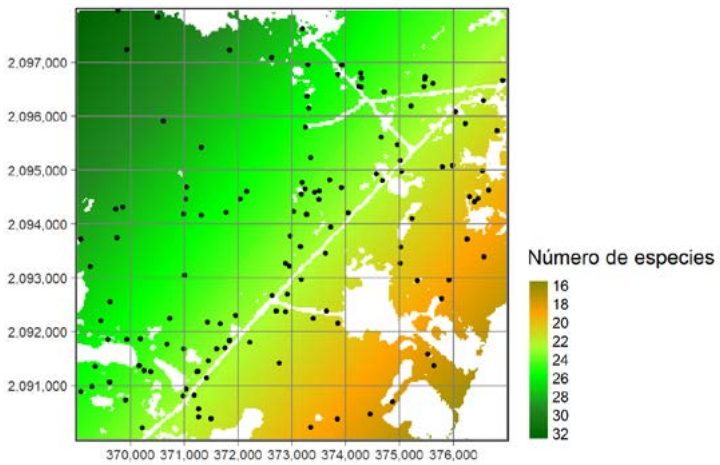
con 3 parámetros: *x* es el objeto *raster* en donde se aplicará la máscara, en este caso el mapa de número de especies "**rgrd**"; **mask** es otro objeto *raster* que es un mapa clasificado con diferentes categorías, en este caso con áreas donde hay vegetación y donde no existe (**rc**, que fue obtenido en el Capítulo 4); y **maskvalue**, que es el valor en el objeto de la máscara, el cual indica los píxeles que serán actualizados en el objeto *raster x* con un valor nulo. En este caso se pondrán en nulo las áreas de no bosque = 0.

```
dat.1st <- SpatialGridDataFrame(
grd, data.frame(var1.pred = predict(lm.1, newdata=grd)))

rgrd <- raster(dat.1st)
grd.mask <- mask(rgrd, rc, maskvalue= 0) #plot(grd.mask)
```

Por último, se grafica el objeto *raster* en donde se aplicó la estimación del número de especies, excluyendo áreas sin vegetación. Para obtener este mapa se utilizan diferentes funciones del paquete **tmap**, de manera similar a como se ha utilizado anteriormente. En este caso se imprimirán de manera conjunta el objeto *raster* "**grd.mask**" con la estimación del número de especies y el objeto **SpatialPointDataFrame** "**spdf**", que contiene la distribución espacial de los sitios de muestreo y que fue creado en el Capítulo 4. Finalmente, este mapa se guarda en disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en disco con el nombre **Pol_1er.tiff** usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow", "green",
          "forestgreen","darkgreen")

tm1 <- tm_shape(grd.mask) +
 tm_raster(style= "cont", n=7, palette =my_col,
          title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid()+
```

```
 tm_layout(legend.outside = TRUE,
           legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
       filename= paste(dir.MAPS,"mapa_pol_1er.tiff", sep="/"),
       width=1720, height=1070, asp= 0)
```

```
#Guardar en disco el mapa interpolado en formato .tiff

writeRaster(grd.mask,
         paste(folder, "Pol_1er.tiff", sep= "/"),
         overwrite = TRUE)
```

5.1.2 Polinomios de segundo orden

El procedimiento para interpolar el número de especies en una superficie continua, usando un polinomio de segundo orden, es similar al procedimiento de los polígonos de primer orden. Primero se define la fórmula con el modelo de regresión lineal en el objeto "***formula***" "**f.2**", usando la función **as.formula()**.

```
#Definir la ecuación con el polinomio de segundo orden

f.2 <- as.formula(Num_esp ~ x + y + I(x^2)+ I(y^2) )
```

Enseguida se ajusta el modelo de regresión utilizando la función **lm()** descrita anteriormente. En este caso se utilizaron 2 argumentos, el objeto "**f.2**" y el *dataframe* "**datos**" que contiene la información para ajustar el modelo de regresión. Esta función regresa un objeto de la clase "**lm**", que en este caso se nombró como "**lm.2**". El coeficiente de determinación en este modelo de regresión ajustado, así como los coeficientes de los parámetros y la demás información del modelo de regresión, se imprime en un resumen con la función **summary()**. Se puede observar que todos los parámetros asociados con las diferentes variables, además de la ordenada al origen difieren de "0". El porcentaje de variabilidad explicado por este modelo es del 17.9 %, como lo indica el coeficiente de determinación.

```
#Ejecutar el modelo de regresión

lm.2 <- lm( f.2, data=datos)
summary(lm.2)
```

```
# Call:
# lm(formula = f.2, data = datos)
#
# Residuals:
#    Min     1Q Median     3Q    Max
# -26.596  -8.088   3.107   8.347 22.468
```

```
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept) -4.511e+06  1.140e+06  -3.956 0.000127 ***
# x            3.948e-01  1.724e-01   2.290 0.023686 *
# y            4.238e+00  1.089e+00   3.890 0.000162 ***
# I(x^2)      -5.315e-07  2.311e-07  -2.300 0.023110 *
# I(y^2)      -1.012e-06  2.601e-07  -3.889 0.000162 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 11.88 on 125 degrees of freedom
# Multiple R-squared:  0.1787,  Adjusted R-squared:  0.1524
# F-statistic:   6.8 on 4 and 125 DF,  p-value: 5.512e-05
```

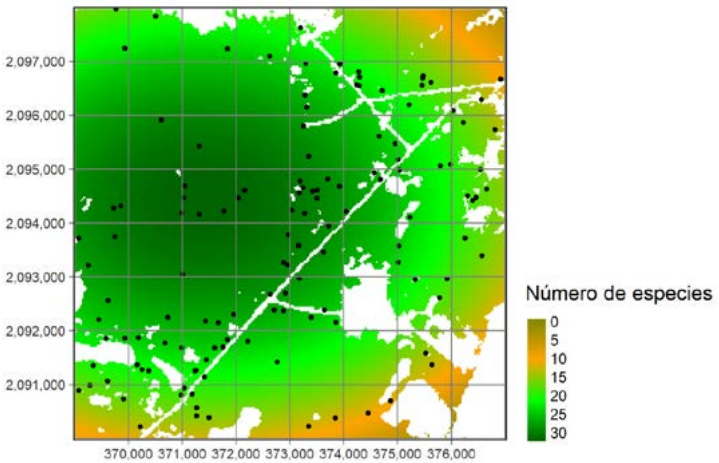
Una vez ajustado el modelo de regresión, se obtuvo una superficie continua estimando valores del número de especies a través de la función **SpatialGridDataFrame()**, que tiene como argumentos un objeto **SpatialGrid** "***grid***", en el que se guardan las estimaciones de la riqueza de especies. El otro argumento es la variable predicha "**var1.pred**" del modelo ajustado "**lm.2**". Esta función regresa un objeto **SpatialGrid** "**dat.2st**" con la superficie continua que contiene los valores estimados del número de especies en el área de estudio. Después, este *grid* es convertido a un objeto de tipo *raster* "**rgrd2**", utilizando la función **raster()**.

Se utiliza la función **mask()** para eliminar estimaciones de la riqueza de especies donde no hay bosque, de manera similar como se hizo para los polinomios de primer orden.

```
dat.2nd <- SpatialGridDataFrame(
 grd, data.frame(var1.pred = predict(lm.2, newdata=grd)))

rgrd2 <- raster(dat.2nd)

grd.mask2 <- mask(rgrd2, rc, maskvalue= 0)
```

Se grafica el objeto *raster* "**grd.mask2**" para obtener un mapa con la riqueza de especies usando la interpolación de polinomios de segundo orden. Se utilizaron las mismas funciones del paquete **tmap** como se hizo con los polinomios de primer orden. Finalmente, este mapa se guarda en disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre **Pol_2nd.tiff**, usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange","green","darkgreen")

tm1 <- tm_shape(grd.mask2) +
 tm_raster(style= "cont", n=10, palette =my_col,
           title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid()+
 tm_layout(legend.outside = TRUE,
       legend.position= c("right", "bottom"))

tm1
```



```
#Grabar mapa en disco

tmap_save(tm1,
      filename= paste(dir.MAPS,"mapa_pol_2nd.tiff", sep="/"),
      width=1720, height=1070, asp= 0)
```

```
#Guardar en disco el mapa creado

writeRaster(grd.mask2,
       paste(folder, "Pol_2nd.tiff", sep= "/"),
       overwrite= TRUE)
```

5.2 Modelos de clasificación

Este enfoque de interpolación utiliza clases o tipos de vegetación como un mecanismo para interpolar los valores medios de los atributos dentro de cada una de las clases, a cualquier punto o ubicación dentro de la clase. El modelo de clasificación es:

$$Z_{ij} = \mu + vt_j + \varepsilon_{ij} \quad \text{.....(2)}$$

Donde Z_{ij} es el valor del atributo estudiado, obtenido en la unidad de muestreo *i* dentro de la clase *j*. El parámetro μ es la media general del atributo estudiado, vt_j es la diferencia entre μ y la media de la clase *j*, y $\boldsymbol{\varepsilon_{ij}}$ es el error aleatorio. La estimación de Z_{ij} está determinada por el valor medio de las observaciones dentro de la clase *j*.

La estimación del atributo estudiado y sus valores medios dentro de cada clase están sujetos a una serie de supuestos, entre los que se incluyen el que los valores del atributo dentro de la clase se distribuyen de manera aleatoria e independiente, es decir, no están espacialmente autocorrelacionados; además de que la varianza del atributo dentro de cada clase debe ser homogénea y todos los cambios del atributo en el espacio ocurren en los límites de las clases de manera abrupta (Burrough et al., 1998).

Se utiliza un análisis de varianza de una vía para probar si existen diferencias significativas de los valores medios del atributo de interés entre las clases. Para proporcionar un indicador de la bondad de la clasificación en la partición de la variabilidad espacial de los valores del atributo estudiado, se utiliza un índice de uniformidad. Este índice se calcula como la varianza dentro de la clase (σ^2_w) dividida entre la varianza total de la muestra (σ^2_t). Este cociente, que es la varianza relativa, es sustraído de 1, como se muestra en la siguiente ecuación:

$$U = 1 - \left(\sigma^2_W / \sigma^2_T\right) \quad \text{.....(3)}$$

Los valores de uniformidad se encuentran en el rango de 0 a 1, los cuales pueden ser fácilmente convertidos a porcentaje de uniformidad. Cuando el valor del índice es cercano a 0, la varianza dentro de la clase es de tamaño considerable y se aproxima a la varianza total. Lo cual indica que la contribución de la clase a la partición de la variabilidad total no es útil para discriminar el atributo entre las clases consideradas. En cambio, si el valor de la uniformidad es cercano a 1, las clases son relativamente homogéneas y distinguen de manera efectiva el atributo entre ellas.

Entre las desventajas de este método se encuentran el que se estima únicamente un valor medio del atributo para toda la clase, es decir, la variación dentro de clases es ignorada en este modelo. Por otro lado, la precisión de la predicción del atributo estudiado está limitada por la bondad de la clasificación.

Para estimar la riqueza de especies, a partir de una muestra de datos utilizando una predicción del valor medio de cada clase, es necesario verificar que el modelo de clasificación pueda ser utilizado para esta estimación. Se requiere probar que los valores de las medias de la riqueza de especies difieran entre clases de vegetación y que la varianza dentro de clases sea considerablemente menor que la varianza total. Iniciamos con un estudio descriptivo de la riqueza de especies en las 6 clases de vegetación estudiadas. Aquí se utiliza la librería "**stats**" de R. Se calcula el promedio y la desviación estándar de la riqueza de especies por clase de vegetación, utilizando la función **tapply()**. Esta función tiene como primer argumento un vector "**datos$Num_esp**" con los datos de la riqueza de especies; el segundo argumento es otro vector,

"**datos$t_veg**", que contiene el tipo de vegetación de cada elemento del primer vector. Por último, el tercer argumento es la función que se aplicará, en este caso fue "**mean**" para calcular la media y "**sd**" para la desviación estándar.

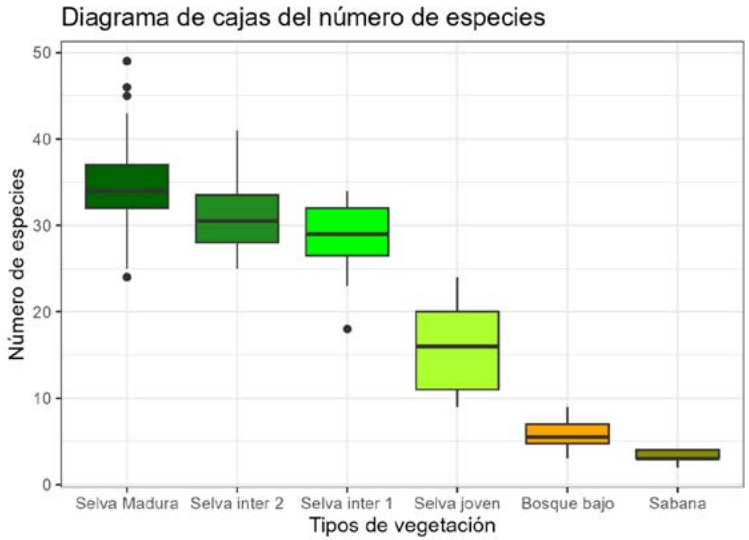
```
#Estadísticas descriptivas por tipo de vegetación

tapply(datos$Num_esp,datos$t_veg,mean)

#       1       2       3       4       5       6
# 34.756098 30.863636 28.684211 15.647059  5.687500  3.133333

tapply(datos$Num_esp,datos$t_veg,sd)

#       1       2       3       4       5       6
# 5.3655405 4.2008554 4.1103208 4.7162111 1.7783419 0.7432234
```

Después, se elabora un diagrama de cajas que contiene los valores de la mediana, así como el percentil del 25 y 75 % de la variable de interés. Esta gráfica se puede obtener utilizando diferentes opciones. La primera es utilizar la función **boxplot()**. Esta función tiene varios parámetros, uno es un objeto "***formula***" donde la variable dependiente es "**Num_esp**" y la variable explicativa es "**t_veg**". Otro parámetro es un *dataframe* donde se encuentran los datos a graficar "**datos**". El resto de los parámetros se utilizan para indicar el color de la gráfica, el título general y el título del eje *y*. En la segunda opción se utiliza la función **ggplot()** de la librería **ggplot2**. Al principio se coloca la función **ggplot()**, la cual tiene como parámetros el conjunto de datos a graficar; en este caso es un dataframe "**datos**" y la función **aes()**. Esta función describe cómo las variables en los datos se asignan a las propiedades visuales; en este caso se colocaron los vectores que van en los ejes *x* y *y*, además se menciona la variable que va a estar en la leyenda. Enseguida se colocan uno o más niveles de funciones que definen el tipo de gráficas y su visualización, las cuales están separadas por el signo "**+**". Las funciones que se utilizaron en esta gráfica fueron: **geom_boxplot()**, que especifica el tipo de gráfica; **scale_x_discrete()**, para renombrar las etiquetas del eje *x*; **labs()**, para especificar las etiquetas del título y los ejes; la función **scale_fill_manual()**, para asignar colores en la clase; y la función **theme_bw()**, para imprimir la cuadrícula al

interior de la gráfica. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**vb**", que después se imprime.

Para poder grabar en disco esta gráfica con el paquete **ggplot2**, primero se define el directorio en donde se guardará; el nombre del directorio es "**Maps**" y está ubicado en "**C:/met_int/Maps/**". Posteriormente, se utiliza la función **ggsave()** para grabar la gráfica en el disco. Esta función puede tener varios parámetros, aquí se utilizaron 4. En el primero se indica el objeto que se debe guardar en disco, que es un objeto de tipo **ggplot** llamado "**vb**"; el segundo corresponde al nombre de archivo, incluida la extensión y la ruta. Después se especifican el largo y ancho de la imagen que se imprimirá.

```
#Diagrama de cajas

#Diagrama de caja simple

boxplot(Num_esp ~ t_veg, data=datos, col="gray", main="Número de especies", xlab="T
Vegetacion")

#diagrama de cajas con ggplot

vb <- ggplot(datos, aes(x=as.factor(t_veg), y = Num_esp,
           fill= as.factor(t_veg))) +
 geom_boxplot(show.legend = FALSE)+
 #scale_y_discrete()+
 scale_x_discrete(labels = c("Selva Madura","Selva inter 2",
           "Selva inter 1", "Selva joven",
           "Bosque bajo", "Sabana"))+
 labs(title="Diagrama de cajas del número de especies",
     x="Tipos de vegetación",
     y = "Número de especies")+
 scale_fill_manual(values = c("darkgreen", "forestgreen", "green",
                 "greenyellow", "orange", "yellow4"))+
 theme_bw()
vb
```




```
#Grabar gráfica

ggsave(vb,
    file=paste(dir.MAPS,"gra_box_nesp.tiff", sep="/"),
    width=5.5, height=4)
```

En los resultados del estudio descriptivo se puede ver que las medias de la riqueza de especies entre clases de vegetación son diferentes. Sin embargo, se debe probar si esas diferencias son estadísticamente significativas utilizando un análisis de varianza de una vía. Antes de realizar este análisis, se deben ejecutar las pruebas de validez de este. En este caso, se requiere que las distintas poblaciones en las que se prueban las diferencias de medias provengan de una distribución normal. Se utilizó la prueba de normalidad de Shapiro-Wilk para cada una de las clases de vegetación, en donde la hipótesis nula supone que la población se distribuye de manera normal y la regla de decisión es rechazar la hipótesis nula si el valor observado de significancia (**p**) es menor que alpha (el nivel de significancia establecido por el investigador o investigadora). Para ejecutar la prueba de normalidad se utiliza la función **shapiro.test()** junto con la función **tapply()**, en la cual se especifican el vector de

datos "**datos$Num_esp**", el vector con los tipos de vegetación que tienen cada registro de la riqueza de especies "**datos$t_veg**" y la función por aplicar. Los resultados muestran que, en la mayoría de los 6 tipos de vegetación, el valor de **p** no es menor a 0.05 (alpha). Por lo tanto, no existe evidencia para rechazar la hipótesis nula y se asume normalidad.

```
#Prueba de normalidad

tapply(datos$Num_esp,datos$t_veg,shapiro.test)

# $`1`
#
#   Shapiro-Wilk normality test
#
# data:  X[[i]]
# W = 0.97213, p-value = 0.4035
#
#
# $`2`
#
#   Shapiro-Wilk normality test
#
# data:  X[[i]]
# W = 0.94644, p-value = 0.268
#
#
# $`3`
#
#   Shapiro-Wilk normality test
#
# data:  X[[i]]
# W = 0.91852, p-value = 0.1062
#
```

```
#
# $`4`
#
#   Shapiro-Wilk normality test
#
# data:  X[[i]]
# W = 0.93397, p-value = 0.2534
#
#
# $`5`
#
#   Shapiro-Wilk normality test
#
# data:  X[[i]]
# W = 0.95695, p-value = 0.607
#
#
# $`6`
#
#   Shapiro-Wilk normality test
#
# data:  X[[i]]
# W = 0.81667, p-value = 0.006065
```

El análisis de varianza de una vía se ejecuta con la función **aov()**, con los siguientes parámetros: un objeto "***formula***" donde se especifica la variable dependiente "**Num_esp**" y la variable explicativa "**t_veg**", de la cual nos aseguramos que se utilice como factor, es decir, una variable no numérica. Además, del *dataframe* donde están los datos "**datos**". Los resultados de aplicar este análisis de varianza se guardan en un objeto de tipo **aov**, que nombramos como "**model**". Por último, se utiliza **summary()**, que tiene como argumento un objeto del tipo **aov**. En esta prueba, la hipótesis nula indica que las medias de los 6 tipos de vegetación son iguales, contra la hipótesis alternativa de que al menos una media difiere de las demás. La regla de decisión es rechazar la hipótesis nula si el valor de p < alpha. Los resultados muestran una tabla

de análisis de varianza con un valor de p = 2e-16, el cual es menor que cualquier alpha. Por lo tanto, se rechaza la hipótesis nula y la prueba es estadísticamente significativa. Esto quiere decir que, al menos, una media difiere de las demás.

```
#Análisis de varianza

(model<-aov(Num_esp ~ as.factor(t_veg), data = datos))

# Call:
#    aov(formula = Num_esp ~ as.factor(t_veg), data = datos)
#
# Terms:
#                 as.factor(t_veg) Residuals
# Sum of Squares         19245.16   2237.31
# Deg. of Freedom               5       124
#
# Residual standard error: 4.247685
# Estimated effects may be unbalanced

#anova(model)
summary(model)

#                Df Sum Sq Mean Sq F value Pr(>F)
# as.factor(t_veg)   5  19245    3849   213.3 <2e-16 ***
# Residuals        124   2237     18
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Para saber qué tipos de vegetación tuvieron diferencias en cuanto al valor promedio de la riqueza de especies, se utilizó la prueba de Tukey con la función **TukeyHSD()**. Esta función utilizó como parámetro el objeto de tipo **aov** "**model**" y arroja como resultado un objeto de tipo **Tukey** al que llamamos "**prueba**". Al imprimir este objeto se muestra una comparación múltiple entre los valores medios de la riqueza de especies entre pares de tipos de vegetación. Los resultados muestran que la mayoría de los valores

medios de riqueza de especies difieren en la mayoría de los tipos de vegetación. Solamente entre los pares de tipos de vegetación 3-2 y 6-5, el valor de p es mayor que 0.05 (0.5749883 y 0.5520836, respectivamente). Entre esas clases no existen diferencias significativas en los valores medios de la riqueza de especies.

```
#Pruebas Post HOC: Tukey

prueba<-TukeyHSD(model)
prueba

#   Tukey multiple comparisons of means
#     95% family-wise confidence level
#
# Fit: aov(formula = Num_esp ~ as.factor(t_veg), data = datos)
#
# $`as.factor(t_veg)`
#          diff       lwr        upr     p adj
# 2-1  -3.892461  -7.142101  -0.6428216 0.0092336
# 3-1  -6.071887  -9.484406  -2.6593679 0.0000145
# 4-1 -19.109039 -22.656076 -15.5620011 0.0000000
# 5-1 -29.068598 -32.693144 -25.4440508 0.0000000
# 6-1 -31.622764 -35.333198 -27.9123307 0.0000000
# 3-2  -2.179426  -6.030412   1.6715605 0.5749883
# 4-2 -15.216578 -19.187256 -11.2458993 0.0000000
# 5-2 -25.176136 -29.216204 -21.1360685 0.0000000
# 6-2 -27.730303 -31.847599 -23.6130071 0.0000000
# 4-3 -13.037152 -17.142199  -8.9321042 0.0000000
# 5-3 -22.996711 -27.168913 -18.8245076 0.0000000
# 6-3 -25.550877 -29.797906 -21.3038482 0.0000000
# 5-4  -9.959559 -14.242486  -5.6766314 0.0000000
# 6-4 -12.513725 -16.869577  -8.1578737 0.0000000
# 6-5  -2.554167  -6.973364   1.8650309 0.5520836
```

En cuanto a la uniformidad interna de las clases de vegetación para la riqueza de especies, calculada como el **índice U**, los resultados también confirman que las clases de vegetación tuvieron una uniformidad interna relativamente alta (U = 0.895). Esto indica que el mapa de vegetación generado a partir de la clasificación supervisada es útil para dividir la variabilidad espacial del número de especies.

```
#Bondad de la clasificación U = 1- (varianza dentro de grupos/varianza total)

U <- 1-(2237.3/21482.5)
U

# [1] 0.8958548
```

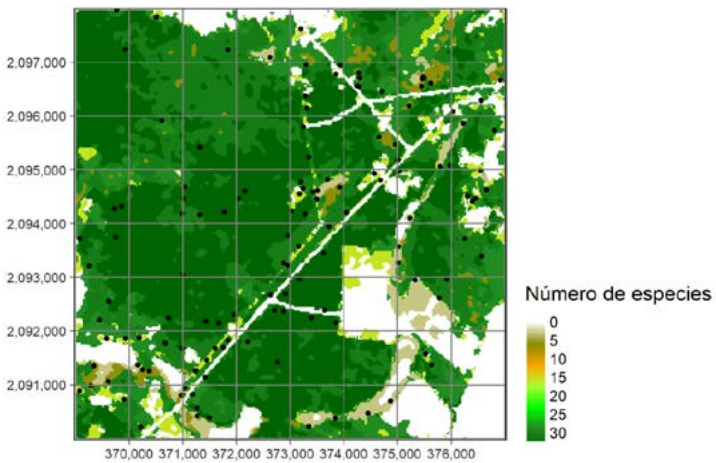
Una vez que se confirma que el modelo de clasificación puede ser utilizado para la interpolación, se procede a crear el mapa. Para ello, se realizó una reclasificación de las coberturas terrestres usando la función **reclassify()**, con el objeto de tipo *raster* que tiene los tipos de vegetación "**clases**", y con una matriz de reclasificación con los valores medios de cada tipo de vegetación.

```
#Mapa de diversidad para cada tipo de vegetación, todos los
#valores > 0 y <= 1 son igual 0, etc.

m1 <- c(0, 1, 0 , 1, 2, 34.8, 2,3, 30.9, 3, 4,
     28.7, 4, 5, 15.7, 5, 6, 5.6, 6, 7, 3.1)
mbiomat <- matrix(m1, ncol=3, byrow=TRUE)

mbio <- reclassify(rveg, mbiomat, right = TRUE)
```

Para obtener este mapa se utilizan diferentes funciones del paquete **tmap**, de manera similar a como se ha utilizado anteriormente. En este caso se imprimirán de manera conjunta el objeto *raster* "**mbio**" con la estimación del número de especies y el objeto **SpatialPointDataFrame** "**spdf**", que contiene la distribución espacial de los sitios de muestreo. Finalmente, este mapa se guarda en el disco usando la función **tmap_**

save(). Además, se guarda el objeto *raster* en el disco con el nombre **Clas_glob.tiff**, usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("white", "yellow4", "orange", "greenyellow",
        "green","forestgreen","darkgreen")

tm1 <- tm_shape(mbio) +
 tm_raster(style= "cont", n=6, palette =my_col,
        title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid()+
 tm_layout(legend.outside = TRUE,
        legend.position= c("right", "bottom"))

tm1
```



```
#Grabar mapa en disco

tmap_save(tm1,
        filename= paste(dir.MAPS,"mapa_clas_glob.tiff", sep="/"),
        width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(mbio,
        paste(folder, "Clas_glob.tiff", sep= "/"),
        overwrite= TRUE)
```

6. Métodos locales de interpolación

Los métodos locales de interpolación son procedimientos que se aplican de manera repetida en subconjuntos de datos, lo cual significa que el cambio de información en alguno de los sitios de muestreo no afecta a todo el mapa, sino exclusivamente a la porción donde se realiza la estimación. Existe una gran diversidad de métodos de interpolación local. En este capítulo se revisan los más utilizados en la literatura.

6.1 Polígonos de Thiessen

Este procedimiento de interpolación construye polígonos alrededor de cada sitio o punto de muestreo. Además, todos los puntos estimados dentro del polígono tienen el mismo valor que el punto con el que fue construido. Por lo tanto, la estimación del atributo de interés en sitios no medidos se realiza a partir de los valores de los puntos más cercanos.

Este método de interpolación está basado en vectores. Cuando las observaciones de sitios de muestreo se encuentran distribuidas en el espacio de manera regular, como en los muestreos sistemáticos, se forma una malla de cuadrados regulares de lados iguales. Sin embargo, cuando la distribución de los puntos de muestreo es irregular, se produce una red irregular de polígonos (Burrough et al., 1998; Webster et al., 2001).

Los polígonos de Thiessen proporcionan mejores resultados cuando se dispone de una gran cantidad de sitios de muestreo para llevar a cabo la interpolación. Por otro

lado, este procedimiento se ha utilizado de manera exitosa cuando la variable de interés tiene valores categóricos. Sin embargo, una gran desventaja de este método es que no se considera la variabilidad dentro de los polígonos y los cambios de la variable estimada ocurren en los límites de estos.

Para realizar las interpolaciones con polígonos de Thiessen, se deben abrir algunas librerías en las cuales se encuentran las funciones utilizadas en este módulo; estas librerías fueron descritas en la sección 1.3.

```
library(spatstat) # Usada para la función dirichlet
library(maptools)
library(rgeos)
```

La función **dirichlet()** crea una superficie con un conjunto de polígonos a partir de un conjunto de puntos en el espacio. Aquí se utilizaron los puntos que se encuentran en el objeto "**spdf**" del tipo **SpatialPointDataFrame**, que fue creado en el capítulo de preparación de datos. Este conjunto de polígonos no está georreferenciado, por lo que se asigna la proyección geográfica que tiene el mapa base de los tipos de vegetación, utilizando la función **proj4string()**. Debido a que esta superficie de polígonos no almacena la información del objeto "**spdf**", se utiliza la función **over()** para sobreponer esta información a los polígonos. La función **over()** usa como argumentos un objeto del tipo **SpatialPolygons**, en este caso "**Poli**", además del objeto con la información de los puntos "**spdf**" y el valor medio de todos los puntos dentro de un polígono. De esta sobreposición se genera el objeto **Poli.z** que es trasformado a un objeto del tipo **SpatialPolygonsDataFrame** nombrado como "**Poli.spdf**", usando la función del mismo nombre.

```
#Crear una superficie teselada, la función requiere la librería maptools

Poli <- as(dirichlet(as.ppp(spdf)), "SpatialPolygons")

#plot(Poli)
```

```
#La función dirichlet no contiene información sobre la proyección.
#Se agrega manualmente

proj4string(Poli) <-
 "+proj=utm +zone=16 +units=m +ellps=WGS84 +datum=WGS84 +no_defs"
```

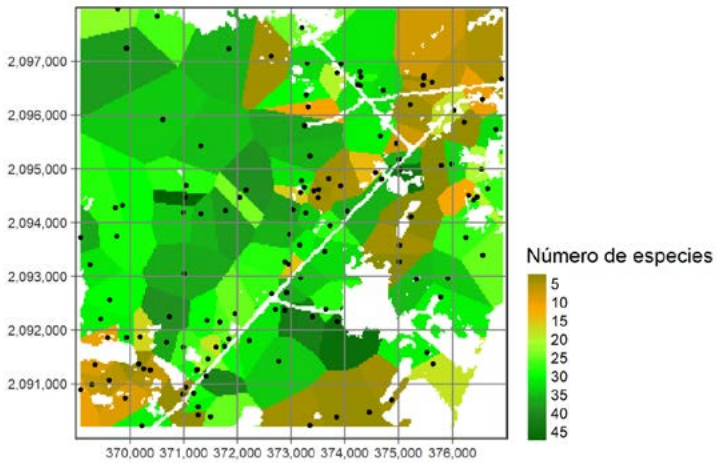
```
#Una superficie teselada no almacena los atributos de una capa
# de puntos. Se utiliza la función over()
#(del paquete sp) para juntar los atributos de puntos y de la
# superficie teselada por medio de un join.
#La función over() crea un objeto que luego necesita ser agregado
# al objeto "Poli". Con esto se crea un #objeto SpatialPolygonsDataFrame

Poli.z <- over(Poli, spdf, fn=mean)
Poli.spdf <- SpatialPolygonsDataFrame(Poli, Poli.z)
```

Con el objeto de utilizar la máscara para eliminar valores de riqueza de especies en sitios donde no hay vegetación, los polígonos con datos de estimación de la riqueza de especies, guardados en "**Poli.spdf**", se convierten a formato *raster* utilizando la función **rasterize()**. Para ello, se utilizaron los siguientes argumentos: el objeto "**Poli.spdf**"; un objeto *raster* "**r**" sin datos, el cual fue previamente convertido de formato *grid* "***grid***" a un objeto de tipo *raster*; y la variable "**Num_esp**". Esta función permite obtener un objeto de tipo *raster* guardado en "**rPoli**". Después, se eliminan los valores donde no hay vegetación utilizando la función **mask()**, como se ha hecho anteriormente.

```
#Convertir el grid a raster

r <- raster(grd)
rPoli <- rasterize(Poli.spdf, r, 'Num_esp') # rasterizar los poligonos
Poli.mask <- mask(rPoli, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**Poli.mask**" para obtener el mapa de riqueza de especies, utilizando las funciones de la librería **tmap**, como se hizo anteriormente. Finalmente, este mapa se guarda en disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre **thiessen.tiff**, usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
            "green","forestgreen","darkgreen")

tm1 <- tm_shape(Poli.mask) +
  tm_raster(style= "cont", n=10,palette =my_col,
            title="Número de especies") +
  tm_shape(spdf) +
  tm_dots(size=0.05) +
  tm_grid()+
  tm_layout(legend.outside = TRUE,
            legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
      filename= paste(dir.MAPS,"mapa_Thissen.tiff", sep="/"),
      width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado
writeRaster(Poli.mask,
        paste(folder, "thiessen.tiff", sep= "/"),
        overwrite= TRUE)
```

6.2 Distancia Inversa Ponderada

Este método de interpolación combina información del vecino más cercano con el cambio gradual. Es decir, las estimaciones del atributo estudiado en ubicaciones donde no se tiene un valor, se obtienen como el promedio ponderado de los valores de sus vecinos más cercanos. La contribución de los vecinos más cercanos al valor estimado del atributo es mayor que la de sitios más alejados. Dicho de otra manera, los vecinos más cercanos tienen mayor peso o importancia. Estos pesos están construidos en función inversa a la distancia de los vecinos. La ecuación de esta técnica de interpolación es la siguiente:

$$Z = \sum_{i=1}^{n} \left(\frac{1/d_i^p}{\sum 1/d_i^p} Z_i \right) \quad \text{.....(4)}$$

En donde *Z* representa el valor estimado del atributo de interés; *Zi* es el valor del atributo calculado en el sitio *i*; *d* es la distancia entre el sitio y el lugar donde se realiza la estimación; *p* es un parámetro de potencia; y *n* representa el número de sitios medidos usados para la estimación.

El valor del parámetro de potencia *p*, tiene una fuerte influencia en la precisión de las estimaciones del atributo en esta técnica de interpolación (Webster et al., 2001). El valor de 2 es el comúnmente más utilizado, aunque se pueden utilizar diferentes parámetros de potencia. Por otro lado, se recomienda utilizar valores de *p* enteros, dado que los valores menores a 1 son más cercanos a la estimación de una media aritmética simple (Webster et al., 2001).

A pesar de que la distancia inversa tiene la ventaja de ser un método simple y de fácil procesamiento, tienen problemas con los puntos aislados, los cuales tienden a producir patrones conocidos como "huevos de pato". Una solución sugiere utilizar diferentes

vecindades dependiendo de la densidad de sitios de muestreo, mediante el uso de radios de búsqueda pequeños en altas densidades y radios de búsqueda grandes en bajas densidades de puntos (Burrough et al., 1998).

Las funciones para llevar a cabo este método de interpolación se encuentran en la librería "**gstat**", por lo cual debe ser abierta.

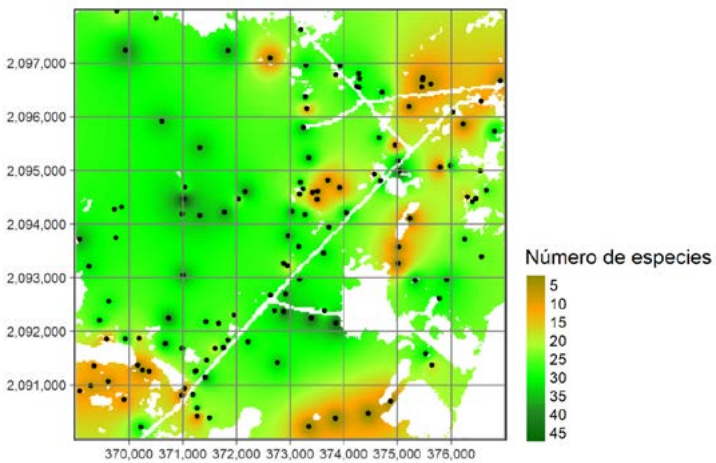
```
#El paquete gstat tiene las rutinas para calcular IDW
library(gstat)
```

Para llevar a cabo una interpolación con el método de distancia inversa, se utilizó la función **idw()** del paquete "**gstat**". Esta función tiene varios parámetros **idw(formula, data, newdata, ...,)**, aquí se utilizaron solamente 4. Primero se requiere un objeto "***formula***" que define la variable dependiente como en un modelo lineal; la variable dependiente tiene el nombre "**Num_esp**", y para la distancia inversa se utiliza la fórmula **Num_esp~1**. Después se ingresan los datos para llevar a cabo la interpolación, los cuales están almacenados en un objeto del tipo **SpatialPointDataFrame** "**spdf**". Enseguida se especifica un objeto de clase *grid* que contiene las ubicaciones donde se guardarán los valores predichos. Por último, idp=2.0 indica que se utilizará un valor de potencia de 2. Como resultado de esta función se genera un *grid* "**Map.idw**" con las estimaciones de la riqueza de especies en toda el área. Este *grid* es convertido a formato *raster* usando la función **raster(),** y después se eliminan los valores donde no hay vegetación utilizando la función **mask()**.

```
#Para hacer la función de distancia inversa se necesita un objeto Spatial#PointDataFrame.
Usaremos #"spdf"Interpolar el grid usando un valor de #potencia de 2 (idp=2.0)

Map.idw <- gstat::idw(Num_esp ~ 1, spdf, newdata=grd, idp=2.0)

# [inverse distance weighted interpolation]

rgrd.iwd <- raster(Map.idw)
grd.mask3 <- mask(rgrd.iwd, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**grd.mask3**" para obtener el mapa de riqueza de especies, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**mapa_dist_inv2.tiff**", usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
        "green","forestgreen","darkgreen")

tm1 <- tm_shape(grd.mask3) +
 tm_raster(style= "cont", n=10,palette =my_col,
      title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid()+
 tm_layout(legend.outside = TRUE,
       legend.position= c("right", "bottom"))

tm1
```



```
#Grabar mapa en disco

tmap_save(tm1,
        filename= paste(dir.MAPS,"mapa_dist_inv2.tiff", sep="/"),
        width=1720, height=1070, asp= 0)
```

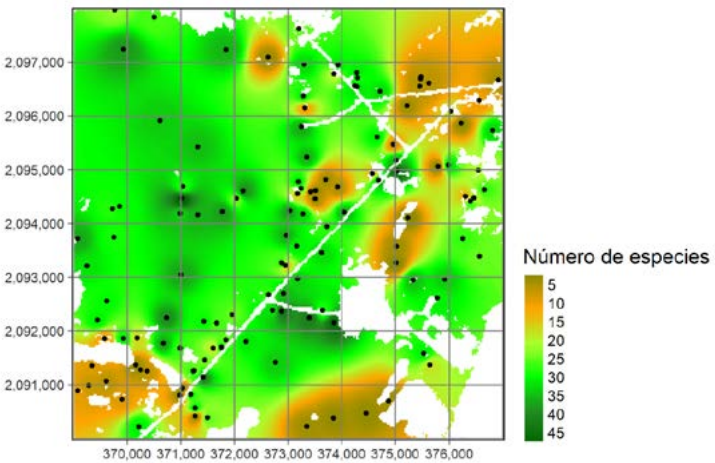
```
#Guardar en el disco el mapa creado

writeRaster(grd.mask3,
        paste(folder, "Dis_inv2.tiff", sep= "/"),
        overwrite= TRUE)
```

Se utiliza la función **idw()** para estimar la riqueza de especies usando el método de funciones de distancia inversa utilizando un parámetro de potencia igual a 3. Los argumentos utilizados fueron los mismos que se emplearon anteriormente, con la diferencia del último parámetro **idp=3.0**. Como resultado de esta función, se genera un *grid* "**Map1.idw**" con las estimaciones de la riqueza de especies en toda el área. Este *grid* es convertido a formato *raster* usando la función **raster()** y después se eliminan los valores donde no hay vegetación utilizando la función **mask()**.

```
#Interpolar el grid usando un valor de potencia de 3 (idp=3.0)

Map1.idw <- gstat::idw(Num_esp ~ 1, spdf, newdata=grd, idp=3.0)

# [inverse distance weighted interpolation]

rgrd1.iwd <- raster(Map1.idw)
grd1.mask3 <- mask(rgrd1.iwd, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**grd1.mask3**" para obtener el mapa de riqueza de especies, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**Dis_inv3.tiff**", usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
            "green","forestgreen","darkgreen")

tm1 <- tm_shape(grd1.mask3) +
 tm_raster(style= "cont", n=10,
           palette =my_col,
           title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid() +
 tm_layout(legend.outside = TRUE,
           legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco
tmap_save(tm1,
      filename= paste(dir.MAPS,"mapa_dist_inv3.tiff", sep="/"),
      width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(grd1.mask3,
        paste(folder, "Dis_inv3.tiff", sep= "/"),
        overwrite= TRUE)
```

El desempeño de las técnicas de interpolación puede ser medido en términos de la precisión de las estimaciones, a través de las desviaciones entre los datos medidos y sus correspondientes valores estimados. En este caso, se utilizó el procedimiento de validación cruzada, que consiste en que los valores de sitios medidos son eliminados del conjunto de datos, particularmente uno a la vez, pero podrían ser más de uno. Entonces, la técnica de interpolación es ejecutada con los sitios restantes para estimar el valor del sitio eliminado. Con ambos procedimientos se produce una lista de valores estimados del atributo de interés, que son comparados con aquellos obtenidos en los sitios de muestreo (Webster et al., 2001).

La estadística utilizada para comparar el desempeño de los mapas interpolados fue la raíz de la media de los errores al cuadrado (raíz del error cuadrático medio, pero usamos las siglas en inglés **RMSE**). El **RMSE** es una medida de la precisión de la predicción. Esta estadística es sensitiva a puntos de influencia, es decir, tiende a enfatizar los errores más grandes. Debido a esto último, el **RMSE** es una medida del error más conservadora que el error medio absoluto. Los valores del error calculados con **RMSE**, se obtienen de:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{Z}(x_i) - Z(x_i)\right)^2} \quad \text{..... (5)}$$

Donde *n* es el número de sitios a validar y $\hat{z}(x_i)$ **y** $z(x_i)$ son los valores estimados y observados en el sitio *i*, respectivamente.

Para llevar a cabo la validación cruzada en la estimación que usa un valor de p=2.0, primero se crea un vector "**IDW.out**" que tiene el mismo número de elementos que el objeto "**spdf**" de la clase **SpatialPoindDataFrame**. Después se realiza la interpolación usando la función **idw()**, excluyendo un dato a la vez. Esto permite obtener

una lista de valores estimados de la riqueza de especies que se almacena en el vector "**IDW.out**", que serán comparados con aquellos medidos en los sitios de muestreo.

Se utiliza "**for()**" para realizar la permutación de cada uno de los datos en "**spdf**", se extrae el valor predicho mediante el aparatado de "**var1.pred**" que se almacena dentro del objeto resultante de la función "**idw**". Posteriormente, este vector se guarda en el *dataframe* donde se encuentran los datos como "**datos$predd2**". Enseguida se calcula el valor del **RMSE** usando el vector de los datos estimados "**datos$predd2**" y el de los observados "**datos$Num-esp**". Por último, se redondea este valor a 2 dígitos y se guarda en la variable "**RMSE_i**". Adicionalmente, se ajusta un modelo de regresión lineal entre los valores observados y predichos, usando la función **lm()**; para ello, se utilizó la fórmula **datos$predd2 ~ datos$Num_esp**. El valor del coeficiente de determinación $\mathbf{R^2}$ es igual a 0.30.

```
#Validación cruzada para las funciones de distancia inversa p=2

IDW.out <- vector(length = length(spdf))

for (i in 1:length(spdf)) {
IDW.out[i] <- idw(Num_esp ~ 1, spdf[-i,], spdf[i,],
          idp=2.0)$var1.pred
}

# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
...

# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
```

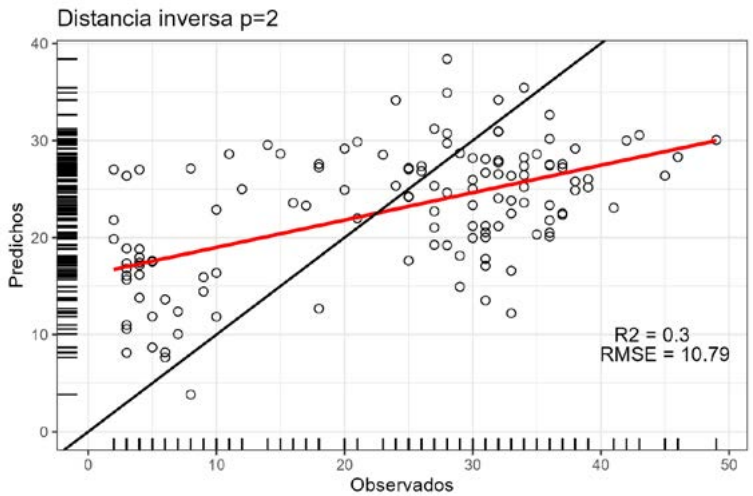
```
datos$predd2 <- IDW.out
```

```
#Cálculo del RMSE p=2

idwrmse <- RMSE(datos$Num_esp, datos$predd2)

RMSE_i <- round(idwrmse, digits = 2)
```

```
#Modelo de regresión entre predichos vs. observados

mod = lm(datos$predd2 ~ datos$Num_esp)
summary(mod)

# Call:
# lm(formula = datos$predd2 ~ datos$Num_esp)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -14.5855  -4.1876   0.2401   3.2955  14.3405
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)   16.1617     1.0201  15.843  < 2e-16 ***
# datos$Num_esp   0.2822     0.0381   7.405 1.55e-11 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 5.585 on 128 degrees of freedom
# Multiple R-squared:  0.2999,  Adjusted R-squared:  0.2945
# F-statistic: 54.84 on 1 and 128 DF,  p-value: 1.55e-11

r2 = format(summary(mod)$r.squared, digits = 2)
```

Para obtener de manera gráfica estos resultados, se utiliza la función **ggplot()** de la librería **ggplot2**. Al principio se coloca la función **ggplot()**, que tiene como parámetros el vector de valores predichos "**predd2**" y el de valores observados "**Num_esp**" del *data-frame* "**datos**". Las funciones que se utilizaron en esta gráfica fueron: **geom_point()** para el diagrama de dispersión; **labs()** para especificar las etiquetas del título y los ejes; la función **geom_abline()** para colocar la línea 1:1, y la función **geom_smooth()** para la línea de regresión. Para colocar las anotaciones con los valores del coeficiente de determinación y el **RMSE**, se usa la función **annotate()**. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Gráfica de los valores predichos vs. valores observados IDW2

p1 <- ggplot(datos, aes(x=Num_esp, y=predd2)) +
 geom_point(size=2, shape=21 )+geom_rug()+geom_smooth(method=lm, se=FALSE,
size=1,color="red")+
 labs(title= "Distancia inversa p=2",
    x="Observados", y = "Predichos")+
 expand_limits(x = 0, y = 0)+
 theme_bw() +
 geom_abline(size=0.7) +
 annotate("text", x = 44, y = 10,
       label = paste("R2 =", r2), size = 4)+
 annotate("text", x = 45, y = 8,
       label = paste("RMSE =", RMSE_i), size = 4)

p1
```




```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_dis_Idw2.tiff", sep="/"),
    width=6, height=4)
```

El procedimiento de validación cuando se usa el valor de p=3.0, es similar al descrito anteriormente (ver descripción del código de arriba). El valor del coeficiente de determinación $\mathbf{R}^2$ es igual a 0.39.

```
#Validación cruzada para las funciones de distancia inversa p=3

IDW1.out <- vector(length = length(spdf))

for (i in 1:length(spdf)) {
IDW1.out[i] <- idw(Num_esp ~ 1, spdf[-i,],
            spdf[i,], idp=3.0)$var1.pred
}

# [inverse distance weighted interpolation]
```

```
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
...
```

```
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]

datos$predd3 <- IDW1.out
```

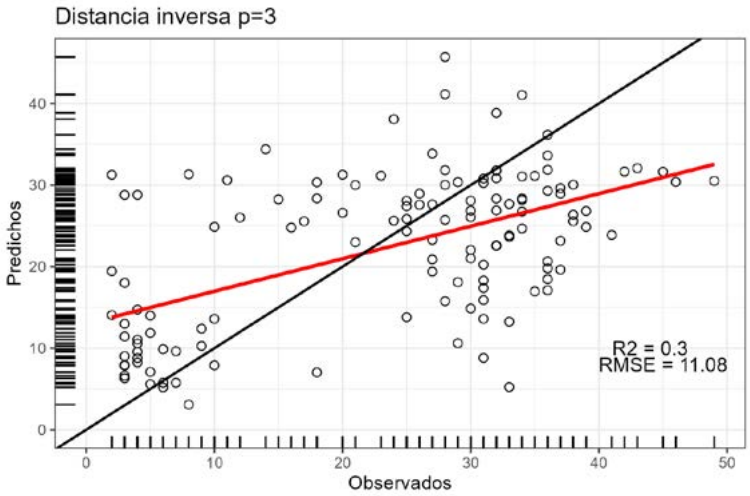
```
#Cálculo del RMSE p=3

idwrmse1 <- RMSE(datos$Num_esp, datos$predd3)
RMSE_i <- round(idwrmse1, digits = 2)
```

```
#Modelo de regresión entre predichos vs. observados p=3

mod1 = lm(datos$predd3 ~ datos$Num_esp)
summary(mod1)

# Call:
# lm(formula = datos$predd3 ~ datos$Num_esp)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -20.9210  -6.0600   0.1754   5.3357  21.5489
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  12.99104    1.44602   8.984 2.88e-15 ***
# datos$Num_esp  0.39889    0.05401   7.385 1.72e-11 ***
```

```
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 7.916 on 128 degrees of freedom
# Multiple R-squared:  0.2988,  Adjusted R-squared:  0.2933
# F-statistic: 54.54 on 1 and 128 DF,  p-value: 1.722e-11

r2 = format(summary(mod1)$r.squared, digits = 2)
```

Para obtener de manera gráfica estos resultados, se utiliza la función **ggplot()** de la librería **ggplot2**. Esto de forma similar como se hizo en la interpolación de distancia inversa para p=2.0 (ver descripción del código).

```
#Gráfica de los valores predichos vs. valores observados IDW3

p1 <- ggplot(datos, aes(x=Num_esp, y=predd3)) +
 geom_point(size=2, shape=21 )+
 geom_rug( )+
 geom_smooth(method=lm, se=FALSE, size=1,color="red")+
 labs(title= "Distancia inversa p=3",
    x="Observados", y = "Predichos")+
expand_limits(x = 0, y = 0)+
theme_bw( ) +
 expand_limits(x=c(0,45), y=c(0, 45)) +
 annotate("text",x = 44, y = 10,
       label = paste("R2 =", r2), size = 4)+
 annotate("text",x = 45, y = 8,
      label = paste("RMSE =", RMSE_i), size = 4)+
 geom_abline(size=0.7)

p1
```




```
#Grabar gráfica

ggsave(p1,
       file=paste(dir.MAPS,"gra_dis_Idw3.tiff", sep="/"),
       width=6, height=4)
```

6.3 Interpolación con Kriging

Con la técnica de interpolación denominada Kriging, se obtienen estimaciones de los valores del atributo de interés en ubicaciones no muestreadas, a partir de la información proporcionada por la estructura de dependencia espacial del atributo estudiado. Es decir, el grado de dependencia espacial que presentan las observaciones en relación con sus vecinos. Esta dependencia espacial se representa con una función de semivarianza del atributo estudiado. La semivarianza se calcula de la siguiente expresión:

$$\gamma(h) = \frac{1}{2n}\sum_{i=1}^{n}\left(Z(x_i) - Z(x_i+h)\right)^2 \quad \text{..... (6)}$$

Donde $Z(x_i)$ es el valor del atributo de interés medido en la parcela *i*, $Z(x_i + h)$ es el valor del atributo medido en otras parcelas separadas de x_i por una distancia discreta *h*; *n* representa el número de pares de observaciones separados por *h*, y $\gamma\ (h)$ es el valor de semivarianza estimado para todos los pares de parcelas en una distancia *h*.

Se calculan las semivarianzas para cada posible par de parcelas de muestreo, y los valores medios de las semivarianzas se grafican para intervalos de distancia crecientes (*h*); con esto se obtiene el semivariograma experimental (observar los puntos en la **Figura 3**). Posteriormente, para realizar predicciones, el semivariograma experimental es convertido a uno teórico por medio del ajuste de un modelo estadístico. Los parámetros de los semivariogramas teóricos son obtenidos matemáticamente por medio del ajuste de diferentes modelos que relacionan la distancia *h* con la semivarianza. Los modelos más comunes son el esférico, exponencial, gaussiano y lineal. Los modelos ajustados tienen los siguientes parámetros: la varianza total, también conocida como "**sill**", la cual se define como los valores asintóticos de la semivarianza y está dividida en dos: la varianza que representa la dependencia espacial y la varianza aleatoria o varianza "**nugget**". Esta última, refleja la variación espacial a distancias más cortas que la mínima separación que existe entre dos sitios de muestreo o bien, la varianza no explicada por los modelos. El rango es la máxima distancia en la cual el atributo es espacialmente dependiente (**Figura 3**) (Burrough et al., 1998; Webster et al., 2001). El coeficiente de determinación ($\mathbf{R^2}$) resultante del ajuste por mínimos cuadrados de los modelos a los semivariogramas experimentales y los procedimientos de validación cruzada, se utilizan como criterios para seleccionar los mejores modelos.

Las estimaciones del atributo de interés se obtienen por medio de la siguiente expresión:

$$Z(x_0) = \sum_{i=1}^{n} \lambda_i Z(x_i) \quad (7)$$

Donde λ_i son los pesos óptimos seleccionados para minimizar la estimación de la varianza. La suma de estos debe ser igual a 1 a partir de un conjunto de n + 1, ecuaciones lineales simultáneas (Webster et al., 2001); $Z(x_i)$ son los valores del atributo en los sitios medidos; y $Z(x_0)$ es la estimación no sesgada del atributo de interés.

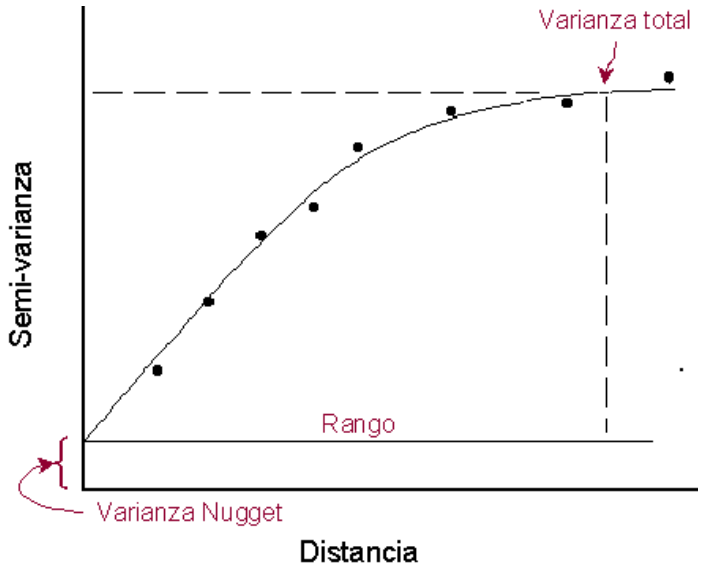


Figura 3. *Semivariograma experimental y teórico. Parámetros de los semivariogramas teóricos.*

Para llevar a cabo este procedimiento de interpolación, primero se abre la librería "**gstat**", que es en donde se encuentran las funciones para ejecutar el método de interpolación con Kriging.

```
#El paquete gstat tiene las rutinas para calcular variogramas y Kriging

library(gstat)
```

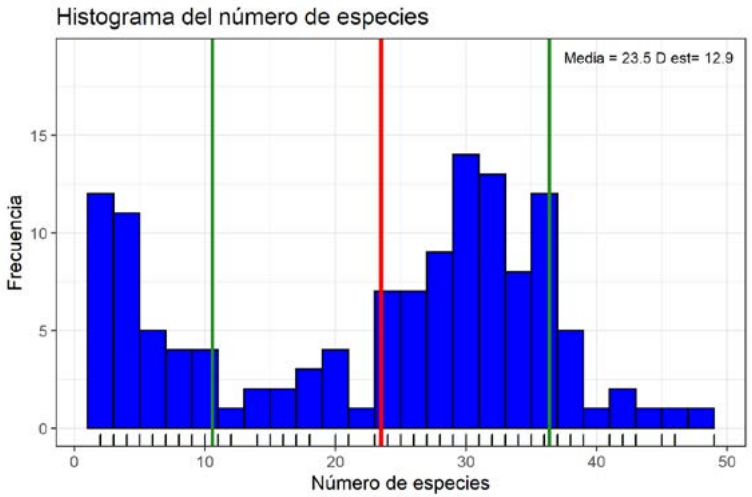
Para ejecutar este procedimiento de interpolación es necesario que la variable a estudiar tenga una distribución normal. En caso contrario, se requerirá de hacer una trasformación de la variable usando logaritmos, el inverso o la raíz cuadrada. Para verificar la normalidad de la variable se obtuvo un histograma con la distribución de frecuencias de la riqueza de especies.

La gráfica del histograma se obtuvo usando la función **ggplot()** de la librería **ggplot2**. Se colocó la función **ggplot()**, que tiene como parámetros el vector "**Num_esp**" del

dataframe "**datos**". Las funciones que se utilizaron en esta gráfica fueron: **geom_histogram ()**, indicando que la gráfica es de un histograma; **geom_vline()** para imprimir líneas horizontales con el valor de la media y la desviación estándar; **labs()**, que especifica las etiquetas del título y los ejes; anotaciones con los valores de la media y la desviación estándar con **annotate()**, entre otras. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Estadísticas

med <- mean(datos$Num_esp) #Media
m_n <- round(med, digits = 1)
sd <- sd(datos$Num_esp)
s_n <- round(sd, digits = 1)
```

```
#Revisar si la variable a interpolar tiene una distribución normal. Si no
#existe normalidad, intentar una trasformación log() o sqrt()

p1 <- ggplot(datos, aes(x=Num_esp)) +
 geom_histogram(fill="blue", color="black", binwidth = 2          )+
 geom_rug()+
 geom_vline(aes(xintercept=mean(Num_esp)), color="red", lwd=1.2)+
 geom_vline(aes(xintercept=mean(Num_esp)+
           sd(Num_esp)), color="forestgreen", lwd=1)+
 geom_vline(aes(xintercept=mean(Num_esp)-sd(Num_esp)),
        color="forestgreen", lwd=1)+
 annotate("text",x = 44, y = 19,
      label = paste("Media =", m_n,"D est=", s_n), size = 3)+
 labs(title="Histograma del número de especies",
    x="Número de especies", y = "Frecuencia")+
 theme_bw()

p1
```




```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_hist1-6.tiff", sep="/"),
    width=6, height=4)
```

El histograma de la riqueza de especies muestra una tendencia bimodal en los datos, lo cual sugiere que existen dos grupos separados que analizaremos en el siguiente capítulo. Por otro lado, se utilizó la prueba de normalidad de Shapiro-Wilk con la función **shapiro.test()**. Debido a que el valor de p es menor a 0.05 (alpha), se rechaza la hipótesis nula y la normalidad. Esto puede deberse a que la distribución de frecuencias es bimodal, por lo que se decide no realizar una trasformación de la variable.

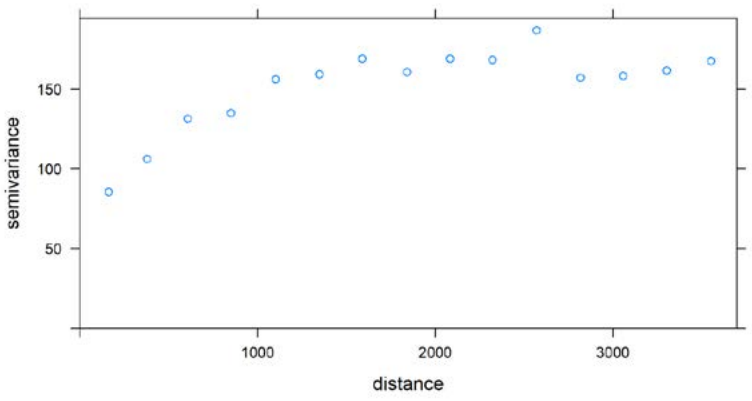
```
#Prueba de normalidad

shapiro.test(datos$Num_esp)

#
#   Shapiro-Wilk normality test
#
```

```
# data:  datos$Num_esp
# W = 0.90143, p-value = 9.219e-08
```

El procedimiento de interpolación con Kriging inicia con el cálculo del variograma experimental utilizando la función **variogram()** del paquete **gstat**. Esta función puede tener varios argumentos, en un primer variograma se utilizaron 2 argumentos: un objeto de tipo "***formula***", que especifica la variable dependiente y las posibles covariables. En este caso, como no existen covariables, se colocó el número 1; y un objeto de la clase **SpatialPointDataFrame** "**spdf**" con los datos. Esta función regresa un objeto de la clase **variogram** que nombramos como "**ns.vgm**", el cual se imprimió con la función **plot()**.

Se utilizó la función **tiff()** para almacenar este gráfico en un archivo de tipo **TIFF**, que significa Tag Image File Format. En esta función se especifica el directorio y nombre del archivo como se ha hecho anteriormente, el nombre del archivo es "**var_krig1.tiff**", y el del directorio "**C:\met_int\Maps**". Se especifican, además, el ancho y alto de la imagen y sus unidades, así como la resolución que tendrá la misma en dpi's.

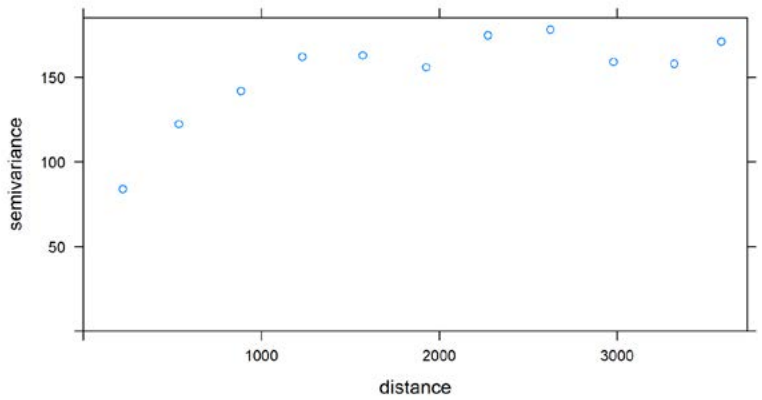
```
#Crear el variograma experimental

ns.vgm <- variogram(Num_esp ~ 1, spdf)
plot(ns.vgm)
```



```
#Guardar la gráfica

tiff(paste(dir.MAPS,"var_krig1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns.vgm)
dev.off()
```

En un segundo variograma experimental, se usó la función **variogram()** con 3 argumentos: la fórmula donde se especifica la variable dependiente; el objeto "**spdf**" con los datos y el ancho de los intervalos de distancia en los que se agrupan los pares de puntos de datos para estimaciones de semivarianza, que en este caso fue de 350. Es decir, se redujo el número de intervalos para tener un mejor ajuste del modelo. Esta función regresa un objeto de la clase **variogram** que nombramos como "**ns.vgm**", el cual se imprimió con la función **plot()** y se guardó en el disco con la función **tiff()**.

```
#Crear el variograma

ns.vgm <- variogram(Num_esp ~ 1, spdf, width = 350)
plot(ns.vgm)
```



```
#Guardar la gráfica

tiff(paste(dir.MAPS,"var_krig2.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns.vgm)
dev.off()
```

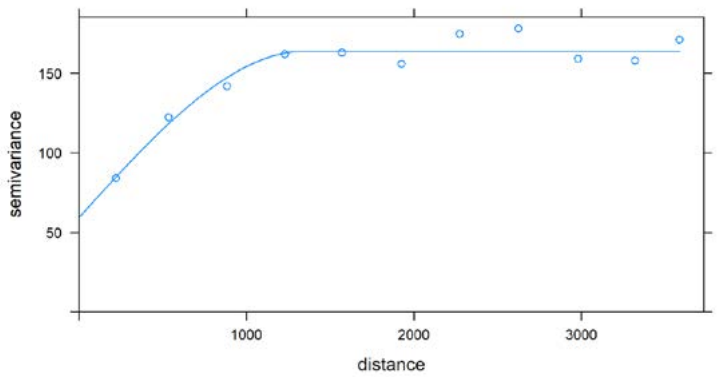
El siguiente paso consiste en ajustar un modelo al variograma experimental. Para ello se utiliza la función **fit.variogram(object, model, ...)**, que tiene diferentes argumentos, pero en este caso se utilizaron 2: un objeto de tipo **variogram** "**ns.vgm**", que fue la salida al ejecutar la función **variogram()**; y el modelo, que se especifica por medio de la función **vgm()**. Esta función genera un modelo de semivariograma y tiene diferentes argumentos: un valor de referencia del "**sill**", en este caso fue de 40; el tipo de modelo "**Exp**", "**Sph**", "**Gau**", "**Mat**"; un valor que indique el rango y si el componente "**nugget**" es omitido o debe ser parte del modelo, se colocó 1 indicando que existe "**nugget**" en el modelo. Como resultado, la función **fit.variogram()** regresa un objeto al cual llamamos "**ns.fit**" y se imprime.

Se ajustó un modelo esférico al semivariograma experimental. La varianza estructural, que determina la dependencia espacial explicada por el modelo y calculada como (varianza total– varianza nugget/varianza total) *100, fue (104.71/164.03) *100 y tuvo un valor de 63.8 %. Esto significa que una parte de la variabilidad de la distribución espacial de la riqueza de especies no es explicada por el modelo (36.2 %).

El modelo ajustado se imprime con la función **plot()** y se guardó en el disco con la función **tiff()**. Adicionalmente, se calculó el coeficiente de determinación del modelo $\mathbf{R^2}$ que fue igual a 0.99, indicando un buen ajuste del modelo.

```
#Shp esférico, (Exp) exponencial, (Gau) Gaussiano
#La función vgm() tiene como parámetros sill, modelo, rango y nugget

ns.fit <- fit.variogram(ns.vgm,
          model = vgm(40, "Sph", 1500, 1))
ns.fit
```

```
#  model    psill   range
# 1  Nug  59.30618   0.000
# 2   Sph 104.71517 1350.112

plot(ns.vgm, ns.fit)
```




```
#Guardar la gráfica

tiff(paste(dir.MAPS,"var_fitkrig1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns.vgm, ns.fit)
dev.off()
```

```
#Cálculo del coeficiente de determinación del modelo ajustado
#SSErr es la suma ponderada de los cuadrados de los residuales

SSErr<-attr(ns.fit,"SSErr")
```

```
#SStot suma total de cuadrados ponderados

weig<-ns.vgm$np / ns.vgm$dist^2
```

```
#Se utiliza objeto generado en la función fit() de gstat.
#El método por default usa N_h/h^2 con N_h el número de pares de
#puntos y h es la distancia.

SStot<- sum(weig*(ns.vgm$gamma - mean(ns.vgm$gamma))^2)

(R2<-1-SSErr / SStot) #Coeficiente de determinacion

# [1] 0.9924446
```

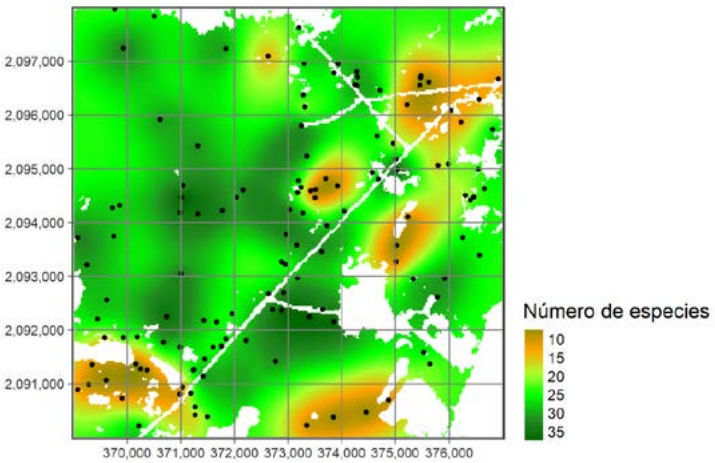
El mapa de riqueza de especies usando interpolación con Kriging, se obtiene utilizando la función **krige()** de la librería **gstat**. De la función **krige(formula, data, grid, model, ...)**, aquí se utilizaron solamente 4 argumentos: la fórmula que define la variable dependiente, y el valor de 1 cuando se ejecuta un Kriging ordinario; el objeto que contiene los datos "**spdf**"; un *grid* donde se almacenan los datos y el modelo de semivariograma ajustado "**ns.fit**". Esta función regresa un *grid* con el mapa interpolado, el cual se nombró como "**ns.k**". Este *grid* es convertido a formato *raster* usando la función **raster()** y después se eliminan los valores donde no hay vegetación utilizando la función **mask()**.

```
#Interpolación con Kriging

ns.k = gstat::krige(Num_esp ~ 1, spdf, grd,
          model = ns.fit)

# [using ordinary kriging]
```

```
rgrd.k <- raster(ns.k)

grd.mask4 <- mask(rgrd.k, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**grd.mask4**" para obtener el mapa de riqueza de especies, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**mapa_kriging.tiff**", usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
            "green","forestgreen","darkgreen")

tm1 <- tm_shape(grd.mask4) +
  tm_raster(style= "cont", n=10,palette =my_col,
          title="Número de especies") +
  tm_shape(spdf) +
  tm_dots(size=0.05) +
  tm_grid() +
  tm_layout(legend.outside = TRUE,
          legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
        filename= paste(dir.MAPS,"mapa_kriging.tiff", sep="/"),
        width=1720, height=1070, asp= 0)
```

```
#Guardar en disco el mapa creado

writeRaster(grd.mask4,
        paste(folder, "Kriging.tiff", sep= "/"),
        overwrite= TRUE)
```

Una de las ventajas del procedimiento de interpolación con Kriging, es que se puede obtener la varianza de las estimaciones. Para obtener un mapa con la varianza de las estimaciones, se utilizó la función **raster()** para transformar de formato *grid* a *raster* el objeto "**ns.k**" obtenido en la interpolación con Kriging, solo que en esta ocasión se utilizó la variable "**var1.var**", que es la varianza de las estimaciones. Se obtuvo un *raster* con la desviación estándar de la varianza utilizando la función **sqrt()**. Se eliminaron los valores donde no hay vegetación utilizando la función **mask()**.

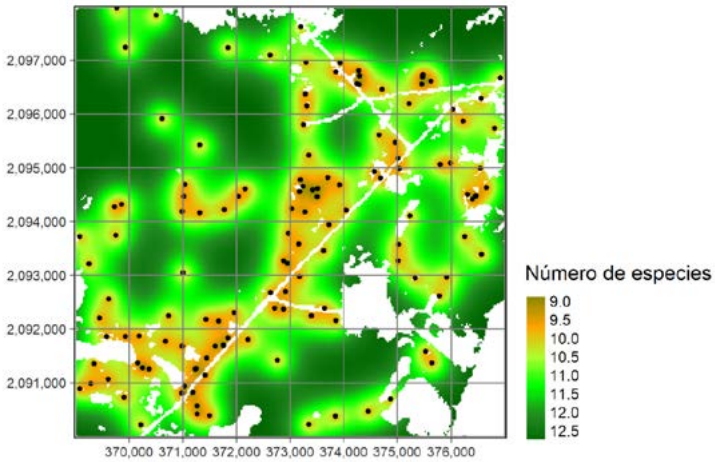
```
#Desviación estándar de las interpolaciones con Kriging. Obtener
#el mapa de desviación estándar
```

```
rgrd_var.k <- raster(ns.k, layer = "var1.var")
rgrd_sd.k <- sqrt(rgrd_var.k)
grd.mask5 <- mask(rgrd_sd.k, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**grd.mask5**" para obtener el mapa de desviación estándar de las estimaciones de la riqueza de especies, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**SD_Kriging.tiff**", usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
          "green","forestgreen","darkgreen")

tm1 <- tm_shape(grd.mask5) +
 tm_raster(style= "cont", n=10,
```

```
        title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid() +
 tm_layout(legend.outside = TRUE,
        legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
      filename= paste(dir.MAPS,"mapa_sd_kriging.tiff", sep="/"),
      width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(grd.mask5,
       paste(folder, "SD_Kriging.tiff", sep= "/"),
       overwrite= TRUE)
```

Para llevar a cabo la validación cruzada de la estimación con Kriging, primero se crea un vector "**KRG.out**", que tiene el mismo número de elementos que el objeto "**spdf**" de la clase **SpatialPoindDataFrame**. Después se realiza la interpolación usando la función **krige()**, excluyendo un dato a la vez. Esto permite obtener una lista de valores estimados de la riqueza de especies que se almacena en el vector "**KRG.out**", que serán comparados con aquellos medidos en los sitios de muestreo. Posteriormente, este vector se guarda en el *dataframe* donde se encuentran los datos como "**datos$predk**". Enseguida, se calcula el valor del **RMSE** usando el vector de los datos estimados "**datos$predk**" y el de los observados "**datos$Num-esp**". Por último, se redondea este valor a 2 dígitos y se guarda en la variable "**RMSE_i**". Adicionalmente, se ajusta un modelo de regresión lineal entre los valores observados y predichos usando la función **lm()**. Para ello, se utilizó la fórmula **datos$predk ~ datos$Num_esp**. El valor del coeficiente de determinación $\mathbf{R^2}$ es igual 0.31.

```
#Validación cruzada para Kriging

KRG.out <- vector(length = length(spdf))

for (i in 1:length(spdf)) {
KRG.out[i] <- krige(Num_esp ~ 1, spdf[-i,],
          spdf[i,],
          model = ns.fit)$var1.pred
}

# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
.

.

.
```

```
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]

datos$predk <- KRG.out
```

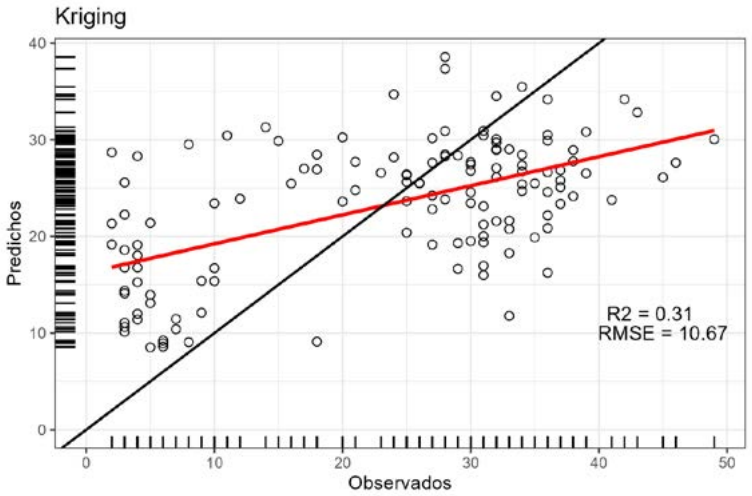
```
#Cálculo del RMSE

krgrmse <- RMSE(datos$Num_esp, datos$predk)
RMSE_i <- round(krgrmse, digits = 4)
```

```
#Modelo de regresión entre predichos vs. observados

mod <- lm(datos$predk ~ datos$Num_esp)
summary(mod)

#
# Call:
# lm(formula = datos$predk ~ datos$Num_esp)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -14.3634  -4.3007   0.0141   3.6749  13.9284
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  16.22322    1.05704  15.348  < 2e-16 ***
# datos$Num_esp  0.30059    0.03948   7.613 5.15e-12 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 5.787 on 128 degrees of freedom
```

```
# Multiple R-squared:  0.3117,  Adjusted R-squared:  0.3063
# F-statistic: 57.96 on 1 and 128 DF,  p-value: 5.15e-12

r2 = format(summary(mod)$r.squared, digits = 2)
```

Para obtener de manera gráfica estos resultados se utiliza la función **ggplot()** de la librería **ggplot2**. Al principio se coloca la función **ggplot()**, que tiene como parámetros el vector de valores predichos "**predk**" y el de valores observados "**Num_esp**" del *dataframe* "**datos**". Se utilizaron funciones similares a como se hizo la validación cruzada con distancia inversa. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Gráfica de los valores predichos vs. valores observados

p1 <- ggplot(datos, aes(x=Num_esp, y=predk)) +
 geom_point(size=2, shape=21 )+
 geom_rug( )+
 geom_smooth(method=lm, se=FALSE, size=1,color="red")+
 labs(title= "Kriging",x="Observados", y = "Predichos")+
 expand_limits(x = 0, y = 0)+
 theme_bw( ) +
 geom_abline(size=0.7) +
 annotate("text",x = 44, y = 12,
       label = paste("R2 =", r2), size = 4)+
 annotate("text",x = 45, y = 10,
       label = paste("RMSE =", RMSE_i), size = 4)

p1
```




```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_kriging.tiff", sep="/"),
    width=6, height=4)
```

7. Métodos de interpolación con datos auxiliares

Una manera de incrementar la precisión de la distribución espacial de los atributos estudiados es con el uso de información auxiliar. Frecuentemente, se dispone de información adicional al atributo de interés, la cual puede ser utilizada junto con la distancia inversa y Kriging para mejorar la precisión de las estimaciones. Las principales fuentes de información que se pueden tener son una apropiada estratificación del área de estudio, que nos permita obtener clases con varianzas más homogéneas o bien, el uso de una variable asociada con la variable de interés. Estas variables son más fáciles de medir o se dispone de ellas de manera continua en toda el área de estudio. Para ello, se puede utilizar un modelo lineal o no lineal que asocie la variable de interés con las variables auxiliares y se combine con la dependencia espacial de los residuales (Burrough et al., 1998). En este capítulo se presentan tres métodos de interpolación espacial que utilizan datos auxiliares derivados de imágenes de satélite: interpolación dentro de estratos, regresión con Kriging y Random Forest con Kriging.

7.1 Interpolación dentro de estratos

La interpolación dentro de estratos se utiliza cuando existen diferencias significativas en los valores medios de un atributo estudiado entre las diferentes clases o tipos de vegetación. Es decir, se encuentran clases que son más homogéneas reduciendo la variabilidad interna, pero difieren con las demás clases. Una manera apropiada para realizar la estimación del atributo de interés es utilizar la variabilidad explicada dentro de clases e introducir esa información *a priori* en combinación con otro procedi-

miento de interpolación como el Kriging o distancia inversa. De esta forma se puede mejorar la precisión de las estimaciones de dicho atributo.

7.1.1 Distancia inversa dentro de estratos

En este procedimiento de interpolación espacial se aplica una interpolación de distancia inversa para cada uno de los estratos considerados. En este caso se consideraron 2 estratos: el primero que tiene aquellas áreas cubiertas con selva mediana subperennifolia, incluyendo las 4 etapas de sucesión, y el segundo corresponde a áreas cubiertas con sabana y bosque bajo inundable.

Primero se abre la librería "**gstat**", que contiene las funciones para llevar a cabo este método de interpolación. La interpolación usando distancia inversa utiliza la función **idw(formula, data, newdata, ...)**, la cual tienen varios argumentos. En esta apartado se utilizaron solamente 4 argumentos: una fórmula que define la variable dependiente "**Num_esp**" y para la distancia inversa se utiliza el número 1; los datos para llevar a cabo la interpolación, los cuales están almacenados en un objeto del tipo **SpatialPointDataFrame**, se utilizó "**spdf_sel**" para llevar a cabo la interpolación en la selva mediana subperennifolia y "**spdf_baj**" para los otros dos tipos de vegetación; un objeto de clase *grid* que especifica las ubicaciones donde se guardarán los valores predichos; e idp=2.0 indicando que se utilizará un valor de potencia de 2. Como resultado de aplicar estas funciones, se generan dos objetos del tipo *grid* "**idw.sel**" e "**idw.baj**", respectivamente para la selva y los otros dos tipos de vegetación. Estos *grid* son convertidos a formato *raster* usando la función **raster()** y después se eliminan los valores donde no hay vegetación utilizando la función **mask()** y sus respectivas máscaras "**rs**", que indica áreas con selva mediana subperennifolia, y "**rb**", que es la máscara para identificar áreas cubiertas con sabana y bosque bajo inundable. También se define el argumento "updatevalue=0" para asignar un valor de 0 y no NA a los valores enmascarados.

```
#Interpolar el grid usando un valor de potencia de 2 (idp=2.0) para cada estrato

library(gstat)
```

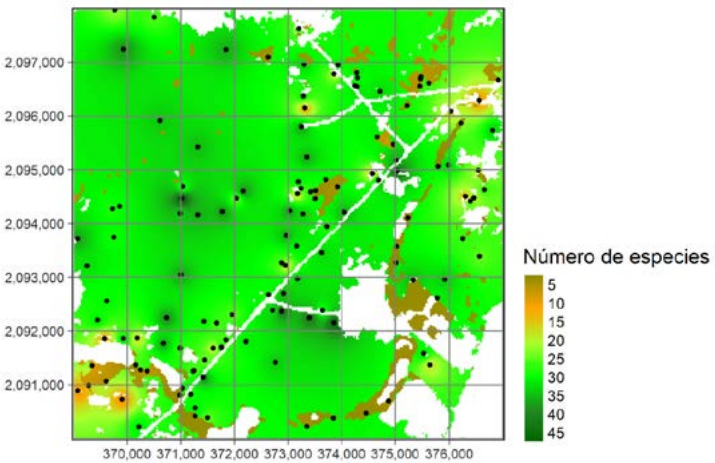
```
idw.sel <- gstat::idw(Num_esp ~ 1, spdf_sel, newdata=grd, idp=2.0)

# [inverse distance weighted interpolation]

idw.baj <- gstat::idw(Num_esp ~ 1, spdf_baj, newdata=grd, idp=2.0)

# [inverse distance weighted interpolation]

rgrd.iwd_sel <- raster(idw.sel)
grd.masks <- mask(rgrd.iwd_sel, rs, maskvalue= 0, updatevalue = 0)

rgrd.iwd_baj <- raster(idw.baj)
grd.maskb <- mask(rgrd.iwd_baj, rb, maskvalue= 0, updatevalue = 0)
```

Enseguida se suman los objetos *raster* generados para obtener un solo objeto de toda el área de estudio, que incluya los dos estratos considerados. A este nuevo objeto *raster* "**grd.masksb**", se le aplicaron las máscaras que corresponde a todos los tipos de vegetación. Esto con el objeto de eliminar los valores donde no hay vegetación o que no pudieron ser eliminados antes; para esto se usa la función **mask()**.

```
#Función para remplazar valores nulos con 0 en raster

grd.masksb <- grd.masks + grd.maskb
grd.mask_todos <- mask(grd.masksb, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**grd.mask_todos**" para obtener el mapa de riqueza de especies, con la técnica de interpolación de distancia inversa dentro de estratos, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**Dis_inv_est.tiff**", usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
        "green","forestgreen","darkgreen")

tm1 <- tm_shape(grd.mask_todos) +
 tm_raster(style= "cont", n=10,
        palette =my_col,
 title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid() +
 tm_layout(legend.outside = TRUE,
        legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
      filename= paste(dir.MAPS,"mapa_iwd_est.tiff", sep="/"),
      width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(grd.mask_todos,
        paste(folder, "Dis_inv_est.tiff", sep= "/"),
        overwrite= TRUE)
```

Para llevar a cabo la validación cruzada, se crean dos vectores "**IDW_out_sel**" e "**IDW_out_baj**", que tienen el mismo número de elementos que los objetos "**spdf_sel**" y "**spdf_baj**", respectivamente, y que son de la clase **SpatialPoindDataFrame**. La interpolación se realiza usando la función **idw()**, excluyendo un dato a la vez. Esto permite obtener una lista de valores estimados de la riqueza de especies que se almacena en los vectores creados y que serán comparados con los valores observados de riqueza de especies en los sitios de muestreo. Posteriormente, estos vectores se concatenan usando la función **c()** y se guardan en un vector que llamamos "**IDW.out_est**". Para poder comparar este vector con los datos estimados, habrá que colocar las observaciones en el mismo orden. Para ello se crean el vector "**a**" y el vector "**b**" con los datos de la riqueza de especies para las selvas y para los otros dos tipos de vegetación, respectivamente. Los vectores "**a**" y "**b**" se concatenan en un solo vector, para después guardar los valores observados y estimados en un *dataframe* "**datos2**".

```
#Validación cruzada para las funciones de distancia inversa dentro de
#estratos

IDW_out.sel <- vector(length = length(spdf_sel))

for (i in 1:length(spdf_sel)) {
 IDW_out.sel[i] <- idw(Num_esp ~ 1, spdf_sel[-i,],
                spdf_sel[i,],
                idp=2.0)$var1.pred
 }

# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
```

```
# [inverse distance weighted interpolation]
...

# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]

IDW_out.baj <- vector(length = length(spdf_baj))

for (i in 1:length(spdf_baj)) {
 IDW_out.baj[i] <- idw(Num_esp ~ 1,
          spdf_baj[-i,], spdf_baj[i,],
          idp=2.0)$var1.pred
 }

# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
...

# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
# [inverse distance weighted interpolation]
```

```
IDW.out_est <- c(IDW_out.sel, IDW_out.baj)

a <- spdf_sel$Num_esp
b <- spdf_baj$Num_esp

obs <- c(a, b)
datos2<- data.frame(obs, IDW.out_est)
```

Se calcula el valor del **RMSE** usando el vector de los datos estimados "**datos2$IDW.out_est**" y el de los observados "**datos2$obs**". Por último, se redondea este valor a 2 dígitos y se guarda en la variable "**RMSE_i**". Adicionalmente, se ajusta un modelo de regresión lineal entre los valores observados y predichos, usando la función **lm()**; para ello, se utilizó la fórmula **datos2$IDW.out_est ~ datos2$obs**. El valor del coeficiente de determinación $\mathbf{R^2}$ es igual a 0.66, un desempeño mejor al que tiene la distancia inversa sin considerar la estratificación.

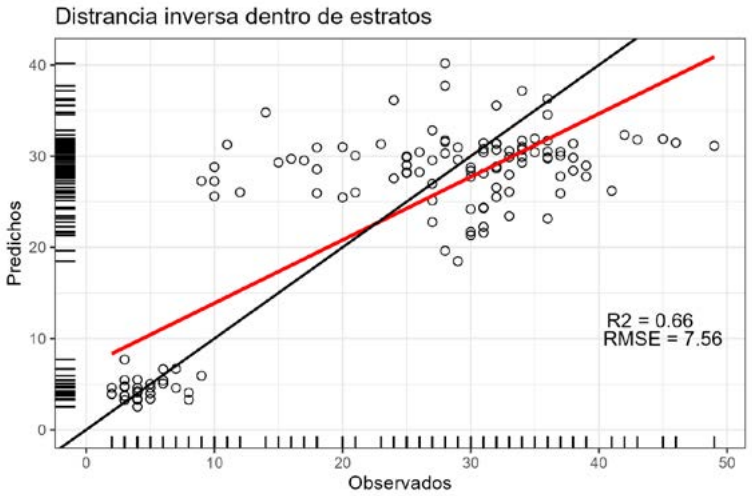
```
#Cálculo del RMSE

idw_est.rmse <- RMSE(datos2$obs, datos2$IDW.out_est)
RMSE_i <- round(idw_est.rmse, digits = 2)
```

```
#Modelo de regresión entre predichos vs. observados

mod1 <- lm(datos2$IDW.out_est ~ datos2$obs)
summary(mod1)

#
# Call:
# lm(formula = datos2$IDW.out_est ~ datos2$obs)
#
# Residuals:
#    Min     1Q Median     3Q    Max
# -9.754 -5.301 -1.361  3.965 18.137
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  6.95398    1.18557   5.866  3.6e-08 ***
# datos2$obs   0.69238    0.04428  15.636  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 6.49 on 128 degrees of freedom
```

```
# Multiple R-squared:  0.6563,  Adjusted R-squared:  0.6537
# F-statistic: 244.5 on 1 and 128 DF,  p-value: < 2.2e-16

r2 = format(summary(mod1)$r.squared, digits = 2)
```

Se obtiene una gráfica de los resultados de la validación con la función **ggplot()** de la librería **ggplot2**. Al principio se coloca la función **ggplot()**, que tiene como parámetros el vector de valores predichos "**IDW.out_est**" y el de valores observados de riqueza de especies "**obs**" del *dataframe* "**datos2**". Las funciones que se utilizaron en esta gráfica fueron: **geom_point()** para el diagrama de dispersión; **labs()** para especificar las etiquetas del título y los ejes; la función **geom_abline()** para colocar la línea 1:1, la línea de regresión **geom_smooth()** y anotaciones con los valores del coeficiente de determinación y el **RMSE annotate()**. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Gráfica de los valores predichos vs. valores observados

p1 <- ggplot(datos2, aes(x=obs, y=IDW.out_est)) +
 geom_point(size=2, shape=21 )+
 geom_rug()+
 geom_smooth(method=lm, se=FALSE, size=1,color="red")+
 labs(title= "Distancia inversa dentro de estratos",
    x="Observados", y = "Predichos")+
  expand_limits(x = 0, y = 0)+
 theme_bw() +
 geom_abline(size=0.7) +
 annotate("text",x = 44, y = 12,
       label = paste("R2 =", r2), size = 4)+
 annotate("text",x = 45, y = 10,
       label = paste("RMSE =", RMSE_i), size = 4)

p1
```




```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_idw_est.tiff", sep="/"),
    width=6, height=4)
```

7.1.2 Kriging dentro de estratos

En la interpolación con Kriging dentro de estratos, se utiliza el Kriging de manera separada para cada uno de los estratos considerados. En este caso se consideraron 2 estratos: el primero que tiene áreas cubiertas con selva mediana subperennifolia, incluyendo las 4 etapas de sucesión, y el segundo con los otros dos tipos de vegetación, la sabana y el bosque bajo inundable.

Se abre la librería "**gstat**", que es en donde se encuentran las funciones para ejecutar la interpolación con Kriging. Hay que recordar que para ejecutar Kriging es necesario que la variable a estudiar tenga una distribución normal. En caso contrario, se requerirá de hacer una trasformación de la variable usando logaritmos, el inverso o la raíz cuadrada. Para verificar la normalidad de la riqueza de especies se obtuvieron dos

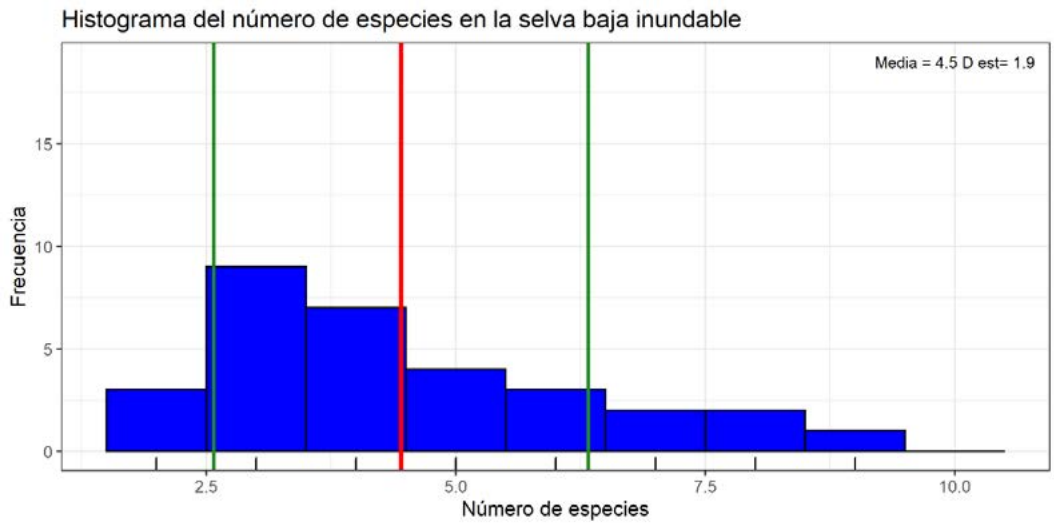
histogramas con la distribución de frecuencias, tanto para las selvas como para los otros dos tipos de vegetación. Para ello se obtuvieron dos subconjuntos del dataframe "**datos**": una para las selvas como "**datos_selva**"; y otra para otros tipos de vegetación como "**datos_bajo**".

Las gráficas se obtuvieron usando la función **ggplot()**. Se colocó la función **ggplot()**, que tiene como eje *x* el parámetro "**Num_esp**" del subconjunto. Las funciones que se utilizaron en esta gráficas fueron: **geom_histogram()**, indicando que la gráfica es de un histograma; **geom_vline()** para imprimir líneas horizontales con el valor de la media y la desviación estándar; **labs()**, que especifica las etiquetas del título y los ejes; anotaciones con los valores de la media y la desviación estándar con **annotate()**; entre otras. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar las gráficas en el disco.

Los histogramas de la riqueza de especies para otros tipos de vegetación y para la selva mediana, muestran una distribución de frecuencias similar a la normal, por lo que se decide no realizar una trasformación de las variables.

```
#Estadísticas

library(gstat)

med_baj <- mean(datos_bajo$Num_esp) #Media
m_n_baj <- round(med_baj, digits = 1)

sd_baj <- sd(datos_bajo$Num_esp)
s_n_baj <- round(sd_baj, digits = 1)

p1 <- ggplot(datos_bajo, aes(x=Num_esp)) +
 geom_histogram(fill="blue", color="black", binwidth = 1)+
 geom_rug()+
 geom_vline(aes(xintercept=mean(Num_esp)), color="red", lwd=1.2)+
 geom_vline(aes(xintercept=mean(Num_esp)+
```

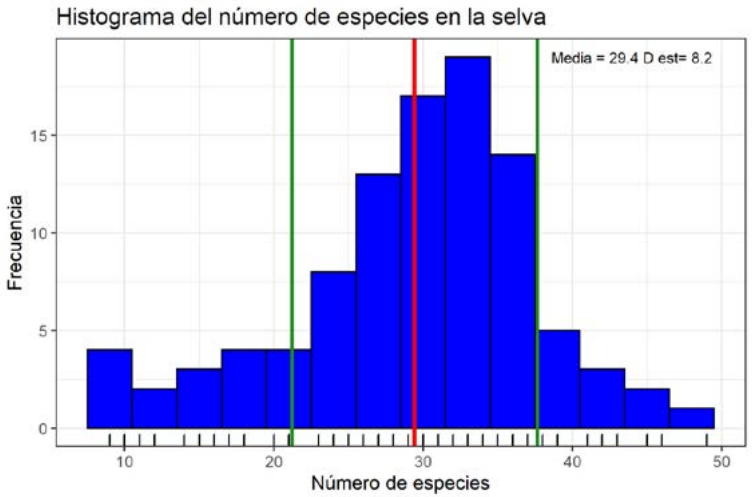
```
            sd(Num_esp)), color="forestgreen", lwd=1)+
 geom_vline(aes(xintercept=mean(Num_esp) - sd(Num_esp)),
          color="forestgreen", lwd=1)+
 annotate("text",x = 10, y = 19,
         label = paste("Media =", m_n_baj,"D est=", s_n_baj),
         size = 3)+
 labs(title=
       "Histograma del número de especies en la selva baja inundable",
     x="Número de especies", y = "Frecuencia")+
 theme_bw()

p1
```




```
#Grabar gráfica

ggsave(p1,
     file=paste(dir.MAPS,"gra_hist5-6.tiff", sep="/"),
     width=6, height=4)
```

```
#Estadísticas

med_sel <- mean(datos_selva$Num_esp) #Media
m_n_sel <- round(med_sel, digits = 1)

sd_sel <- sd(datos_selva$Num_esp)
s_n_sel <- round(sd_sel, digits = 1)
```

```
#Revisar si la variable a interpolar tiene una distribución normal.
#Si no existe normalidad, intentar una transformación log() o sqrt()

p1 <- ggplot(datos_selva, aes(x=Num_esp)) +
 geom_histogram(fill="blue", color="black", binwidth = 3)+
 geom_rug()+
 geom_vline(aes(xintercept=mean(Num_esp)),
        color="red", lwd=1.2)+
 geom_vline(aes(xintercept=mean(Num_esp)+
             sd(Num_esp)), color="forestgreen", lwd=1)+
 geom_vline(aes(xintercept=mean(Num_esp) - sd(Num_esp)),
        color="forestgreen", lwd=1)+
 annotate("text",x = 44, y = 19,
       label = paste("Media =", m_n_sel,"D est=",
                  s_n_sel), size = 3)+
 labs(title="Histograma del número de especies en la selva",
    x="Número de especies", y = "Frecuencia")+
 theme_bw()

p1
```

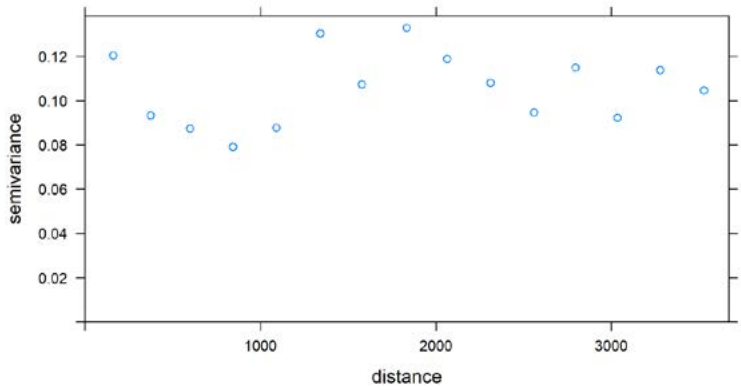



```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_hist1-4.tiff", sep="/"),
    width=6, height=4)
```

Se calculó un variograma experimental utilizando la función **variogram()** con los siguientes argumentos: una fórmula que especifica la variable dependiente "**Num_esp**" y las posibles covariables; en este caso, como no existen covariables, se colocó el número 1, y un objeto de la clase **SpatialPointDataFrame** "**spdf_sel**" con los datos. Esta función regresa un objeto de la clase **variogram** que nombramos como "**ns_sel.vgm**", el cual se imprimió con la función **plot()**.

Se utilizó la función **tiff()** para almacenar este gráfico en un archivo de tipo **TIFF**. En esta función se especifica el directorio y nombre del archivo. El nombre del archivo es "**var_krig_selva1.tiff**" y el del directorio es "**C:\met_int\Maps**". Se especifican, además, el ancho y alto de la imagen y sus unidades, así como la resolución que tendrá la misma en dpi's.

```
#Crear los variogramas experimentales para la selva

ns_sel.vgm = variogram(log(Num_esp) ~ 1, spdf_sel)
plot(ns_sel.vgm)
```



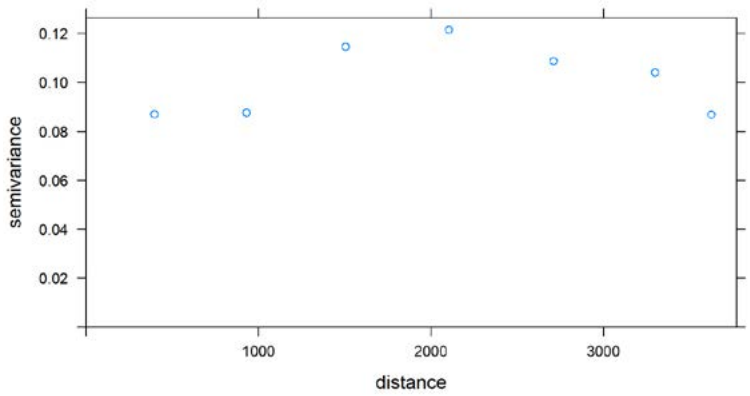

```
#Imprimir la gráfica

tiff(paste(dir.MAPS,"var_krig_selva1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns_sel.vgm)
dev.off()
```

Dado que no se puede ajustar un modelo al variograma, se obtuvo un segundo variograma experimental. En este nuevo variograma se modificó el ancho de los intervalos de distancia en los que se agrupan los pares de puntos para estimaciones de semivarianza. Se usó con un valor de 600. Es decir, se redujo el número de intervalos para tener un mejor ajuste del modelo. Esta función regresa un objeto de la clase **variogram** que nombramos como "**ns_sel.vgm**"., el cual se imprimió con la función **plot()** y se guardó en el disco con la función **tiff()**.

```
ns_sel.vgm = variogram(log(Num_esp) ~ 1, spdf_sel, width=600)
ns_sel.vgm
```

```
#   np     dist     gamma dir.hor dir.ver   id
# 1 147  396.2225 0.08693962      0      0 var1
# 2 282  931.7018 0.08758027      0      0 var1
# 3 399 1506.2290 0.11448685      0      0 var1
# 4 541 2106.4151 0.12145798      0      0 var1
# 5 558 2714.1675 0.10860351      0      0 var1
# 6 634 3301.2633 0.10399808      0      0 var1
# 7  47 3627.2383 0.08668630      0      0 var1
```

```
plot(ns_sel.vgm)
```

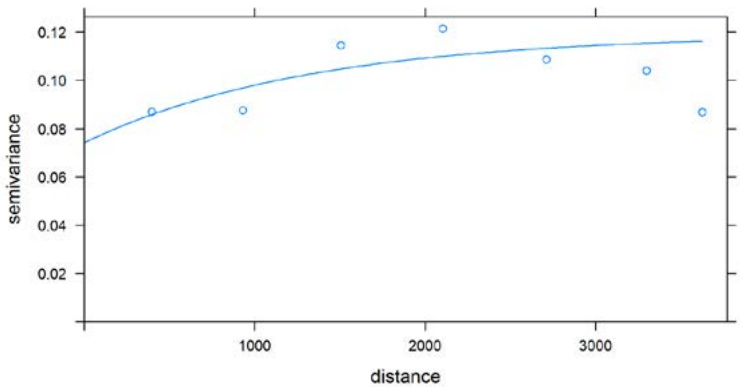



```
#Imprimir la gráfica

tiff(paste(dir.MAPS,"var_krig_selva2.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns_sel.vgm)
dev.off()
```

Se ajustó un modelo al variograma experimental con la función **fit.variogram()** y los siguientes argumentos: un variograma experimental "**ns_sel.vgm**"; el modelo, que se especifica por medio de la función **vgm()**. Esta función tiene como argumentos: un valor de referencia del "**sill**", en este caso fue de 0.04; el tipo de modelo "**Exp**"; un valor

que indique el rango y, por último, el valor de 1 para especificar la existencia de la varianza "**nugget**". Como resultado de la función **fit.variogram()**, se regresa un objeto al cual llamamos "**ns_sel.fit**" y se imprime.

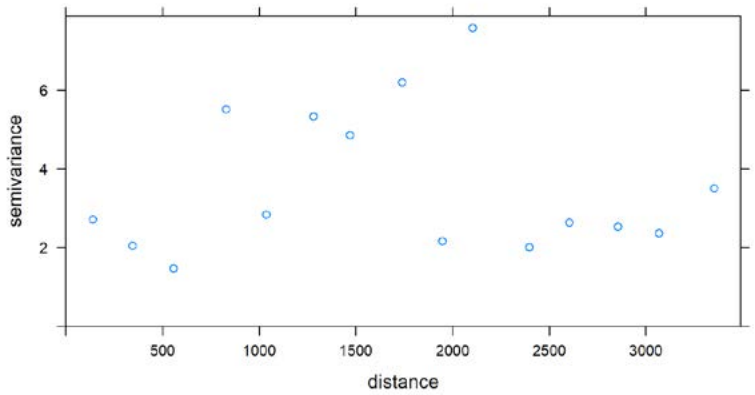
```
#La función vgm() tiene como parámetros sill, modelo, rango y nugget

ns_sel.fit <- fit.variogram(ns_sel.vgm,
                  model = vgm(0.04, "Exp", 2000, 1))
ns_sel.fit

#   model     psill   range
# 1   Nug 0.07420799   0.000
# 2   Exp 0.04487949 1321.36

plot(ns_sel.vgm, ns_sel.fit)
```




```
#Imprimir la gráfica

tiff(paste(dir.MAPS,"var_fitkrig_selva1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns_sel.vgm, ns_sel.fit)
dev.off()
```

Se repite el procedimiento calculando un variograma experimental para otros tipos de vegetación. Se utiliza la función **variogram()** con los mismos argumentos para la selva, pero ahora los datos están en el objeto "**spdf_baj**". También se utilizó la función **tiff()** para almacenar este gráfico en el disco con el nombre "**var_krig_bajo1.tiff**".

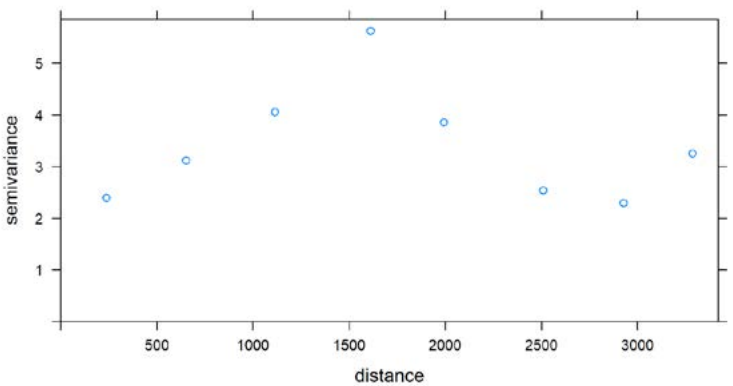
```
#Crear los variogramas experimentales para la selva baja inundable

ns_baj.vgm <- variogram(Num_esp ~ 1, spdf_baj)
plot(ns_baj.vgm)
```




```
#Imprimir la gráfica

tiff(paste(dir.MAPS,"var_krig_bajo1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns_baj.vgm)
dev.off()
```

Dado que no se puede ajustar un modelo al variograma, se obtuvo un segundo variograma experimental. En este nuevo variograma se modificó el ancho de los intervalos de distancia con un valor de 450. Esta función regresa un objeto de la clase **variogram** que nombramos como "**ns_baj.vgm**", el cual se imprimió con la función **plot()** y se guardó en el disco con la función **tiff()**.

```
ns_baj.vgm = variogram(Num_esp ~ 1, spdf_baj, width = 450)
ns_baj.vgm

#   np     dist    gamma dir.hor dir.ver   id
# 1 23  238.4515 2.391304       0       0 var1
# 2 21  653.0104 3.119048       0       0 var1
# 3 27 1114.8299 4.055556       0       0 var1
# 4 31 1611.5653 5.629032       0       0 var1
# 5 21 1993.6041 3.857143       0       0 var1
# 6 29 2509.0464 2.534483       0       0 var1
# 7 36 2928.7586 2.291667       0       0 var1
# 8  8 3287.2742 3.250000       0       0 var1

plot(ns_baj.vgm)
```

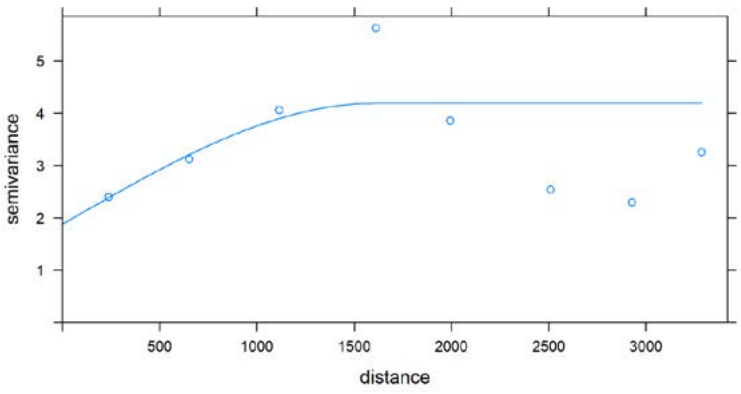



```
#Imprimir la gráfica

tiff(paste(dir.MAPS,"var_krig_bajo2.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns_baj.vgm)
dev.off()
```

```
# png
#   2
```

Se ajustó un modelo al variograma experimental con la función **fit.variogram()** y los siguientes argumentos: un variograma experimental "**ns_baj.vgm**"; el modelo, que se especifica por medio de la función **vgm()**. Esta función tiene como argumentos: un valor de referencia del "**sill**", en este caso fue de 1; el tipo de modelo "**Sph**"; un valor que indique el rango y, por último, el valor de 1 para especificar la existencia de la varianza "**nugget**". Como resultado de la función **fit.variogram()**, se regresa un objeto al cual llamamos "**ns_baj.fit**" y se imprime.

```
#Ajustar semivariograma en selva baja inundable

ns_baj.fit = fit.variogram(ns_baj.vgm,
          model = vgm(1, "Sph", 2000, 1))
ns_baj.fit

#   model    psill    range
# 1   Nug 1.875275    0.000
# 2   Sph 2.314454 1606.631

plot(ns_baj.vgm, ns_baj.fit)
```



```
#Imprimir la gráfica

tiff(paste(dir.MAPS,"var_fitkrig_bajo1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(ns_baj.vgm, ns_baj.fit)
dev.off()

# png
#  2
```

La interpolación con Kriging se obtiene utilizando la función **krige(formula, data, grid, model, ...)**, aquí se utilizaron solamente 4 argumentos: la fórmula que define la variable dependiente, y el valor de 1 cuando se ejecuta un Kriging ordinario; el objeto que contiene los datos "**spdf_sel**" y "**spdf_baj**", respectivamente para cada estrato; un *grid* donde se almacenan los datos y el modelo de semivariograma ajustado "**ns_sel.fit**" y "**ns_baj.fit**". Esta función regresa un *grid* con el mapa interpolado, el cual se nombró como "**ns_sel.k**" y "**ns_baj.k**", respectivamente para cada estrato. Estos *grid* se convierten a formato *raster* usando la función **raster()** y después se eliminan los valores donde no hay vegetación utilizando la función **mask()**, con sus respectivas máscaras "**rs**" y "**rb**".

```
#interpolación con Kriging para cada estrato

ns_sel.k <- gstat::krige(Num_esp~1,
          spdf_sel, grd,
          model = ns_sel.fit)

# [using ordinary kriging]

ns_baj.k <- gstat::krige(Num_esp~1,
          spdf_baj, grd,
          model = ns_baj.fit)

# [using ordinary kriging]
```

```
rgrd_baj.k <- raster(ns_baj.k)
grd.kg_baj <- mask(rgrd_baj.k, rb, maskvalue= 0, updatevalue = 0)

rgrd_sel.k <- raster(ns_sel.k)
grd.kg_sel <- mask(rgrd_sel.k, rs, maskvalue= 0, updatevalue = 0)
```

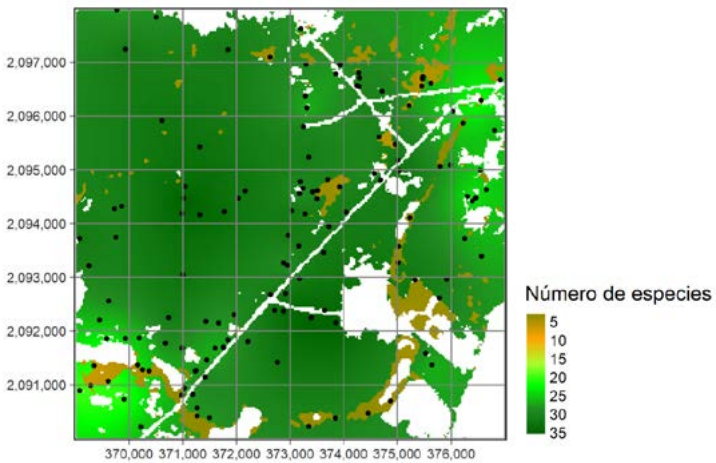
Enseguida se suman los objetos *raster* generados para obtener un solo objeto de toda el área de estudio, que incluya los dos estratos considerados. A este nuevo objeto *raster* "**grd.kgsb**", se le aplicó la máscara que corresponde a todos los tipos de vegetación. Esto, con el objeto de eliminar los valores donde no hay vegetación, o que no pudieron ser eliminados antes; para esto se usa la función **mask()**.

```
#Función para remplazar valores nulos con 0 en raster

grd.kgsb <- grd.kg_baj + grd.kg_sel
grd.kg_todos <- mask(grd.kgsb, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**grd.kg_todos**" para obtener el mapa de riqueza de especies con la técnica de Kriging dentro de estratos, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**Int_kri_est.tiff**", usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
        "green","forestgreen","darkgreen")

tm1 <- tm_shape(grd.kg_todos) +
 tm_raster(style= "cont", n=10,
           palette =my_col,
           title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
```

```
 tm_grid( ) +
 tm_layout(legend.outside = TRUE,
          legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
          filename= paste(dir.MAPS,"mapa_kg_est.tiff", sep="/"),
          width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(grd.kg_todos,
          paste(folder, "Int_kri_est.tiff", sep= "/"),
          overwrite= TRUE)
```

Para llevar a cabo la validación cruzada, se crean los vectores "**KG_out.sel**" y "**KG_out.baj**", que tienen el mismo número de elementos que los objetos "**spdf_sel**" y "**spdf_baj**" de la clase **SpatialPoindDataFrame**. Después se realiza la interpolación usando

la función **krige()**, excluyendo un dato a la vez. Esto permite obtener una lista de valores estimados de la riqueza de especies que se almacenarán en los vectores. Estos serán comparados con aquellos valores de riqueza de especies medidos en los sitios de muestreo. Posteriormente, estos vectores se concatenan usando la función **c()** y se guardan en un vector que llamamos "**KG.out_est**". Para poder comparar este vector con los datos estimados, habrá que colocar las observaciones en el mismo orden. Para ello se crean el vector "**a1**" y el vector "**b1**" con los datos de la riqueza de especies para las selvas y para los otros dos tipos de vegetación, respectivamente. Los vectores "**a1**" y "**b1**" se concatenan en un solo vector, para después guardar los valores observados y estimados en un *dataframe* "**datos2**".

```
#Validación cruzada para las interpolaciones con Kriging dentro
#de estratos

KG_out.sel <- vector(length = length(spdf_sel))

for (i in 1:length(spdf_sel)) {
 KG_out.sel[i] <- krige(Num_esp ~ 1, spdf_sel[-i,],
          spdf_sel[i,],
          model = ns_sel.fit)$var1.pred
 }

# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
...
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
```

```
KG_out.baj <- vector(length = length(spdf_baj))

for (i in 1:length(spdf_baj)) {
 KG_out.baj[i] <- krige(Num_esp ~ 1, spdf_baj[-i,],
                spdf_baj[i,],
                model = ns_baj.fit)$var1.pred
 }

# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
...
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]

KG.out_est <- c(KG_out.sel, KG_out.baj)

a1 <- datos_selva$Num_esp
b1 <- datos_bajo$Num_esp

KG.obs <- c(a1, b1)
datos2<- data.frame(KG.obs, KG.out_est)
```

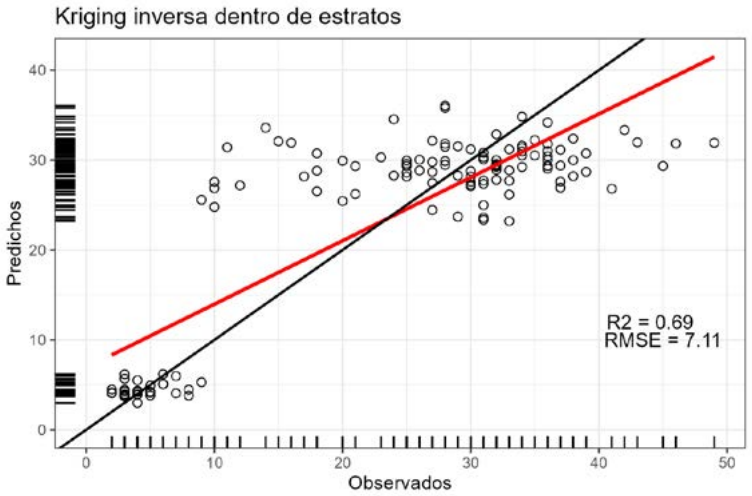
Se calcula el valor del **RMSE** usando el vector de los datos estimados "**datos2$KG.out_est**" y el de los observados "**datos2$KG.obs**". Por último, se redondea este valor a 2 dígitos y se guarda en la variable "**RMSE_i**". Adicionalmente, se ajusta un modelo de regresión lineal entre los valores observados y predichos usando la función **lm()**, para ello se utilizó la fórmula **datos2$KG.out_est ~ datos2$KG.obs**. El valor del coeficiente de determinación $\mathbf{R^2}$ es igual a 0.69, un desempeño mejor al que tiene la distancia inversa dentro de estratos.

```
#Cálculo del RMSE

kg_est.rmse <- RMSE(datos2$KG.obs, datos2$KG.out_est)
RMSE_i <- round(kg_est.rmse, digits = 2)
```

```
#Modelo de regresión entre predichos vs. observados

mod2 <- lm(datos2$KG.out_est ~ datos2$KG.obs)
summary(mod2)

#
# Call:
# lm(formula = datos2$KG.out_est ~ datos2$KG.obs)
#
# Residuals:
#    Min     1Q Median     3Q    Max
# -9.550 -5.089 -1.067  3.778 16.804
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept)   6.91530    1.10725   6.245 5.75e-09 ***
# datos2$KG.obs 0.70533    0.04136  17.054  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 6.062 on 128 degrees of freedom
# Multiple R-squared:  0.6944,  Adjusted R-squared:  0.692
# F-statistic: 290.9 on 1 and 128 DF,  p-value: < 2.2e-16

r2 <- format(summary(mod2)$r.squared, digits = 2)
```

Se obtiene una gráfica de los resultados de la validación con la función **ggplot()** de la librería **ggplot2**. Al principio se coloca la función **ggplot()**, que tiene como parámetros el vector de valores predichos "**KG.out_est**" y el de valores observados de riqueza de especies "**KG.obs**" del *dataframe* "**datos2**". Las funciones que se utilizaron en esta gráfica fueron: **geom_point()** para el diagrama de dispersión; **labs()** para especificar las etiquetas del título y los ejes; la función **geom_abline()** para colocar la línea 1:1, la línea de regresión **geom_smooth()** y anotaciones con los valores del coeficiente de determinación y el **RMSE** usando la función **annotate()**. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Gráfica de los valores predichos vs. valores observados

p1 <- ggplot(datos2, aes(x=KG.obs, y=KG.out_est)) +
 geom_point(size=2, shape=21 )+
 geom_rug()+
 geom_smooth(method=lm, se=FALSE, size=1, color="red")+
 labs(title= "Kriging inversa dentro de estratos",
    x="Observados", y = "Predichos")+
 expand_limits(x = 0, y = 0)+
 theme_bw() +
 geom_abline(size=0.7) +
 annotate("text",x = 44, y = 12,
       label = paste("R2 =", r2), size = 4)+
 annotate("text",x = 45, y = 10,
       label = paste("RMSE =", RMSE_i), size = 4)

p1
```




```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_kg_est.tiff", sep="/"),
    width=6, height=4)
```

7.2 Regresión con Kriging

Este enfoque combina un modelo de regresión múltiple con Kriging ordinario de los residuales de regresión. Es decir, este método aborda tanto la dependencia espacial de las observaciones como la relación entre la variable de interés y el conjunto de las variables auxiliares, que funcionan como variables explicativas en el modelo de regresión (Webster et al., 2001). El estimador Kriging de regresión (**RK**) del atributo de interés $\mathbf{Z_{rk}(x)}$, se define como la suma de la estimación de regresión $\mathbf{Z_r(x)}$ obtenida como una función lineal entre la riqueza de especies (variable dependiente) y las características espectrales y de textura de la imagen Landsat 7 (variables explicativas), y la estimación de valores residuales espacialmente correlacionados $\boldsymbol{\epsilon_{ok}(x)}$ usando Kriging. Estas relaciones se representan en la siguiente ecuación:

$$Z_{rk}(x) = Z_r(x) + \varepsilon_{ok}(x) \quad \text{.....(7)}$$

Para llevar a cabo este procedimiento de interpolación, se abre la librería "**gstat**", que es en donde se encuentran las funciones de **variogram()** y **krige()**.

Para obtener estimaciones de regresión con Kriging, se usó un modelo de regresión lineal para predecir la riqueza de especies de árboles. El modelo de regresión lineal múltiple usó valores de reflectancia de las bandas espectrales TM3 y TM4 de la imagen Landsat 7, de los valores del NDVI y 7 medidas de textura para cada una de las 3 capas anteriores. En total hay 27 variables explicativas. Estas, junto con la variable dependiente, se asignan a la fórmula "**f.r**". Debido al número elevado de variables explicativas, se usó un método de selección de variables; en este caso fue el algoritmo de "**stepwise**" con selección hacia atrás "**backward**". Las funciones para realizar este procedimiento se encuentran en la librería **MASS**.

```
#Definir la ecuación del modelo de regresión

library(gstat)

f.r <- as.formula(Num_esp ~ TM3+TM4+NDVI+TM3_med+TM3_var+
          TM3_hom+TM3_con+TM3_dis+TM3_entr+TM3_smo+
          TM3_cor+TM4_med+TM4_var+TM4_hom+TM4_con+
          TM4_dis+TM4_entr+TM4_smo+TM4_cor+NDVI_med+
          NDVI_var+NDVI_hom+NDVI_con+NDVI_dis+
          NDVI_entr+NDVI_smo+NDVI_cor)

library(MASS) # Para poder usar la función stepAIC
```

La ejecución del modelo de regresión lineal múltiple con selección de variables por el método **backward**, requiere la ejecución de un modelo de regresión lineal que incluya todas las variables explicativas. Para ello, se utilizó la función **lm()**, que tiene como argumentos la fórmula "**f.r**" y el *dataframe* donde se encuentran los registros

con la información de todas las variables "**datos**". Esta función arroja como resultado un objeto de la clase **lm**, que fue guardado en "**full.model**". Se utilizó la función **stepAIC(object, scope, scale = 0, direction = c("both", "backward", "forward"), ...)** para ejecutar el modelo de regresión con selección de variables. Aquí se utilizaron solamente 2 argumentos, el primero es un objeto que representa un modelo de regresión lineal y se utiliza como modelo inicial en el algoritmo "**stepwise selection**". En este caso, es el resultado del modelo completo "**full.model**". Además, se especifica el método particular que se utilizará en el algoritmo de "**stepwise selection**", que puede ser "**both**", "**backward**" o "**forward**". Los resultados se guardaron en un objeto que llamamos "**modback**". Como resultado se obtienen diferentes modelos en donde se fueron eliminando algunas variables. En cada modelo se especifica el valor del **AIC**, así como las variables que fueron seleccionadas. En este caso, el mejor modelo tuvo un valor de AIC=540.17, con 10 variables explicativas.

```
full.model <- lm(f.r, data=datos)

modback <-stepAIC(full.model,direction = "backward")

# Start:  AIC=566.72
# Num_esp ~ TM3 + TM4 + NDVI + TM3_med + TM3_var + TM3_hom + TM3_con +
#    TM3_dis + TM3_entr + TM3_smo + TM3_cor + TM4_med + TM4_var +
#    TM4_hom + TM4_con + TM4_dis + TM4_entr + TM4_smo + TM4_cor +
#    NDVI_med + NDVI_var + NDVI_hom + NDVI_con + NDVI_dis + NDVI_entr +
#    NDVI_smo + NDVI_cor
#
#           Df Sum of Sq    RSS    AIC
# - NDVI_dis  1     0.17 6608.9 564.72
# - TM4_med   1     0.19 6609.0 564.72
# - TM3_var   1     0.66 6609.4 564.73
# - TM3_smo   1     0.91 6609.7 564.74
# - NDVI_entr 1     1.22 6610.0 564.74
# - NDVI_smo  1     1.55 6610.3 564.75
# - NDVI_var  1     1.61 6610.4 564.75
# - NDVI_cor  1     3.28 6612.0 564.78
```

```
# - TM3_med   1     4.78 6613.5 564.81
# - NDVI_med  1     6.58 6615.3 564.85
# - NDVI_con  1     9.62 6618.4 564.91
# - TM4_con   1    13.55 6622.3 564.99
#  ...

#
# Step:  AIC=540.17
# Num_esp ~ TM3 + TM4 + NDVI + TM3_med + TM3_con + TM3_dis + TM3_entr +
#    TM3_cor + TM4_var + TM4_hom
#
#         Df Sum of Sq    RSS    AIC
# <none>               6998.5 540.17
# - TM3_cor   1   123.44 7121.9 540.44
# - TM3_med   1   253.96 7252.5 542.80
# - TM3_entr  1   509.74 7508.2 547.31
# - TM4_hom   1   516.82 7515.3 547.43
# - TM4_var   1   564.33 7562.8 548.25
# - TM3_con   1   727.97 7726.5 551.03
# - TM3       1   832.03 7830.5 552.77
# - NDVI      1   921.90 7920.4 554.26
# - TM3_dis   1  1079.72 8078.2 556.82
# - TM4       1  1397.83 8396.3 561.84
```

Utilizando la función **summary()** se pueden ver las variables y parámetros del modelo de regresión final. El valor del coeficiente de determinación ($\mathbf{R^2}$) fue de 0.67, sin embargo, se puede observar que la variable **TM3-cor** no fue significativa; esto quiere decir que el coeficiente asociado a esta variable no difiere de cero.

```
#Definir la ecuación del modelo de regresión

summary(modback)

#
```

```
# Call:
# lm(formula = Num_esp ~ TM3 + TM4 + NDVI + TM3_med + TM3_con +
#     TM3_dis + TM3_entr + TM3_cor + TM4_var + TM4_hom, data = datos)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -17.3946  -5.1182   0.3718   4.9962  19.3587
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  76.69461   31.55596   2.430 0.016573 *
# TM3          -3.31477    0.88128  -3.761 0.000264 ***
# TM4           2.41346    0.49504   4.875 3.39e-06 ***
# NDVI       -356.55852   90.05700  -3.959 0.000128 ***
# TM3_med      -2.31929    1.11609  -2.078 0.039855 *
# TM3_con      -1.04850    0.29802  -3.518 0.000616 ***
# TM3_dis      18.64365    4.35114   4.285 3.73e-05 ***
# TM3_entr    -10.41495    3.53762  -2.944 0.003896 **
# TM3_cor      -0.42453    0.29303  -1.449 0.150037
# TM4_var      -0.22512    0.07267  -3.098 0.002434 **
# TM4_hom      25.82740    8.71244   2.964 0.003664 **
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 7.669 on 119 degrees of freedom
# Multiple R-squared:  0.6742,  Adjusted R-squared:  0.6468
# F-statistic: 24.63 on 10 and 119 DF,  p-value: < 2.2e-16
```

Se ejecutan dos modelos de regresión lineal utilizando una fórmula con las variables seleccionadas en el modelo de "**backward**", pero eliminando las variables que no fueron significativas, quedando un modelo final "**best.model**" con un valor de $\mathbf{R^2}$ igual a 0.65 y con 8 variables. Finalmente, los residuales en este modelo se guardan en la variable "**res**" dentro del objeto "**spdf"** del tipo **SpatialPointDataframe**, por medio de la función **residuals()**, que tiene como argumento el objeto "**best.model**".

```
f.rb <- as.formula(Num_esp ~ TM3+TM4+NDVI+TM3_med+ TM3_con+
          TM3_dis+TM3_cor+TM4_var+TM4_hom)

back.model <- lm(f.rb,data=datos)
summary(back.model)

#
# Call:
# lm(formula = f.rb, data = datos)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -18.1267  -5.0617   0.4139   4.8065  18.1196
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  47.96510   30.95306   1.550 0.123869
# TM3          -3.07217    0.90502  -3.395 0.000932 ***
# TM4           2.54574    0.50850   5.006 1.93e-06 ***
# NDVI       -359.82700   92.88252  -3.874 0.000175 ***
# TM3_med      -3.17760    1.11123  -2.860 0.005005 **
# TM3_con      -0.64284    0.27256  -2.359 0.019964 *
# TM3_dis      11.05853    3.61664   3.058 0.002751 **
# TM3_cor       0.14153    0.22809   0.620 0.536113
# TM4_var      -0.24058    0.07476  -3.218 0.001661 **
# TM4_hom       21.44198    8.85416   2.422 0.016945 *
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 7.91 on 120 degrees of freedom
# Multiple R-squared:  0.6505,  Adjusted R-squared:  0.6243
# F-statistic: 24.82 on 9 and 120 DF,  p-value: < 2.2e-16
```

```
#Extraer los residuales

f.best <- as.formula(Num_esp ~ TM3+TM4+NDVI+TM3_med+
          TM3_con+TM3_dis+TM4_var+TM4_hom)

best.model <- lm(f.best,data=datos)
summary(best.model)

#
# Call:
# lm(formula = f.best, data = datos)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -17.9470  -5.2097   0.4063   4.7113  18.9459
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  52.64670   29.94302   1.758  0.08124 .
# TM3          -3.23449    0.86418  -3.743  0.00028 ***
# TM4           2.62829    0.48954   5.369 3.89e-07 ***
# NDVI       -374.44328   89.61722  -4.178 5.58e-05 ***
# TM3_med      -3.19251    1.10814  -2.881  0.00469 **
# TM3_con      -0.56885    0.24447  -2.327  0.02163 *
# TM3_dis      10.17215    3.31411   3.069  0.00265 **
# TM4_var      -0.23914    0.07454  -3.208  0.00171 **
# TM4_hom      22.54901    8.65047   2.607  0.01029 *
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 7.89 on 121 degrees of freedom
# Multiple R-squared:  0.6494,  Adjusted R-squared:  0.6262
# F-statistic: 28.01 on 8 and 121 DF,  p-value: < 2.2e-16

spdf$res <- residuals(best.model)
```

Se selecciona el directorio en donde se encuentra la lista de archivos con las imágenes de las bandas TM3, TM4, el NDVI y las 5 imágenes de texturas, que en este caso es "**C:/met_int/img/**". Se guardó el directorio en la variable "**dir.tif**" para usarlo posteriormente. Enseguida, se utiliza la función **list.files()** que tiene la siguiente forma: **list.files(path = ".", pattern = NULL, all.files = FALSE, full.names = FALSE, recursive = FALSE, ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)**. Para este caso particular, se utilizaron solamente 3 argumentos: el primero "**path**", que tiene el directorio completo donde se encuentran los archivos; aquí se utilizó el que se guardó en "**dir.tif**"; el segundo argumento es "**pattern**", el cual se refiere a un grupo de caracteres que contiene los archivos que se van a leer; aquí se utilizó la extensión de los archivos que es "**.tif**"; y, por último, el argumento "**full.names**", se colocó el valor de "**TRUE**" para indicar que se consideran los nombres de todos los archivos. Esta función produce un vector de caracteres con los nombres de archivos leídos y los directorios en donde estos se encuentran. Este vector se guardó en "**l.tif**". Posteriormente, se imprime este vector, que arroja la lista de archivos leídos y su ubicación.

```
#Creación del mapa usando el modelo de regresión
#Cargar directorio de imágenes tif

dir.tif <- "C:/met_int/img/"
```

```
#Lista de archivos raster

l.tif <- list.files(path = dir.tif,
            pattern = ".tif$$",
            full.names = TRUE)
l.tif

# [1] "D:/met_int/img/NDVI.tif"    "D:/met_int/img/TM3.tif"
# [3] "D:/met_int/img/TM3_con.tif" "D:/met_int/img/TM3_dis.tif"
# [5] "D:/met_int/img/TM3_med.tif" "D:/met_int/img/TM4.tif"
# [7] "D:/met_int/img/TM4_hom.tif" "D:/met_int/img/TM4_var.tif"
```

Un objeto **StackRaster** es una colección de objetos o archivos *raster*, los cuales tienen la misma extensión espacial y resolución. Se creó un objeto **Stackraster** utilizando la función **stack()** de la librería *raster*, con la lista de archivos **tif** leídos antes. Como resultado de ejecutar esta función, se obtuvo un objeto del tipo **stackraster** que se nombró como "**r.tif**" y se imprimió.

```
#Stack de los archivos raster

r.tif <- stack (l.tif)
r.tif

# class       : RasterStack
# dimensions  : 268, 268, 71824, 8 (nrow, ncol, ncell, nlayers)
# resolution  : 30, 30 (x, y)
# extent      : 368985, 377025, 2089965, 2098005 (xmin, xmax, ymin, ymax)
# crs         : +proj=utm +zone=16 +datum=WGS84 +units=m +no_defs
# names       : NDVI, TM3, TM3_con, TM3_dis, TM3_med, TM4, TM4_hom, TM4_var
```

El mapa con la distribución espacial de la riqueza de especies utilizando el método de regresión, se obtiene aplicando la función que se ajustó [52.65-(3.23*TM3) + (2.63*TM4) - (374.44*NDVI) - (3.19*TM3_med) - (0.57*TM3_con) + (10.17*TM3_dis) - (0.24*TM4_var) + (22.55*TM4_hom)] al *stack* de archivos *raster*, que contienen los *raster* de estas variables por medio del álgebra de mapas con la siguiente expresión:

```
#Cálculo del mapa

map.reg <- 52.64670+(-3.23449*r.tif$TM3)+(2.62829*r.tif$TM4)+
 (-374.44328*r.tif$NDVI)+(-3.19251*r.tif$TM3_med)+
 (-0.56885*r.tif$TM3_con)+(10.17215*r.tif$TM3_dis)+
 (-0.23914*r.tif$TM4_var)+(22.54901*r.tif$TM4_hom)
```

Se eliminan los valores donde no hay vegetación utilizando la función **mask()** y su máscara "**rc**", que indica zonas de no bosque. Enseguida se define la función **fun()**,

la cual es utilizada para remplazar valores menores a 0 por NA, en un objeto de tipo *raster*. Esta función fue aplicada al objeto *raster* "**map.reg**". Después se eliminan los valores donde no hay vegetación o que no pudieron ser eliminados antes, para esto se usa la función **mask()**.

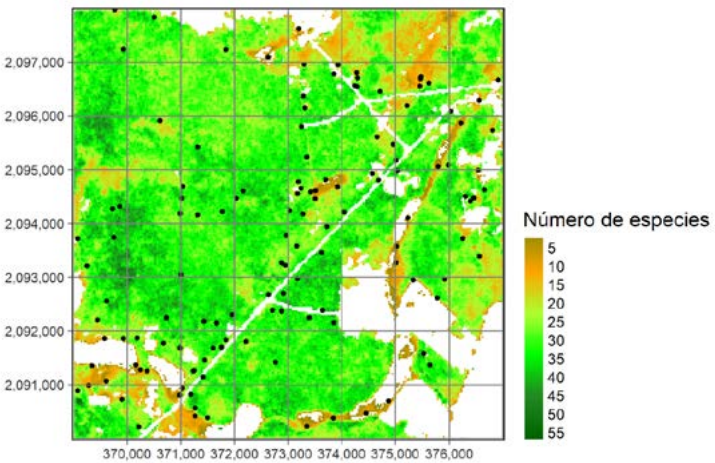
```
#Eliminar areas de no vegetación

map.regf <- mask(map.reg, rc, maskvalue= 0)
```

```
#Establecer valores menores a cero como NA

fun <- function(x){
 x[ x<0 ] <- NA; return(x)
}

map.regf2 <- calc(map.regf, fun)
```

Se imprime el objeto *raster* "**map.regf2**" para obtener el mapa de riqueza de especies con la técnica de regresión, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**regresion.tiff**", usando la función **writeRaster()**.

```
#Gráfica del mapa de regresión y los sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
            "green","forestgreen","darkgreen")

tm1 <- tm_shape(map.regf2) +
 tm_raster(style= "cont", n=10,
           palette =my_col,
           title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid() +
```

```
tm_layout(legend.outside = TRUE,
          legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
          filename= paste(dir.MAPS,"mapa_regresion.tiff", sep="/"),
          width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(map.regf2,
          paste(folder, "regresion.tiff", sep= "/"),
          overwrite= TRUE)
```

Para llevar a cabo la validación cruzada de la estimación usando regresión, primero se crea un vector "**reg.out**". Después se obtiene un modelo de regresión usando la función **lm()**, este modelo se crea de manera repetida excluyendo un dato a la vez. Esto permite obtener una lista de valores estimados de la riqueza de especies que se almacena

en el vector "**reg.out**", que serán comparados con aquellos medidos en los sitios de muestreo. Posteriormente, este vector se guarda en el *dataframe* donde se encuentran los datos como "**datos$predreg**". Enseguida, se calcula el valor del **RMSE** usando el vector de los datos estimados "**datos$predreg**" y el de los observados "**datos$Num-esp**". Por último, se redondea este valor a 2 dígitos y se guarda en la variable "**RMSE_i**". Adicionalmente se ajusta un modelo de regresión lineal entre los valores observados y predichos usando la función **lm()**, para ello se utilizó la fórmula **datos$predreg ~ datos$Num_esp**. El valor del coeficiente de determinación $\mathbf{R^2}$ es igual a 0.60.

```
#Validación cruzada para regresión

n <- 0
reg.out <- as.vector(0)

for (i in 1:nrow(datos)){
 n <- 1 + n
 model <- lm(Num_esp ~ TM3+TM4+NDVI+TM3_med+TM3_con+
          TM3_dis+TM4_var+TM4_hom,
         data=datos[-n,])

 reg.out[i] <- model$coefficients[1]+
  (model$coefficients[2]*datos$TM3[i])+
  (model$coefficients[3]*datos$TM4[i])+
  (model$coefficients[4]*datos$NDVI[i])+
  (model$coefficients[5]*datos$TM3_med[i])+
  (model$coefficients[6]*datos$TM3_con[i])+
  (model$coefficients[7]*datos$TM3_dis[i])+
  (model$coefficients[8]*datos$TM4_var[i])+
  (model$coefficients[9]*datos$TM4_hom[i])
}
datos$predreg <- reg.out
```

```
#Cálculo del RMSE
```

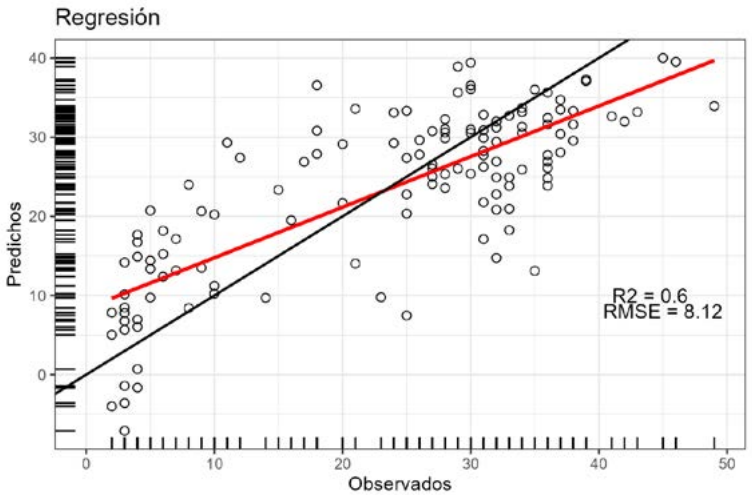
```
reg.rmse <- RMSE(datos$Num_esp, datos$predreg)
RMSE_i <- round(reg.rmse, digits = 2)
```

```
#Modelo de regresión entre predichos vs. observados para
#regresión múltiple

mod = lm(datos$predreg ~ datos$Num_esp)
summary(mod)

#
# Call:
# lm(formula = datos$predreg ~ datos$Num_esp)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -17.6243  -3.9609   0.4173   4.2024  16.6740
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)   8.37556    1.22738   6.824 3.18e-10 ***
# datos$Num_esp  0.63901    0.04584  13.939  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 6.719 on 128 degrees of freedom
# Multiple R-squared:  0.6028,  Adjusted R-squared:  0.5997
# F-statistic: 194.3 on 1 and 128 DF,  p-value: < 2.2e-16

r2 = format(summary(mod)$r.squared, digits = 2)
```

Para obtener de manera gráfica estos resultados, se utiliza la función **ggplot()** de la librería **ggplot2**. Al principio se coloca la función **ggplot()**, que tiene como parámetros el vector de valores predichos "**predreg**" y el de valores observados "**Num_esp**" del *dataframe* "**datos**". Se utilizaron funciones similares a como se hizo la validación cru-

zada anteriormente. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Gráfica de los valores predichos vs. valores observados

p1 <- ggplot(datos, aes(x=Num_esp, y=predreg)) +
 geom_point(size=2, shape=21 )+
 geom_rug( )+
 geom_smooth(method=lm, se=FALSE, size=1, color="red")+
 labs(title= "Regresión",x="Observados", y = "Predichos")+
 expand_limits(x = 0, y = 0)+
 theme_bw( ) +
 geom_abline(size=0.7) +
 annotate("text",x = 44, y = 10,
          label = paste("R2 =", r2), size = 4)+
 annotate("text",x = 45, y = 8,
          label = paste("RMSE =", RMSE_i), size = 4)

p1
```

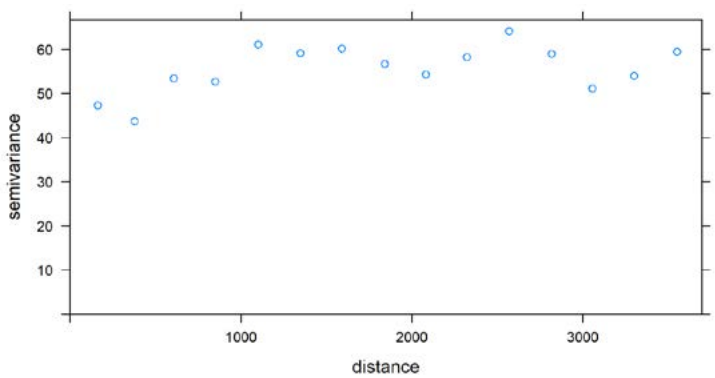


```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_regresion.tiff", sep="/"),
    width=6, height=4)
```

Para continuar con el procedimiento de interpolación de regresión con Kriging, se calcula el variograma experimental de los residuales utilizando la función **variogram()**. Se utilizaron 2 argumentos: un objeto de tipo "***formula***", que especifica la variable dependiente y las posibles covariables, en este caso como no existen covariables se colocó el numero 1; y un objeto de la clase **SpatialPointDataFrame** "**spdf**" con los datos que son los residuales "**res**". Esta función regresa un objeto de la clase **variogram** que nombramos como "**res.vgm**", el cual se imprimió con la función **plot()**.

Se utilizó la función **tiff()** para almacenar este gráfico en un archivo de tipo **TIFF**. En esta función se especifica el directorio y nombre del archivo, como se ha hecho anteriormente; el nombre del archivo es "**var_krig_reg1.tiff**", y el del directorio es "**C.\met_int\Maps**". Se especifican, además, el ancho y alto de la imagen y sus unidades, así como la resolución que tendrá la misma en dpi's.

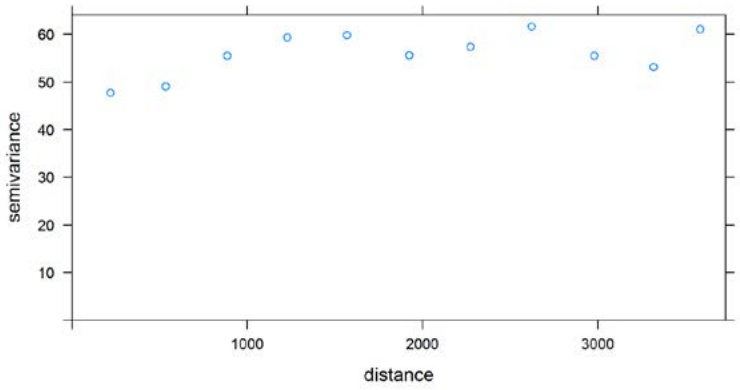
```
#Crear el variograma experimental

res.vgm = variogram(res ~ 1, spdf)
plot(res.vgm)
```



```
#Guardar la gráfica

tiff(paste(dir.MAPS,"var_krig_reg1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(res.vgm)
dev.off()
```

En un segundo variograma experimental, se usó la función **variogram()** con 3 argumentos: la fórmula donde se especifica la variable dependiente; el objeto "**spdf**" con los datos "**res**" y el ancho de los intervalos de distancia en los que se agrupan los pares de puntos de datos para estimaciones de semivarianza, que en este caso fue de 350. Es decir, se redujo el número de intervalos para tener un mejor ajuste del modelo. Esta función regresa un objeto de la clase **variogram** que nombramos como "**res.vgm**", el cual se imprimió con la función **plot()** y se guardó en el disco con la función **tiff()**.

```
#Crear el variograma experimental ancho 350

res.vgm = variogram(res~1, spdf, width = 350)
plot(res.vgm)
```




```
#Guardar la gráfica

tiff(paste(dir.MAPS,"var_krig_reg2.tiff", sep="/"),
```

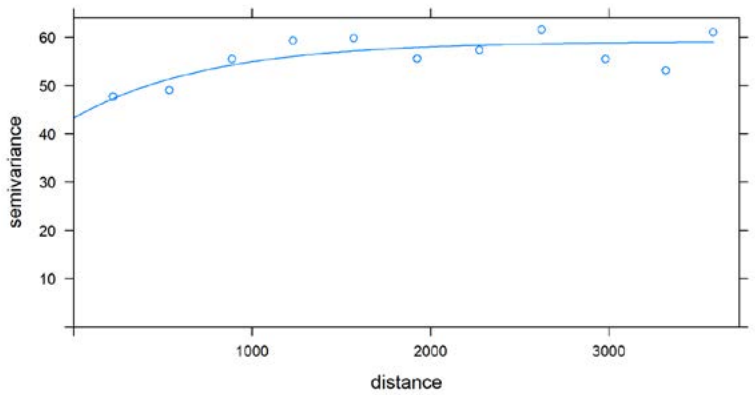
```
   width = 7, height = 4, units = 'in', res = 300)
plot(res.vgm)
dev.off()
```

Enseguida, se ajustó un modelo al variograma experimental usando la función **fit.variogram()**, con 2 argumentos: un objeto de tipo **variogram** "**res.vgm**", que fue la salida al ejecutar la función **variogram()**; y el modelo, que se especifica por medio de la función **vgm()**. Esta función genera un modelo de semivariograma y tiene diferentes argumentos: un valor de referencia del "**sill**", en este caso fue de 40; el tipo de modelo "**Exp**"; un valor que indique el rango y el componente "**nugget**". Como resultado, la función **fit.variogram()** regresa un objeto al cual llamamos "**res.fit**" y se imprime.

Se ajustó un modelo exponencial al semivariograma experimental. La varianza estructural, que determina la dependencia espacial explicada por el modelo y calculada como (varianza total – varianza nugget/varianza total) *100, fue (15.86/59.15) *100 y tuvo un valor de 26.8 %. Esto significa que una parte de la variabilidad de la distribución espacial de la riqueza de especies no es explicada por el modelo (73.2 %).

El modelo ajustado se imprime con la función **plot()** y se guardó en el disco con la función **tiff()**.

```
#Ajustar una función al variograma experimental

res.fit = fit.variogram(res.vgm,
                model = vgm(30, "Exp", 1500, 1))
res.fit

#  model   psill   range
# 1  Nug 43.28775   0.0000
# 2   Exp 15.86054 753.5481

plot(res.vgm, res.fit)
```



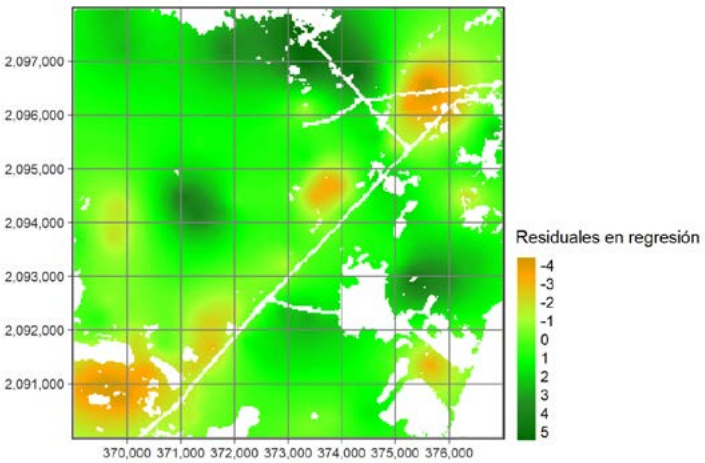

```
#Guardar la gráfica

tiff(paste(dir.MAPS,"var_fitkrig_reg1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(res.vgm, res.fit)
dev.off()
```

El mapa de los residuales usando interpolación con Kriging, se obtiene utilizando la función **krige()** con 4 argumentos: la fórmula que define la variable dependiente y el valor de 1 cuando se ejecuta un Kriging ordinario; el objeto que contiene los datos "**spdf**"; un *grid* donde se almacenarán los datos y el modelo de semivariograma ajustado "**res.fit**". Esta función regresa un *grid* con el mapa interpolado, el cual se nombró como "**res.k**". Este *grid* es convertido a formato *raster* usando la función **raster()** y después se eliminan los valores donde no hay vegetación, utilizando la función **mask()**.

```
#Interpolación de los residuales

res.k = gstat::krige(res~1, spdf, grd, model = res.fit)

# [using ordinary kriging]

res_grd.k <- raster(res.k)
grd.res <- mask(res_grd.k, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**grd.res**" para obtener el mapa de riqueza de especies, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**.

```
#Gráfica del mapa de residuales

my_col <- c("yellow4", "orange", "greenyellow",
            "green","forestgreen","darkgreen")

tm1 <- tm_shape(grd.res) +
 tm_raster(style= "cont", n=10,
           palette =my_col,
           title="Residuales en regresión") +
 tm_grid() +
 tm_layout(legend.outside = TRUE,
           legend.position= c("right", "bottom"))

tm1
```



```
#Grabar mapa en disco

tmap_save(tm1,
          filename= paste(dir.MAPS,"mapa_res_regresion.tiff", sep="/"),
          width=1720, height=1070, asp= 0)
```

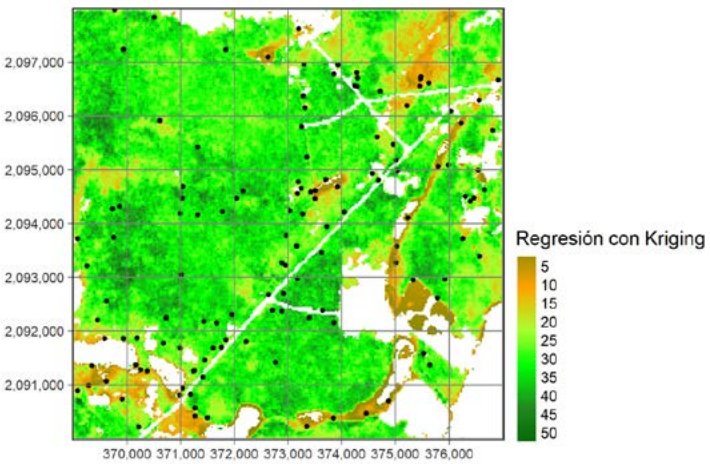
Se define la función **replaceNA()**, la cual es utilizada para remplazar valores nulos con el valor de 0 en un objeto de tipo *raster*. Esta función fue aplicada a los objetos *raster* de la estimación con regresión y de los residuales "**map.regf2**" y "**grd.res**". Enseguida se suman los objetos *raster* generados para incluir la estimación, tanto de la regresión "**raster.reg**" como de los residuales "**raster.res**". A este nuevo objeto *raster* "**reg_kri**" se le aplicó la máscara que corresponde a todos los tipos de vegetación. Esto con el objeto de eliminar los valores donde no hay vegetación o que no pudieron ser eliminados antes; para esto se usa la función **mask()**. Finalmente, se realiza una reclasificación para sustituir los valores negativos con 0.

```
#Función para remplazar valores nulos con 0 en raster

replaceNA <- function(x, na.rm, ...){
 if(is.na(x[1]))
  return(0)
 else
  return(x)
 }

raster.reg <- calc(map.regf2, fun = replaceNA)
raster.res <- calc(grd.res, fun = replaceNA)
```

```
#Mapa de regresión y suma de residuales de regresión con Kriging

reg_kri <- raster.reg + raster.res
map.reg_kri <- mask(reg_kri, rc, maskvalue= 0)

map.reg_kri2 <- reclassify(map.reg_kri,
          cbind(-Inf, 0, NA),
          right=FALSE) #sustituir negativos con NA
```

Se imprime el objeto *raster* "**map.reg_kri2**" para obtener el mapa de riqueza de especies con la técnica de regresión con Kriging, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**Reg_con_kri.tiff**", usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
            "green","forestgreen","darkgreen")

tm1 <- tm_shape(map.reg_kri2) +
  tm_raster(style= "cont", n=10,
            palette =my_col,
            title="Regresión con Kiging") +
  tm_shape(spdf) +
  tm_dots(size=0.05) +
  tm_grid() +
  tm_layout(legend.outside = TRUE,
            legend.position= c("right", "bottom"))

tm1
```



```
#Grabar mapa en disco

tmap_save(tm1,
          filename= paste(dir.MAPS,"mapa_regresion_kri.tiff", sep="/"),
          width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(map.reg_kri2,
          paste(folder, "Reg_con_kri.tiff", sep= "/"),
          overwrite= TRUE)
```

Para llevar a cabo la validación cruzada usando regresión con Kriging, primero se validan los residuales. Para ello se crean el vector "**res.out**", que tiene el mismo número de elementos que el objeto "**spdf**" de la clase **SpatialPointDataFrame**. Después se realiza la interpolación usando la función **krige()**, excluyendo un dato a la vez. Esto permite obtener una lista de valores estimados de residuales "**res.out**", los cuales se suman al vector de valores estimados con regresión "**reg.out**" y se almacenarán en el vector "**datos$predrk**". Estos serán comparados con aquellos valores de campo. Enseguida, se calcula el valor del **RMSE** usando el vector de los datos estimados "**datos$predrk**" y el de los observados "**datos$Num-esp**". Por último, se redondea este valor a 2 dígitos y se guarda en la variable "**RMSE_i**". Adicionalmente, se ajusta un modelo de regresión lineal entre los valores observados y predichos usando la función **lm()**, para ello se utilizó la fórmula **datos$predrk ~ datos$Num_esp**. El valor del coeficiente de determinación $\mathbf{R^2}$ es igual a 0.63.

```
#Validación cruzada de residuales

res.out <- vector(length = length(spdf))

for (i in 1:length(spdf)) {
 res.out[i] <- krige(res ~ 1, spdf[-i,],
          spdf[i,],
          model = res.fit)$var1.pred
```

```
 }

# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
...
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
```

```
#Vector de predichos más residuales: regresión con Kriging
datos$predrk <- reg.out + res.out
```

```
#Cálculo del RMSE para regresión con Kriging

rekg.rmse <- RMSE(datos$Num_esp, datos$predrk)
RMSE_i <- round(rekg.rmse, digits = 2)
```

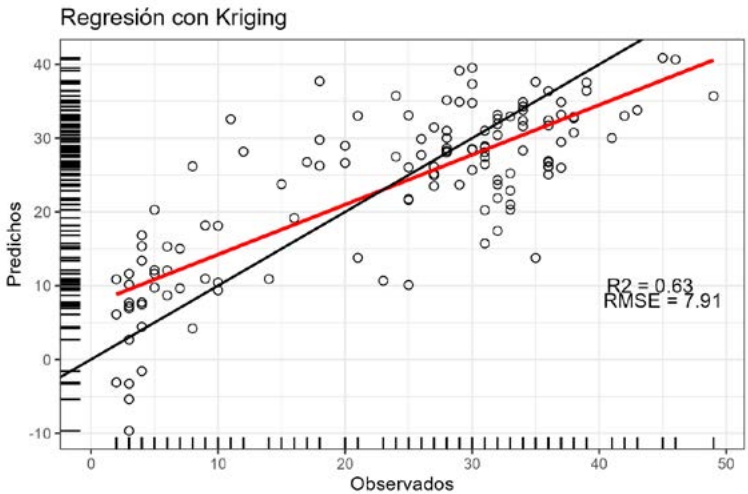
```
#Modelo de regresión entre predichos vs. observados para
#regresión con Kriging

mod1 = lm(datos$predrk ~ datos$Num_esp)
summary(mod1)

#
# Call:
# lm(formula = datos$predrk ~ datos$Num_esp)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -19.1540  -3.6863   0.5818   3.7918  18.0878
```

```
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)   7.46543    1.23786   6.031 1.63e-08 ***
# datos$Num_esp  0.67576    0.04624  14.615  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 6.777 on 128 degrees of freedom
# Multiple R-squared:  0.6253,  Adjusted R-squared:  0.6224
# F-statistic: 213.6 on 1 and 128 DF,  p-value: < 2.2e-16

r2 = format(summary(mod1)$r.squared, digits = 2)
```

Para obtener de manera gráfica estos resultados, se utiliza la función **ggplot()**. Al principio se coloca la función **ggplot()**, que tiene como parámetros el vector de valores predichos "**predrk**" y el de valores observados "**Num_esp**" del *dataframe* "**datos**". Se utilizaron funciones similares a como se hizo la validación cruzada anteriormente. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Gráfica de los valores predichos vs. valores observados

p1 <- ggplot(datos, aes(x=Num_esp, y=predrk)) +
 geom_point(size=2, shape=21 )+
 geom_rug()+
 geom_smooth(method=lm, se=FALSE, size=1,color="red")+
 labs(title= "Regresión con Kriging",
   x="Observados", y = "Predichos")+
 expand_limits(x = 0, y = 0)+
 theme_bw() +
 geom_abline(size=0.7) +
 annotate("text",x = 44, y = 10,
          label = paste("R2 =", r2), size = 4)+
```

```
 annotate("text", x = 45, y = 8,
          label = paste("RMSE =", RMSE_i), size = 4)

p1
```




```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_regre_Krig.tiff", sep="/"),
    width=6, height=4)
```

7.3 Random Forest de regresión con Kriging

El algoritmo de Random Forest (RF) de regresión, se basa en el ensamblaje de múltiples iteraciones de árboles de decisión, usada para generar predicciones sobre una variable de interés. Este método se ha convertido en uno de los análisis de datos más importantes para la predicción. El algoritmo de RF, a diferencia de la regresión lineal, no tiene requisitos considerando la forma de la función de distribución de probabilidades de la variable de estudio. Además, RF permite estimar la importancia de

variables medidas por la disminución media en la precisión de la predicción antes y después de permutar las variables (Hengl et al., 2015).

El método de interpolación de Random Forest con Kriging funciona de manera similar a la regresión con Kriging (Hengl et al., 2015). Primero, se ajusta el modelo de RF utilizando como variable dependiente el atributo de interés (en este caso la riqueza de especies), y un conjunto de variables explicativas (variables derivadas de las imágenes Landsat 7). Después se obtienen los residuales del modelo, para los cuales se aplicará Kriging ordinario con el objeto de hacer una estimación espacial de los residuales en toda el área de estudio. Finalmente, los valores predichos utilizando regresión con RF y los residuales estimados con Kriging, se suman para estimar la distribución espacial de la riqueza de especies.

Para iniciar con la construcción del modelo para la estimación de la riqueza de especies usando Random Forest con Kriging, se cargan los paquetes que se utilizarán en este módulo.

```
#Cargar paquetes que se utilizarán en este módulo

library(gstat)
library(randomForest)
library(ModelMap)
library(raster)
library(sp)
```

El total de observaciones que se dispone para estimar la distribución espacial de la riqueza de especies serán utilizadas, tanto para la calibración como para la validación del modelo, es decir, se utilizará una validación cruzada. Sin embargo, es posible utilizar un grupo de datos independientes para la validación si así se requiere. Se creará una variable que guardará el nombre del archivo con los datos de campo y los valores de reflectancia y textura de la imagen Landsat7 (**dat.csv**).

En caso de dividir el archivo de datos en dos, se escribiría el nombre del archivo con los datos de calibración (**d_train.csv.csv**) y el nombre del archivo con los datos de

validación (**d_test.csv**). Para hacer esta división del archivo de datos original en aquellos que se utilizarán durante la calibración y la validación del modelo, se utiliza la función **get.test()** del paquete **ModelMap**. Es importante recordar que, en este caso, los datos de entrenamiento del modelo y los que se usarán para la validación son los mismos, por lo cual este procedimiento no se ejecutó.

```
#Definición de tipos de archivos

qdatafn <- "dat.csv" #archivo de datos
```

```
#Cuando la validación se hace con un grupo de datos independientes

#trainfn <- "d_train.csv" # archivo de datos para entrenar el modelo
#testfn <- "d_test.csv" # archivo de datos para validar el modelo

#Los datos de entrenamiento y validación serán iguales
#get.test(proportion.test=0.3,
#       qdatafn=qdatafn,
#       seed=49,
#       folder=folder,
#       qdata.trainfn=trainfn,
#       qdata.testfn=testfn)
```

Se continúa con la definición de los parámetros que se utilizarán en el modelo, y lo primero es decidir sobre el tipo de modelo a utilizar. El paquete **ModelMap** tiene tres tipos de modelos distintos, pero en este caso se utiliza el Random Forest, por lo que se coloca RF. Adicionalmente, se debe especificar el nombre del archivo donde se guardará la salida del modelo; todos los archivos de salida del modelo tendrán el prefijo "**Nesp**", como se especifica a continuación.

```
#Definición de los parámetros del modelo

mtype <- "RF" # el Modelo es de Random Forest
MODELfn <- "Nesp"
```

Se define la lista de variables explicativas y cuáles de estas son categóricas. En el estudio de caso de este manual, se tienen las capas de las imágenes Landsat 7 con los valores de reflectancia (TM3 y TM4), el índice NDVI calculado con los valores de reflectancia y las 24 medidas de textura, es decir, 8 medidas de textura calculadas para cada capa (TM3 TM4 y NDVI), en total 27 variables. Como ninguna de estas variables es categórica, se define este parámetro como FALSE.

```
#Definición variables explicativas del modelo

predList <- c( "TM3", "TM4", "NDVI", "TM3_med", "TM3_var",
          "TM3_hom", "TM3_con","TM3_dis", "TM3_entr",
          "TM3_smo", "TM3_cor", "TM4_med", "TM4_var",
          "TM4_hom", "TM4_con", "TM4_dis", "TM4_entr",
          "TM4_smo", "TM4_cor","NDVI_med", "NDVI_var",
          "NDVI_hom", "NDVI_con", "NDVI_dis",
          "NDVI_entr", "NDVI_smo", "NDVI_cor")

predFactor <- FALSE
```

Se define la variable de respuesta y si esta es continua, binaria o categórica. La variable de respuesta que se calculó con los datos de campo y que está en el archivo (**dat. csv**), es "**Num_esp**". Esta variable es continua. Los parámetros que restan por definir son la semilla aleatoria para el modelo y la columna de la base de datos que contiene un identificador único para cada renglón. Este identificador será utilizado para crear el archivo de salida con los valores observados y predichos de la riqueza de especies cuando se ejecute la función para la validación del modelo, en este caso "**ID**".

```
#Definición de variable respuesta del modelo

resname <- "Num_esp"
restype <- "continuous"
seed <- 25
urowname <- "ID" #identificador unico de cada renglon
```

Para construir el modelo se utiliza la función **model.build()** del paquete **ModelMap**. Esta función regresa como resultado un objeto con el modelo. Por otro lado, esta función guarda un archivo de texto con los parámetros que se utilizaron en el modelo. Hay que recordar que el prefijo de nombre de archivo que se utilizó fue **Nesp**, entonces, este archivo de texto tiene por nombre **Nesp_model_biulding_arguments.txt** y se almacena en el directorio de trabajo establecido.

La función utilizada para construir el modelo tiene los siguientes parámetros que se establecieron arriba: 1) el tipo de modelo (**model.type**), 2) los datos de calibración (**qdata.trainfn**), 3) el directorio donde se guardan los resultados (**folder**), 4) la columna que contiene los identificadores únicos (**unique.rwoname**), 5) el nombre del modelo con el que se guardarán los archivos de salida (**MODELfn**), 6) la lista de las variables explicativas (**predList**), 7) el tipo de datos de las variables explicativas (**predFactor**) (recordar que, como no hay variables categóricas, colocamos **FALSE**), 8) el nombre de la variable de respuesta "**response.name**", 9) el tipo de datos de la variable de respuesta (**response.type**), y 10) la semilla aleatoria.

Después de ejecutar la función, se imprime el objeto "**model.sp**". Se observa que se utilizaron 500 árboles en este modelo y que 9 variables fueron las más importantes. Además, se proporciona el promedio del cuadrado de los residuales (68.76), que al aplicarle la raíz cuadrada se obtiene el **RMSE** = 8.29 especies. Por último, tenemos el porcentaje de variabilidad explicada por el modelo, que en este caso es del 58.4 % (o bien $\mathbf{R^2}$ = 0.58).

```
#Construcción del modelo usando Random Forest

model.sp <- model.build( model.type=mtype,
                qdata.trainfn=qdatafn,
                folder=folder,
                unique.rowname=urowname,
                MODELfn=MODELfn,
                predList=predList,
                predFactor=predFactor,
                response.name=resname,
                response.type=restype,
```

```
                seed=seed,
                na.action = "na.omit")

# mtry = 9  OOB error = 69.20556
# Searching left ...
# mtry = 5  OOB error = 69.64732
# -0.006383263 0.05
# Searching right ...
# mtry = 18     OOB error = 76.01928
# -0.09845619 0.05

model.sp

#
# Call:
#  randomForest(x = qdata.x, y = qdata.y, ntree = 500, mtry = 9,     replace = TRUE,
importance = TRUE, proximity = FALSE)
#               Type of random forest: regression
#                     Number of trees: 500
# No. of variables tried at each split: 9
#
#         Mean of squared residuals: 68.76155
#                   % Var explained: 58.39
```

La función **model.diagnostics()** permite llevar a cabo la validación, en este caso será una validación cruzada. Esta función regresa un *dataframe* que tiene como nombre "**val.sp**" y las variables siguientes: el ID; los valores observados y los valores predichos de riqueza de especies. Adicionalmente, los guarda en un archivo ***.csv**, que en este caso es (**Nesp_pred.csv**) en el directorio de trabajo. Por otro lado, crea varios gráficos y tablas de resultados. Esta función utiliza algunos parámetros similares a la función **model.build()**; los parámetros adicionales son: 1) el tipo de predicción (**prediction.type**), que en este caso es para la validación cruzada "**CV**", y 2) la especificación del formato en el que se grabarán los archivos de salida **device.type**, que es un "**pdf**". Los archivos de salida corresponden a: 1) gráfica con la importancia de las variables en el modelo

grabado con el nombre de archivo (**Nesp_pred_importance.pdf**), 2) una gráfica de correlaciones entre las variables predichas (**Nesp_pred_corrplot.pdf**), 3) el diagrama de dispersión entre los valores observados y predichos de riqueza de especies (**Nesp_pred_scatterplot.pdf**), y 4) los errores en el modelo en función del número de árboles (**Nesp_pred_error.pdf**). Enseguida, se ejecuta la función **model.diagnostics()** con sus respectivos parámetros. Con la función **head()** se imprime el nombre de la variable del *dataframe* "**val.sp**" y las observaciones de los primeros renglones.

```
#Debido a que qdata.testfn=FALSE, se realiza una validación cruzada que #se indica
en prediction.type="CV"

val.sp <- model.diagnostics( model.obj=model.sp,
                qdata.trainfn=qdatafn,
                qdata.testfn=FALSE,
                folder=folder,
                MODELfn=MODELfn,
                unique.rowname=urowname,
                #se realiza una validacion cruzada
                prediction.type="CV",
                device.type=c("pdf"),
                cex=1.2)

# [1] "Begining 10-fold cross validation:"
#
# cv.index 2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18 20 21 23 24 25 26 27 28 29 30
#    1 0 0 1 0 0 0 0 0  0 0 0 0 1 0 0 0 2 0 1 0 0 1 1 0 0 0
#    2 1 0 0 2 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 2
#    3 0 1 0 0 1 0 0 0  0 0 1 0 0 0 1 0 0 0 0 0 0 0 2 0 1 0
#    4 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 2 0 0
#
# cv.index 31 32 33 34 35 36 37 38 39 41 42 43 45 46 49
#    1 1 1 0 0 0 0 0 2 0 0 1 0 0 1 0
#    2 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0
#    3 0 1 0 1 1 1 0 0 0 0 0 1 0 0 1
```

```
#     4  1 0 1 2  0 1 1 0  0 0 0 0  0 0 0 0
# [1] "starting fold 1"
# [1] "starting factor checks"
# [1] "starting fold 2"
```

```
head(val.sp)
```

```
#   ID obs    pred VFold
# 1  1  27 28.90910     6
# 2  2  24 36.07663     6
# 3  3  32 31.44620     8
# 4  4  34 36.83157     7
# 5  5  25 37.01807     7
# 6  6  31 29.04810     1
```

Para obtener un mapa con la distribución de la riqueza de especies en el área estudiada, se utiliza el modelo de Random Forest obtenido anteriormente, así como las capas con los valores de reflectancia y textura de las imágenes de Landsat 7. Para realizar esta actividad, primero se asigna en el directorio donde se encuentran los datos de las imágenes ("c**:/met_int/imgrf/**") a la variable "**dir.img**". Después, se utiliza la función **list.files()** con 2 argumentos: el primero "**path**", que tiene el directorio completo donde se encuentran los archivos; aquí se utilizó el que se guardó en "**dir.img**"; y el segundo argumento es "**pattern**", el cual se refiere a un grupo de caracteres que contiene los archivos que se van a leer; aquí se utilizó la extensión de los archivos que es "**.img**". Esta función produce un vector de caracteres con los nombres de archivos leídos y los directorios en donde estos se encuentran. Este vector se guardó en "**l.img**". Posteriormente, se imprime este vector, que arroja la lista de archivos leídos y su ubicación.

```
#Obtención del mapa
#Directorio donde se encuentran las imágenes predictoras/explicativas

dir.img <-"imgrf"
dir.img <- paste(folder,dir.img,sep="/")
```

```
#Crear una lista con las imágenes predictoras

l.img <- list.files(path = dir.img, pattern = ".img$")
l.img

# [1]"NDVI.img"     "NDVI_con.img" "NDVI_cor.img" "NDVI_dis.img"
# [5]"NDVI_entr.img" "NDVI_hom.img" "NDVI_med.img" "NDVI_smo.img"
# [9]"NDVI_var.img" "TM3.img"     "TM3_con.img"  "TM3_cor.img"
#[13]"TM3_dis.img"  "TM3_entr.img" "TM3_hom.img"  "TM3_med.img"
#[17]"TM3_smo.img"  "TM3_var.img"  "TM4.img"     "TM4_con.img"
#[21]"TM4_cor.img"  "TM4_dis.img"  "TM4_entr.img" "TM4_hom.img"
#[25]"TM4_med.img"  "TM4_smo.img"  "TM4_var.img"
```

Se crea un *dataframe* con el nombre de "**rastLUTfn.a**", que contiene 3 columnas: la primera con el directorio y nombre del archivo de la imagen (***.img**); la segunda columna es el nombre de la variable explicativa y, por último, el número de capa del *raster* que corresponde a la variable explicativa. Al imprimir este *dataframe*, se observa la información de las 3 imágenes con las 3 columnas (V1, V2 y V3).

```
#Crear un dataframe de imágenes

rastLUTfn <- data.frame()

for (i in 1: length(l.img)){
 rastLUTfn[i,1]<- paste(dir.img, l.img[i], sep = "/")
 rastLUTfn[i,2]<-sub(".img$", "", l.img[i])
 rastLUTfn[i,3]<- 1 #número de bandas
 }


rastLUTfn
```

```
#                       V1      V2 V3
# 1     C:/met_int/imgrf/NDVI.img     NDVI  1
```

```
# 2  C:/met_int/imgrf/NDVI_con.img  NDVI_con 1
# 3  C:/met_int/imgrf/NDVI_cor.img  NDVI_cor 1
# 4  C:/met_int/imgrf/NDVI_dis.img  NDVI_dis 1
# 5 C:/met_int/imgrf/NDVI_entr.img NDVI_entr 1
# 6  C:/met_int/imgrf/NDVI_hom.img  NDVI_hom 1
# 7  C:/met_int/imgrf/NDVI_med.img  NDVI_med 1
# 8  C:/met_int/imgrf/NDVI_smo.img  NDVI_smo 1
# 9  C:/met_int/imgrf/NDVI_var.img  NDVI_var 1
# 10     C:/met_int/imgrf/TM3.img     TM3 1
# 11  C:/met_int/imgrf/TM3_con.img  TM3_con 1
# 12  C:/met_int/imgrf/TM3_cor.img  TM3_cor 1
# 13  C:/met_int/imgrf/TM3_dis.img  TM3_dis 1
# 14 C:/met_int/imgrf/TM3_entr.img TM3_entr 1
# 15  C:/met_int/imgrf/TM3_hom.img  TM3_hom 1
# 16  C:/met_int/imgrf/TM3_med.img  TM3_med 1
# 17  C:/met_int/imgrf/TM3_smo.img  TM3_smo 1
# 18  C:/met_int/imgrf/TM3_var.img  TM3_var 1
# 19     C:/met_int/imgrf/TM4.img     TM4 1
# 20  C:/met_int/imgrf/TM4_con.img  TM4_con 1
# 21  C:/met_int/imgrf/TM4_cor.img  TM4_cor 1
# 22  C:/met_int/imgrf/TM4_dis.img  TM4_dis 1
# 23 C:/met_int/imgrf/TM4_entr.img TM4_entr 1
# 24  C:/met_int/imgrf/TM4_hom.img  TM4_hom 1
# 25  C:/met_int/imgrf/TM4_med.img  TM4_med 1
# 26  C:/met_int/imgrf/TM4_smo.img  TM4_smo 1
# 27  C:/met_int/imgrf/TM4_var.img  TM4_var 1
```

Se define el nombre del mapa de salida, que en este caso es (**map_RF.img**). Para obtener el mapa de riqueza de especies y sus respectivos mapas de incertidumbre, se utiliza la función **model.mapmake()** del paquete **ModelMap**. Esta función tiene varios parámetros, aquí se utilizaron los siguientes: 1) el objeto modelo que se utiliza para la predicción (**model.obj**), en este caso es "**model.sp**", que usa un modelo de Random Forest; 2) el directorio de salida definido que está almacenado en la variable "**folder**"; 3) el *dataframe* que contiene la ubicación de las imágenes

que se utilizan como variables explicativas (**rastLUTfn**); 4) el nombre del mapa de salida y su ubicación (**OUTPUTfn**); 5) el nombre del prefijo que tendrán todos los archivos de salida, que en este caso se definió como M**ODELfn.a=Nesp**; 6) el último parámetro se especifica si se quiere obtener el mapa de desviaciones estándar y de la media para el cálculo de un mapa de coeficiente de variación.

Esta función genera diferentes archivos de salida que se mencionan a continuación: 1) un archivo de texto (**Nesp_projections**), que tiene la proyección geográfica de las 27 capas utilizadas como variables explicativas; 2) cuatro archivos de imágenes (***.img**), el primero con la distribución espacial de la riqueza de especies (**Nesp.img**), mientras que las otras tres imágenes tienen la media, desviación estándar y coeficiente de variación de la riqueza de especies (**Nesp_mean.img**, **Nesp_stdev.img**, **Nesp_coefv.img**).

```
#Definir el nombre del mapa de salida

OUTPUTfn <-"map_RF.img"
```

```
#Crear una imagen con el mapa del Num esp predicho

model.mapmake( model.obj=model.sp,
        folder=folder,
        rastLUTfn=rastLUTfn,
        # parametros del mapa
        OUTPUTfn=OUTPUTfn,
        MODELfn= MODELfn,
        map.sd = TRUE)
```

```
# [1] "model.NA.ACTION: NULL"
# [1] "Using default 'na.action' of \"na.omit\""
# [1] "starting production prediction"
# [1] "predFactor: "
# [1] "all predictor layer rasters match"
# [1] "brick done"
# [1] "write start C:/met_int/map_RF.img"
```

```
# [1] "starting row by row predictions"
# [1] "    rows = 1"
# [1] "    rows = 2"
# [1] "    rows = 3"
# [1] "    rows = 4"
...
# [1] "    rows = 264"
# [1] "    rows = 265"
# [1] "    rows = 266"
# [1] "    rows = 267"
# [1] "    rows = 268"
# [1] "ARGfn: C:/met_int/map_RF_mapmake_arguments.txt"
```

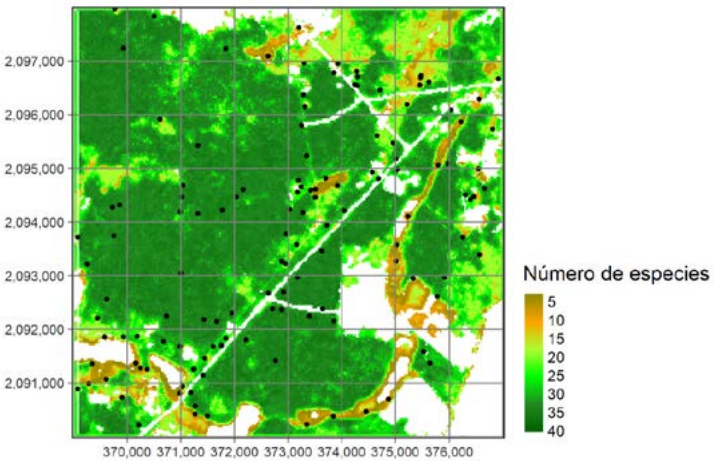
El mapa de salida se convierte de formato **.img** a *raster*, utilizando la función **raster()**, posteriormente se eliminan los valores donde no hay vegetación utilizando la función **mask()** y su máscara "**rc**", que indica áreas de no bosque.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

map.rf <- raster(OUTPUTfn)
map.rarf <- mask(map.rf, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**map.rarf**" para obtener el mapa de riqueza de especies, con la técnica de RF, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**randomf.tiff**", usando la función **writeRaster()**.

```
#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
        "green","forestgreen","darkgreen")

tm1 <- tm_shape(map.rarf) +
 tm_raster(style= "cont", n=10,
```

```
        palette =my_col,
        title="Número de especies") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid() +
 tm_layout(legend.outside = TRUE,
        legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
       filename= paste(dir.MAPS,"mapa_randomforest.tiff", sep="/"),
       width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(map.rarf,
        paste(folder, "randomf.tiff", sep= "/"),
        overwrite= TRUE)
```

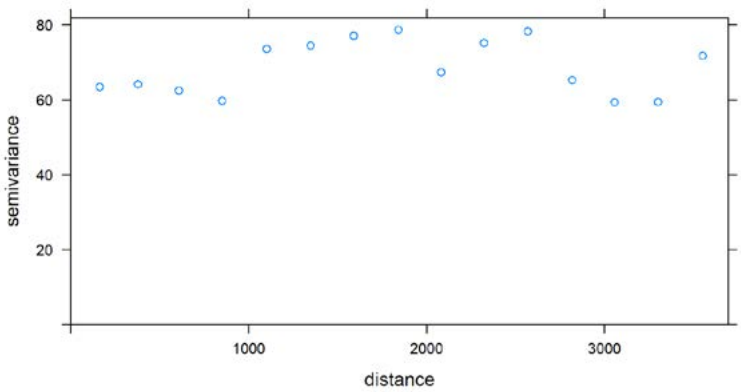
Los residuales se obtuvieron de la diferencia entre los valores de riqueza de especies observados en campo "**datos$Num_esp**" y los valores predichos con el modelo de RF ajustado "**model.sp$predicted**", y se guardaron en el objeto "**spdf**" como "**resrf**".

Para continuar con el procedimiento de interpolación de RF con Kriging, se calcula el variograma experimental de los residuales utilizando la función **variogram()**. Se utilizaron 2 argumentos: un objeto de tipo "***formula***", que especifica la variable dependiente y las posibles covariables; en este caso, como no existen covariables se colocó el numero 1; y un objeto de la clase **SpatialPointDataFrame** "**spdf**" con los datos que son los residuales "**resrf**". Esta función regresa un objeto de la clase **variogram** que nombramos como "**res.rf_vgm**", el cual se imprimió con la función **plot()**.

Se utilizó la función **tiff()** para almacenar este gráfico en un archivo de tipo **TIFF**. En esta función se especifica el directorio y nombre del archivo como se ha hecho anteriormente, el nombre del archivo es "**var_krig_rf1.tiff**" y el del directorio es "**C.\met_int\Maps**". Se especifican, además, el ancho y alto de la imagen y sus unidades, así como la resolución que tendrá la misma en dpi's.

```
#Cálculo de residuales del modelo Random Forest

spdf$resrf <- datos$Num_esp - model.sp$predicted
```

```
#Crear el variograma experimental

res.rf_vgm = variogram(resrf ~ 1, spdf)
plot(res.rf_vgm)
```



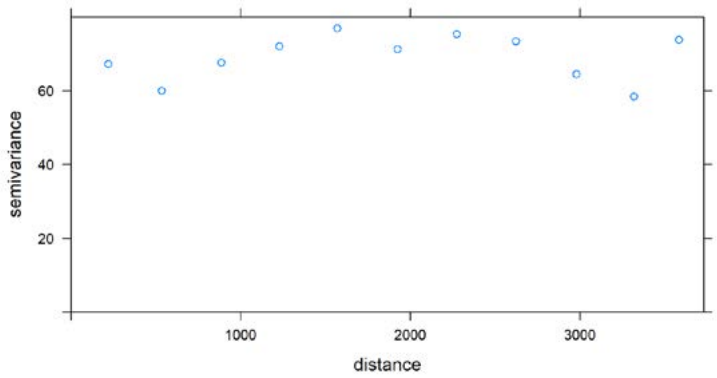

```
#Guardar la gráfica
```

```
tiff(paste(dir.MAPS,"var_krig_rf1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(res.rf_vgm)
dev.off()
```

En un segundo variograma experimental, se usó la función **variogram()** con 3 argumentos: la fórmula donde se especifica la variable dependiente; el objeto "**spdf**" con los datos "**resrf**" y el ancho de los intervalos de distancia en los que se agrupan los pares de puntos de datos para estimaciones de semivarianza, que en este caso fue de 350. Es decir, se redujo el número de intervalos para tener un mejor ajuste del modelo. Esta función regresa un objeto de la clase **variogram**, que nombramos como "**res.rf_vgm**", el cual se imprimió con la función **plot()** y se guardó en el disco con la función **tiff()**.

```
#Crear el variograma experimental ancho 350
```

```
res.rf_vgm = variogram(resrf ~ 1, spdf, width = 350)
plot(res.rf_vgm)
```



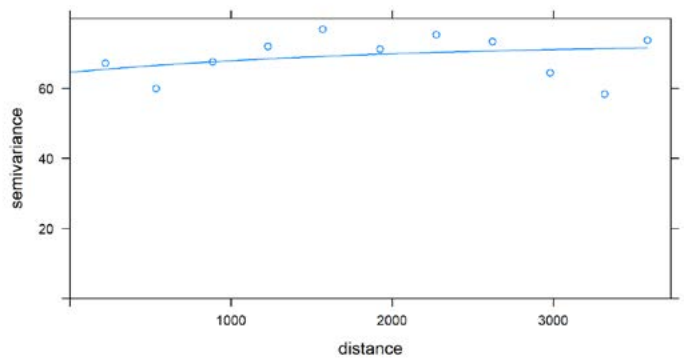

```
#Guardar la gráfica

tiff(paste(dir.MAPS,"var_krig_rf2.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(res.rf_vgm)
dev.off()
```

Enseguida, se ajustó un modelo al variograma experimental usando la función **fit.variogram()**, con 2 argumentos: un objeto de tipo **variogram** "**res.fr_vgm**", que fue la salida al ejecutar la función **variogram()**; el modelo, que se especifica por medio de la función **vgm()**. Esta función genera un modelo de semivariograma y tiene diferentes argumentos: un valor de referencia del "**sill**", en este caso fue de 40; el tipo de modelo "**Exp**"; un valor que indique el rango y el componente "**nugget**". Como resultado, la función **fit.variogram()** regresa un objeto al cual llamamos "**res.fit**" y se imprime.

Se ajustó un modelo exponencial al semivariograma experimental. La varianza estructural, que determina la dependencia espacial explicada por el modelo y calculada como (varianza total– varianza nugget/varianza total) *100, fue (8.35/72.98) *100, tuvo un valor de 11.4 %. Esto significa que una parte de la variabilidad de la distribución espacial de la riqueza de especies no es explicada por el modelo (88.6 %).

El modelo ajustado se imprime con la función **plot()** y se guardó en el disco con la función **tiff()**.

```
#Ajustar una función al variograma experimental

res.rf_fit = fit.variogram(res.rf_vgm,
                 model = vgm(50, "Exp", 1200, 1))
res.rf_fit

#  model    psill   range
# 1  Nug 64.633167   0.000
# 2   Exp  8.354036 1988.267

plot(res.rf_vgm, res.rf_fit)
```




```
#Guardar la gráfica

tiff(paste(dir.MAPS,"var_fitkrig_rf1.tiff", sep="/"),
   width = 7, height = 4, units = 'in', res = 300)
plot(res.rf_vgm, res.rf_fit)
dev.off()
```

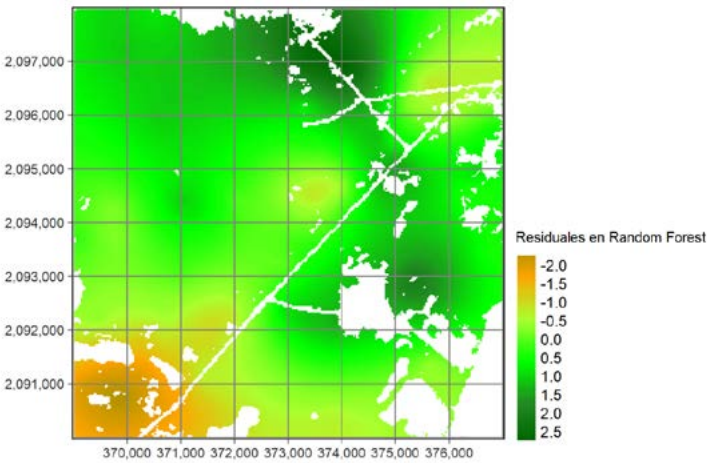
El mapa de los residuales usando interpolación con Kriging, se obtiene utilizando la función **krige()** con 4 argumentos: la fórmula que define la variable dependiente y el valor de 1 cuando se ejecuta un Kriging ordinario; el objeto que contiene los datos "**spdf**"; un *grid* donde se almacenarán los datos y el modelo de semivariograma ajustado

"**res. rf_ fit**". Esta función regresa un *grid* con el mapa interpolado, el cual se nombró como "**res.rf**". Este *grid* es convertido a formato *raster* usando la función **raster()** y después se eliminan los valores donde no hay vegetación utilizando la función **mask()**.

```
#Interpolación de los residuales

res.rf = gstat::krige(resrf ~ 1, spdf, grd,
                model = res.rf_fit)

# [using ordinary kriging]

#spplot(res.rf["var1.pred"],main = "Reciduales RF - Kriging " )

res_grd.rf <- raster(res.rf)
grd.res2 <- mask(res_grd.rf, rc, maskvalue= 0)
```

Se imprime el objeto *raster* "**grd.res2**" para obtener el mapa de riqueza de especies, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en disco usando la función **tmap_save()**.

```
#Gráfica del mapa de residuales

my_col <- c("yellow4", "orange", "greenyellow",
        "green","forestgreen","darkgreen")

tm1 <- tm_shape(grd.res2) +
 tm_raster(style= "cont", n=10,
        palette =my_col,
 midpoint = NA,
        title="Residuales en Random Forest") +
 tm_grid() +
 tm_layout(legend.outside = TRUE,
        legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
      filename= paste(dir.MAPS,"mapa_res_randomf.tiff", sep="/"),
      width=1720, height=1070, asp= 0)
```

Para llevar a cabo la validación cruzada de la estimación de los residuales, primero se crea un vector "**res1.out**", que tiene el mismo número de elementos que el objeto "**spdf**" de la clase **SpatialPointDataFrame**. Después se realiza la interpolación usando la función **krige()**, excluyendo un dato a la vez. Esto permite obtener una lista de valores estimados de la riqueza de especies que se almacena en el vector "**res1.out**", que serán comparados con aquellos medidos en los sitios de muestreo.

```
#Validación de los residuales

res1.out <- vector(length = length(spdf))

for (i in 1:length(spdf)) {
 res1.out[i] <- krige(resrf ~ 1, spdf[-i,],
                spdf[i,],
                model = res.rf_fit)$var1.pred
}
```

```
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
...
```

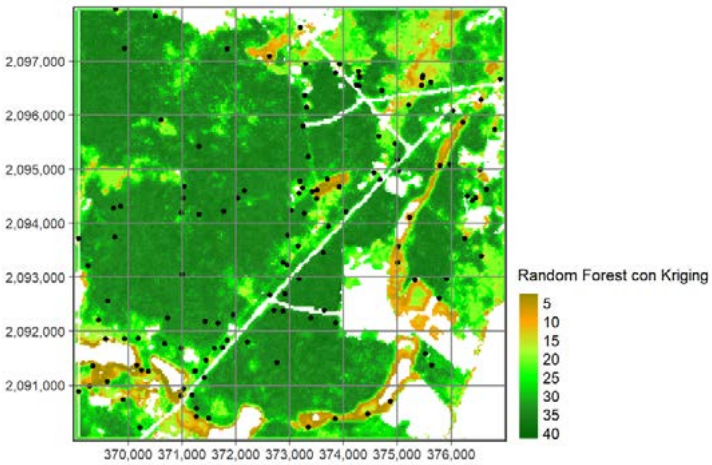
```
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
# [using ordinary kriging]
```

Se suman los objetos *raster* generados para incluir la estimación, tanto de Random Forest "**map.rarf**", como de los residuales "**gr.res2**", y se creó un nuevo objeto *raster* llamado "**rf_kri**".

Se imprime el objeto *raster* "**rf_kri**" para obtener el mapa de riqueza de especies con la técnica de RF con Kriging, utilizando las funciones de la librería **tmap**. Finalmente, este mapa se guarda en el disco usando la función **tmap_save()**. Además, se guarda el objeto *raster* en el disco con el nombre "**RF_con_kri.tiff**", usando la función **writeRaster()**.

```
#Mapa de RF y suma de residuales, Random Forest con Kriging

rf_kri <- map.rarf + grd.res2

#Visualización de la gráfica del mapa interpolado y sitios de muestreo

my_col <- c("yellow4", "orange", "greenyellow",
        "green","forestgreen","darkgreen")

tm1 <- tm_shape(rf_kri) +
 tm_raster(style= "cont", n=10,
```

```
        palette =my_col,
        title="Random Forest con Kriging") +
 tm_shape(spdf) +
 tm_dots(size=0.05) +
 tm_grid() +
 tm_layout(legend.outside = TRUE,
        legend.position= c("right", "bottom"))

tm1
```




```
#Grabar mapa en disco

tmap_save(tm1,
      filename= paste(dir.MAPS,"mapa_randomf_kri.tiff", sep="/"),
      width=1720, height=1070, asp= 0)
```

```
#Guardar en el disco el mapa creado

writeRaster(rf_kri,
        paste(folder, "RF_con_kri.tiff", sep= "/"),
        overwrite= TRUE)
```

Se suman los vectores de la validación en RF y de los residuales.

```
#Vector de predichos más residuales: RF con Kriging

val.sp$predrf <- val.sp$pred + res1.out
```

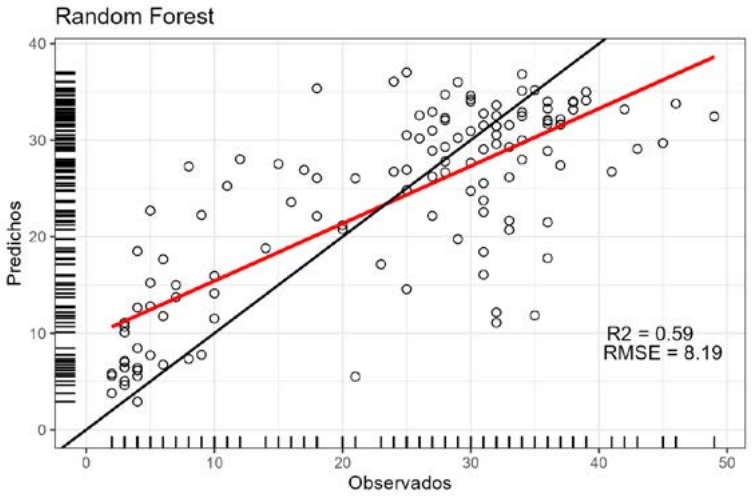
El cálculo de los estadísticos de validación en el modelo de RF, se realiza calculando el valor del **RMSE** usando el vector de los datos estimados "**datos$pred**" y el de los observados "**val.sp$obs**". Por último, se redondea este valor a 2 dígitos y se guarda en la variable "**RMSE_i**". Adicionalmente, se ajusta un modelo de regresión lineal entre los valores observados y predichos usando la función **lm()**, para ello se utilizó la fórmula **val.sp$pred ~ val.sp$obs**. El valor del coeficiente de determinación $\mathbf{R^2}$ es igual a 0.59.

```
#Cálculo del RMSE para el modelo de Random Forest

rf.rmse <- RMSE(val.sp$obs, val.sp$pred)
RMSE_i <- round(rf.rmse, digits = 2)

mod = lm(val.sp$pred ~ val.sp$obs)
summary(mod)

#
# Call:
# lm(formula = val.sp$pred ~ val.sp$obs)
#
# Residuals:
#    Min     1Q  Median     3Q    Max
# -18.450  -4.589   0.691   3.919  15.192
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  9.45726    1.16343   8.129 3.21e-13 ***
# val.sp$obs   0.59520    0.04346  13.697  < 2e-16 ***
# ---
```

```
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 6.369 on 128 degrees of freedom
# Multiple R-squared:  0.5944,  Adjusted R-squared:  0.5913
# F-statistic: 187.6 on 1 and 128 DF,  p-value: < 2.2e-16

r2 = format(summary(mod)$r.squared, digits = 2)
```

Para obtener de manera gráfica estos resultados, se utiliza la función **ggplot()**. Al principio se coloca la función **ggplot()**, que tiene como parámetros el vector de valores predichos "**pred**" y el de valores observados "**obs**" del *dataframe* "**val.sp**". Se utilizaron funciones similares a como se hizo la validación cruzada anteriormente. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Gráfica de los valores predichos vs. valores observados Random Forest

p1 <- ggplot(val.sp, aes(x=obs, y=pred)) +
  geom_point(size=2, shape=21 )+
 geom_rug()+
 geom_smooth(method=lm, se=FALSE, size=1,color="red")+
 labs(title= "Random Forest",
     x="Observados", y = "Predichos")+
 expand_limits(x = 0, y = 0)+
 theme_bw() +
 geom_abline(size=0.7) +
 annotate("text",x = 44, y = 10, label = paste("R2 =", r2), size = 4)+
 annotate("text",x = 45, y = 8, label = paste("RMSE =", RMSE_i), size = 4)

p1
```




```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_randomforest.tiff", sep="/"),
    width=6, height=4)
```

El cálculo de los estadísticos de validación en el modelo de RF con Kriging, se realiza calculando el valor del **RMSE** usando el vector de los datos estimados "**val.sp$predrf**" y el de los observados "**val.sp$obs**". Por último, se redondea este valor a 2 dígitos y se guarda en la variable "**RMSE_i**". Adicionalmente, se ajusta un modelo de regresión lineal entre los valores observados y predichos usando la función **lm()**, para ello se utilizó la fórmula **val.sp$predrf ~ spdf$Num_esp**. El valor del coeficiente de determinación $\mathbf{R^2}$ es igual a 0.59.

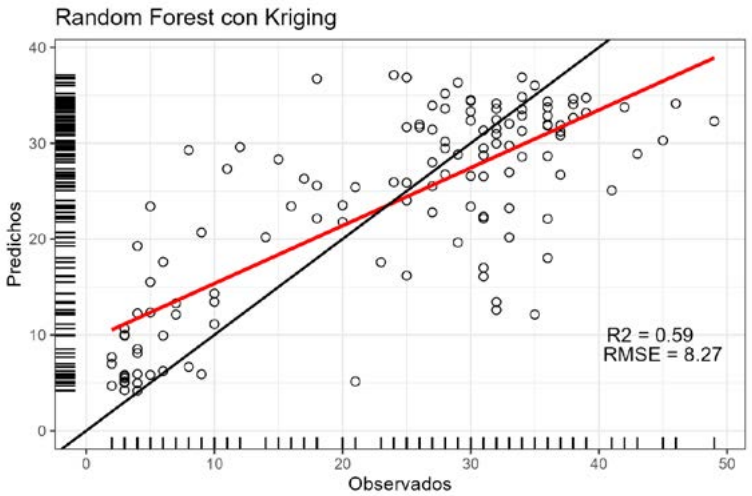
```
#Cálculo del RMSE para el modelo de Random Forest con Kriging

rf_kg.rmse <- RMSE(val.sp$obs, val.sp$predrf)
RMSE_i <- round(rf_kg.rmse, digits = 2)

mod1 = lm(val.sp$predrf ~ spdf$Num_esp)
```

```
summary(mod1)

#
# Call:
# lm(formula = val.sp$predrf ~ spdf$Num_esp)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -18.3094  -5.2238   0.4429   3.9080  16.5325
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)   9.3212     1.1994   7.772 2.21e-12 ***
# spdf$Num_esp   0.6036     0.0448  13.473  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 6.566 on 128 degrees of freedom
# Multiple R-squared:  0.5864,  Adjusted R-squared:  0.5832
# F-statistic: 181.5 on 1 and 128 DF,  p-value: < 2.2e-16

r2 = format(summary(mod1)$r.squared, digits = 2)
```

Para obtener de manera gráfica estos resultados, se utiliza la función **ggplot()**. Al principio se coloca la función **ggplot()**, que tiene como parámetros el vector de valores predichos "**predrf**" y el de valores observados "**obs**" del *dataframe* "**val.sp**". Se utilizaron funciones similares a como se hizo la validación cruzada anteriormente. Este conjunto de funciones es asignado al objeto de tipo **ggplot** llamado "**p1**", que después se imprime. Finalmente, se utiliza la función **ggsave()** para grabar la gráfica en el disco.

```
#Gráfica de los valores predichos vs. valores observados Random Forest # con Kriging

p1 <- ggplot(val.sp, aes(x=obs, y=predrf)) +
 geom_point(size=2, shape=21 )+
```

```
 geom_rug()+
 geom_smooth(method=lm, se=FALSE, size=1,color="red")+
 labs(title= "Random Forest con Kiging",
    x="Observados", y = "Predichos")+
 expand_limits(x = 0, y = 0)+
 theme_bw() +
 geom_abline(size=0.7) +
 annotate("text",x = 44, y = 10,
       label = paste("R2 =", r2), size = 4)+
 annotate("text",x = 45, y = 8,
       label = paste("RMSE =", RMSE_i), size = 4)

p1
```




```
#Grabar gráfica

ggsave(p1,
    file=paste(dir.MAPS,"gra_randomf_Krig.tiff", sep="/"),
    width=6, height=4)
```

8. Comparación de métodos de interpolación

Los diferentes métodos de interpolación pueden producir distintas representaciones espaciales, por lo que se requiere un conocimiento profundo del fenómeno de estudio para evaluar cuál método se acerca más a la realidad. El uso de un método inapropiado o parámetros no factibles, producen como resultado un modelo distorsionado de la distribución espacial del atributo de interés. En términos generales, los métodos de interpolación global presentan una distribución de los datos que no concuerda mucho con la distribución real de la riqueza de especies en el área de estudio. Por ejemplo, en el método de modelos de clasificación; a pesar de que las clases de vegetación identificadas pueden explicar la variabilidad espacial de la riqueza de especies, como lo demuestra el análisis de varianza, solamente existe un valor único dentro de cada clase para predecir la riqueza de especie y no se representa la variación dentro de clases, lo cual no es una representación realista de la distribución de esta variable en el área de estudio.

Una manera de evaluar si un método de interpolación es apropiado, consiste en realizar una evaluación cuantitativa de las capacidades predictivas de la técnica de interpolación. En este caso, para los métodos locales de interpolación y aquellos que usan datos auxiliares, se midió el desempeño de las técnicas de interpolación en términos de la precisión de las estimaciones, a través de las desviaciones entre los datos medidos y sus correspondientes valores estimados usando una validación cruzada. Existen diferentes estadísticas para realizar estas comparaciones. Aquí se utilizó la raíz de la media de los errores al cuadrado (**RMSE**, por sus siglas en inglés), que es una medida de la precisión de la predicción. Esta estadística es sensitiva a puntos de influencia, es decir, tiende a enfatizar los errores más grandes. También se utilizó el coeficiente

de determinación ($\mathbf{R^2}$), que es el porcentaje de variabilidad explicado por el modelo. Los resultados que comparan las técnicas de interpolación local y los métodos que utilizan datos auxiliares en términos de la precisión de las estimaciones (errores de estimación) se presentan en el siguiente cuadro.

Cuadro 1. Estadísticas de validación cruzada para las diferentes técnicas de interpolación.

Clasificación	Procedimiento de interpolación	RMSE	R^2	Corr
Métodos globales	Distancia Inversa Ponderada P = 2.0	10.79	0.30	0.55
	Distancia Inversa Ponderada P = 3.0	11.08	0.30	0.55
	Interpolación con Kriging	10.67	0.31	0.56
Métodos locales con datos auxiliares	Distancia Inversa dentro de estratos	7.56	0.66	0.81
	Kriging dentro de estratos	7.11	0.69	0.83
	Regresión	8.12	0.60	0.77
	Regresión con Kriging	7.91	0.63	0.79
	Random Forest	8.19	0.59	0.77
	Random Forest con Kriging	8.27	0.59	0.77

En el cuadro anterior, se puede observar que el desempeño de las técnicas de interpolación que combinan algún método de interpolación con el uso de datos auxiliares, en términos de precisión de las estimaciones es mejor que la de los métodos locales. Las predicciones de validación cruzada muestran que los métodos que usan datos auxiliares mejoraron significativamente la precisión de las estimaciones en comparación con los métodos locales, como lo demuestra el aumento de $\mathbf{R^2}$ y la disminución de los errores cuadráticos medios (RMSE).

Los resultados del **Cuadro 1** comparando los métodos de interpolación locales, revelan que, aunque el método de la distancia inversa tiene la ventaja de una relativa simplicidad y facilidad de procesamiento, este método es uno de los menos precisos y tiene uno de los niveles más altos de errores entre todos los procedimientos de interpolación. Adicionalmente, la interpolación de distancia inversa produjo patrones cir-

culares alrededor de los puntos de muestreo, lo que se considera una caracterización poco realista de la distribución espacial de la riqueza de especies en el campo.

Las diferencias entre las clases de vegetación consideradas en el área estudiada tienen implicaciones importantes para el desempeño de los métodos de interpolación probados en este estudio. Tales diferencias sugieren que cuando el método de interpolación se ejecuta para toda el área, el desempeño es más pobre. Sin embargo, la relativa uniformidad de las clases de vegetación muestra mejoras de las interpolaciones de Kriging y distancia inversa dentro de los estratos. Esto también sugiere que hay discontinuidades de los valores de riqueza de plantas en los límites del tipo de vegetación. En consecuencia, cuando las especies en distintos hábitats (clases de vegetación) tienen un patrón de distribución espacial significativamente diferente, como en este caso, la estimación final mejorará si los procedimientos de interpolación se aplican por separado dentro de cada clase de vegetación (Hernández-Stefanoni et al., 2006).

Los resultados obtenidos de las comparaciones de los diferentes métodos de interpolación para estimar la riqueza de especies vegetales, cuando se incorporan datos de percepción remota como información auxiliar, permiten tener algunas conclusiones. Primero, debemos enfatizar que los métodos que usan información auxiliar funcionaron mejor que el Kriging ordinario, demostrando una mejora en la precisión de las estimaciones cuando se usan indicadores de datos de sensores remotos para la predicción de la riqueza de especies basada en datos de campo (Hernández-Stefanoni et al., 2011). También es relevante que el modelo de regresión y el de Random Forest funcionaron mejor que el Kriging ordinario, lo que implica fuertes correlaciones entre la riqueza de especies de plantas y los datos de percepción remota. Se debe enfatizar que el potencial de los métodos que usan datos auxiliares para mejorar las estimaciones sobre el Kriging ordinario es alto, solo si las asociaciones entre las variables primarias y secundarias son sólidas y significativas (Simbahan et al., 2006; Hernández-Stefanoni et al., 2011). A pesar de que el modelo de regresión y el de Random Forest son considerados como métodos de interpolación globales, estos utilizan información local que permite refinar las estimaciones. Dicho de otra manera, los resultados de la estimación en estos modelos dependen tanto de la calidad de predicción del modelo como de la calidad y detalle de las superficies continuas que tienen los datos auxiliares (Burrough et al., 1998). Adicionalmente, los modelos que usaron datos

auxiliares y que tenían en cuenta la autocorrelación espacial, mejoraron la precisión de las estimaciones en comparación con los modelos de regresión y Random Forest. Las predicciones de validación cruzada mostraron un **R^2** más alto o **RMSE** más bajo.

Se demostró que la riqueza de especies de árboles está fuertemente relacionada con los datos de sensores remotos del área de estudio. Específicamente, los análisis con regresión y Random Forest sugieren que la riqueza de especies de árboles está fuertemente relacionada con la reflectancia de las bandas roja (TM3) e infrarroja cercana (TM4), el NDVI y las medidas de textura (Vieira et al., 2003; George-Chacon et al., 2019).

Los mapas de biodiversidad producidos mediante modelos precisos pueden contribuir a subsanar la limitación que enfrentan las y los administradores de recursos naturales y quienes se avocan a su conservación, respecto a la falta de información continua sobre los patrones de distribución de la riqueza de especies (Conroy et al., 1996; Araújo et al., 2006) y brindan orientación sobre la selección y eficacia de las áreas de protección de la naturaleza. También pueden ayudar a evaluar las respuestas de las especies al cambio climático global y a valorar la influencia de las especies exóticas (Rodríguez et al., 2007). Esta situación ha llevado a múltiples esfuerzos por desarrollar diferentes procedimientos de estimación destinados a llenar los vacíos existentes en nuestro conocimiento. En lugar de ser una solución real, la existencia de una gran cantidad de formas diferentes de estimar la información no disponible se está convirtiendo cada vez más en un problema en sí mismo, ya que no es obvio cuál usar si se quiere hacer la mejor estimación posible. En este contexto, es de sumo interés para la comunidad investigadora evaluar el desempeño de los diferentes procedimientos que se han implementado es este manual.

9. Consideraciones finales

En este manual se presentan los *scripts* en R para implementar los métodos de interpolación espacial más utilizados. Se ejemplifican estos procedimientos para producir mapas con la distribución espacial de la riqueza de especies, usando datos de campo e información de imágenes de satélite. Sin embargo, estos procedimientos podrían ser utilizados para cualquier otra variable de interés y con diferentes variables auxiliares como clima, topografía, suelo, entre otros, siempre y cuando las variables auxiliares estén disponibles en superficies continuas, tales como WorldClim (https://www.worldclim.org).

Entre los diferentes métodos de interpolación espacial presentados en este manual, la regresión múltiple con Kriging y Random Forest con Kriging tuvieron un mejor desempeño, no solo debido a su mayor potencial para aumentar la precisión de los mapas de riqueza de especies, sino también por la flexibilidad en el modelado de relaciones multivariadas entre la riqueza de especies y los datos de sensores remotos. Estos procedimientos ofrecen una gran oportunidad para la producción de mapas precisos de la riqueza de especies a partir de indicadores ampliamente accesibles, de bajo costo y de detección remota. Sin duda, estos modelos pueden ser valiosas herramientas para orientar futuros esfuerzos encaminados a comprender, conservar y gestionar bosques tropicales muy diversos.

Debido a que el objetivo de este manual fue el de implementar en R diferentes métodos de interpolación, y para no tener demasiada información, no se presenta ningún *script* en R para el procesamiento de imágenes de satélite. A pesar de que existen algunos manuales para utilizar imágenes en el mapeo de atributos de la vegetación usando R, como el *Mapeo de la biomasa aérea de los bosques mediante datos de sensores remotos y R* de Hernández-Stefanoni et al., (2021), es conveniente continuar la investigación e

implementación de nuevas aplicaciones con R, dado que es un *software* de uso libre que cualquiera puede utilizar.

Las observaciones anteriores nos indican que el mapeo de diferentes atributos de la vegetación, tales como la riqueza de especies, es un área activa para los investigadores e investigadoras que buscan mejorar la precisión de los mapas.

Referencias

Anys, H., Bannari, A., He, D. C., & Morin, D. (1994). Texture analysis for the mapping of urban areas using airborne MEIS-II images. *Proceedings of the First International Airborne Remote Sensing Conference and Exhibition*, 3, 231-245.

Araújo, M. B., & Guisan, A. (2006). Five (or so) challenges for species distribution modelling. *Journal of Biogeography*, 33(10), 1677-1688. https://doi.org/10.1111/j.1365-2699.2006.01584.x

Burrough, P. A., & McDonnell, R. A. (1998). *Principles of Geographical Information Systems. Spatial Information Systems and Geostatistics*. Oxford University Press.

Cabrera Cano, E. F., Sousa Sánchez, M., & Téllez Valdez, O. (1982). *Imágenes de la flora quintanarroense*. Centro de Investigaciones de Quintana Roo.

Conroy, M. J., & Noon, B. R. (1996). Mapping of species richness for conservation of biological diversity: conceptual and methodological issues. *Ecological Applications*, 6, 763-773. https://doi.org/10.2307/2269481

Chen, L., Wang, Y., Ren, C., Zhang, B., & Wang, Z. (2019). Assessment of multi-wavelength SAR and multispectral instrument data for forest aboveground biomass mapping using random forest kriging. *Forest Ecology and Management*, 447, 12-25. https://doi.org/10.1016/j.foreco.2019.05.057

Dormann, C. F., McPherson, J. M., Araújo, M. B., Bivand, R., Bolliger, J., Carl, G., Davies, R. G, Hirzel, A., Jetz, W., Kissling, W. D., Kühn, I., Ohlemüller, R., Peres-Neto, P. R, Reineking, B., Schröder, B., Schurr F. M., & Wilson, R. (2007). Methods to account for spatial autocorrelation in the analysis of species distributional data: a review. *Ecography*, 30(5), 609-628. https://doi.org/10.1111/j.2007.0906-7590.05171.x

Fahrig, L. (2003). Effects of habitat fragmentation on biodiversity. *Annual Review of Ecology, Evolution and Systematic*, 34, 487-515. https://doi.org/10.1146/annurev.ecolsys.34.011802.132419

Georgakarakos, S., & Kitsiou, D. (2008). Mapping abundance distribution of small pelagic species applying hydroacoustics and Co-kriging techniques. *Hydrobiologia*, 612, 155-169. https://doi.org/10.1007/s10750-008-9484-z

George-Chacon, S. P., Dupuy, J. M., Peduzzi, A., & Hernández-Stefanoni, J. L. (2019). Combining high resolution satellite imagery and lidar data to model woody species diversity of tropical dry forests. *Ecological Indicators*, 101, 975-984. https://doi.org/10.1016/j.ecolind.2019.02.015

Gillespie, T. W., Foody, G. M., Rocchini, D., Giorgi, A. P., & Saatchi, S. (2008). Measuring and modelling biodiversity from space. *Progress in Physical Geography: Earth and Environment*, 32(2), 203-221. https://doi.org/10.1177/0309133308093606

Gotelli, N. J., & Colwell, R. K. (2001). Quantifying biodiversity: procedures and pitfalls in the measurement and comparison of species richness. *Ecology Letters*, 4(4), 379-391. https://doi.org/10.1046/j.1461-0248.2001.00230.x

Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. IEEE Transactions on Systems, Man and Cybernetics, SMC-3(6), 610-621. https://doi.org/10.1109/TSMC.1973.4309314

Hengl, T., Heuvelink, G. B., Kempen, B., Leenaars, J. G. B., Walsh, M. G., Shepherd, K. D., Sila, A., MacMillan, R. A., Mendes de Jesus, J., Tamene, L., & Tondoh, J. E. (2015). Mapping Soil Properties of Africa at 250 m Resolution: Random Forests Significantly Improve Current Predictions. *PLoS ONE*, 10 (6). https://doi.org/10.1371/journal.pone.0125814

Hengl, T., Nussbaum, M., Wright, M. N., Heuvelink, G. B., & Gräler, B. (2018). Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables. *PeerJ*, 6. https://doi.org/10.7717/peerj.5518

Hernández-Stefanoni, J. L., Castillo-Santiago M. A., Andrés-Mauricio, J., Mas, J. F., Tun-Dzul, F. J., & Dupuy, J. M. (2021). *Mapeo de la biomasa aérea de los bosques mediante datos de sensores remotos y R*. El Colegio de la Frontera Sur y Centro de Investigación Científica de Yucatán, A.C. https://www.researchgate.net/publication/349142637_Mapeo_de_la_biomasa_aerea_de_los_bosques_mediante_datos_de_sensores_remotos_y_R

Hernández-Stefanoni, J. L., Gallardo-Cruz, J. A., Meave, J. A., & Dupuy, J. M. (2011). Combining geostatistical models and remotely sensed data to improve tropical tree richness mapping. *Ecological Indicators*, 11(5), 1046-1056. https://doi.org/10.1016/j.ecolind.2010.11.003

Hernández-Stefanoni, J. L., Gallardo-Cruz, J. A., Meave, J. A., Rocchini, D., Bello-Pineda, J., & López-Martínez, J. O. (2012). Modeling α-and β-diversity in a tropical forest from remotely sensed and spatial data. *International Journal of Applied Earth Observation and Geoinformation*, 19, 359-368. https://doi.org/10.1016/j.jag.2012.04.002

Hernández-Stefanoni, J. L., & Ponce-Hernández, R. (2006). Mapping the spatial variability of plant diversity in a tropical forest: comparison of spatial interpolation methods. *Environmental Monitoring and Assessment*, 117(1), 307-334. https://doi.org/10.1007/s10661-006-0885-z

Le Quéré, C., Moriarty, R., Andrew, R. M., Canadell, J. G., Sitch, S., Korsbakken, J. I., Friedlingstein, P., Peters, G. P., Andres, R. J., Boden, T. A., Houghton, R. A., House, J. I., Keeling, R. F., Tans, P., Arneth, A., Bakker, D. C. E, Barbero, L., Bopp, L., Chang, … Zeng, N. (2015). Global Carbon Budget 2015. *Earth System Science Data*, 7, 349-396. https://doi.org/10.5194/essd-7-349-2015

Lechner, A. M., Foody, G. M., & Boyd, D. S. (2020). Applications in remote sensing to forest ecology and management. *One Earth*, 2(5), 405-412. https://doi.org/10.1016/j.oneear.2020.05.001

Portillo-Quintero, C., Sanchez-Azofeifa, A., Calvo-Alvarado, J., Quesada, M., & do Espirito Santo, M. M. (2015). The role of tropical dry forests for biodiversity, carbon and water conservation in the neotropics: lessons learned and opportunities for its sustainable management. *Regional Environmental Change*, 15, 1039-1049. https://doi.org/10.1007/s10113-014-0689-6

Powers, J. S., Feng, X., Sanchez-Azofeifa, A., & Medvigy, D. (2018). Focus on tropical dry forest ecosystems and ecosystem services in the face of global change. *Environmental research letters*, 13(9). https://doi.org/10.1088/1748-9326/aadeec

Plotkin, J. B., Potts, M. D., Yu, D. W., Bunyavejchewin, S., Condit, R., Foster, R., Hubbell, S., LaFrankie, J., Manokaran, N., Lee, H. S., Sukumar, R., Nowak, M. A., & Ashton, P. S. (2000). Predicting species diversity in tropical forests. *Pro-*

ceedings of the National Academy of Science, 97(20), 10850-10854. https://doi.org/10.1073/pnas.97.20.10850

Rocchini, D., Nagendra, H., Ghate, R., & Cade, B. S. (2009). Spectral distance decay: assessing species beta-diversity by quantile regression. *Photogrammetric, Engineering & Remote Sensing*, 75(10),1225-1230. http://atree.org/sites/default/files/articles/ja_2009_1225-1230.pdf

Rodríguez, J. P., Brotons, L., Bustamante, J., & Seoane, J. (2007). The application of predictive modelling of species distribution to biodiversity conservation. *Diversity and Distributions*, 13(3), 243-251. https://www.jstor.org/stable/4539918

Sales, M. H., Souza Jr., C. M., Kyriakidis, P. C., Roberts, D. A., & Vidal, E. (2007). Improving spatial distribution estimation of forest biomass with geostatistics: a case study for Rondonia, Brazil. *Ecological Modeling*, 205(1-2), 221-230. https://doi.org/10.1016/j.ecolmodel.2007.02.033

Simbahan, G. C., Dobermann, A., Goovaerts, P., Ping, J., & Haddix, M. (2006). Fine-resolution mapping of soil organic carbon based on multivariate secondary data. *Geoderma*, 132(3-4), 471-489. https://doi.org/10.1016/j.geoderma.2005.07.001

Swamy, L., Drazen, E., Johnson, W. R., & Bukoski, J. J. (2018). The future of tropical forests under the United Nations Sustainable Development Goals. *Journal of Sustainable Forestry*, 37(2), 221-256. https://doi.org/10.1080/10549811.2017.1416477

Vieira, I. M. G., De Almeida, A. S., Davidson, E. A., Stone, T. A., Carvalh, C. J. R., & Guerrero, J. B. (2003). Classifying successional forest using Landsat spectral properties and ecological characteristics in eastern Amazonia. *Remote Sensing of Environment*, 87(4), 470-481. https://doi.org/10.1016/j.rse.2002.09.002

Viedma, O., Torres, I., Pérez, B., & Moreno, J. M. (2012). Modeling plant species richness using reflectance and texture data derived from QuickBird in a recently burned area of Central Spain. *Remote Sensing of Environment*, 119, 208-221. https://doi.org/10.1016/j.rse.2011.12.024

Webster, R., & Oliver, M. A. (2001). *Geostatistics for Environmental Science*. John Wiley and Sons.

Zar, J. H. (1999). *Biostatistical analysis* (4th ed.). Prentice Hall.

Apéndice

Los datos requeridos, así como los *scripts* utilizados en este manual, se encuentran disponibles para su descarga en la siguiente liga: https://www.cicy.mx/unidad-de-recursos-naturales/investigador/jose-luis-hernandez-stefanoni/proyecto-metodos-de-interpolacion. En esta se provee un archivo comprimido que contiene los datos y *scripts* dentro de una carpeta que llamamos **met_int**. La estructura de la carpeta contiene lo siguiente:

1. Un archivo nombrado como **Interpolacion.R**, el cual está en el formato de R y contiene el código para poder ejecutar los diferentes métodos de interpolación espacial. Los *scripts* están escritos en el lenguaje de programación R diseñado y se pueden utilizar en la interfaz RStudio.

2. Archivo **dat.csv**, que se encuentra en formato de datos separado por comas y tiene la terminación ***.csv**. Este archivo contiene una base de datos con las coordenadas en **UTM** de cada uno de los sitios de campo, el cálculo de la riqueza de especies de árboles en cada sitio, así como las variables explicativas de reflectancia y textura extraídas de imágenes del satélite Landsat 7 y que son utilizadas como variables explicativas de los modelos de regresión y Random Forest.

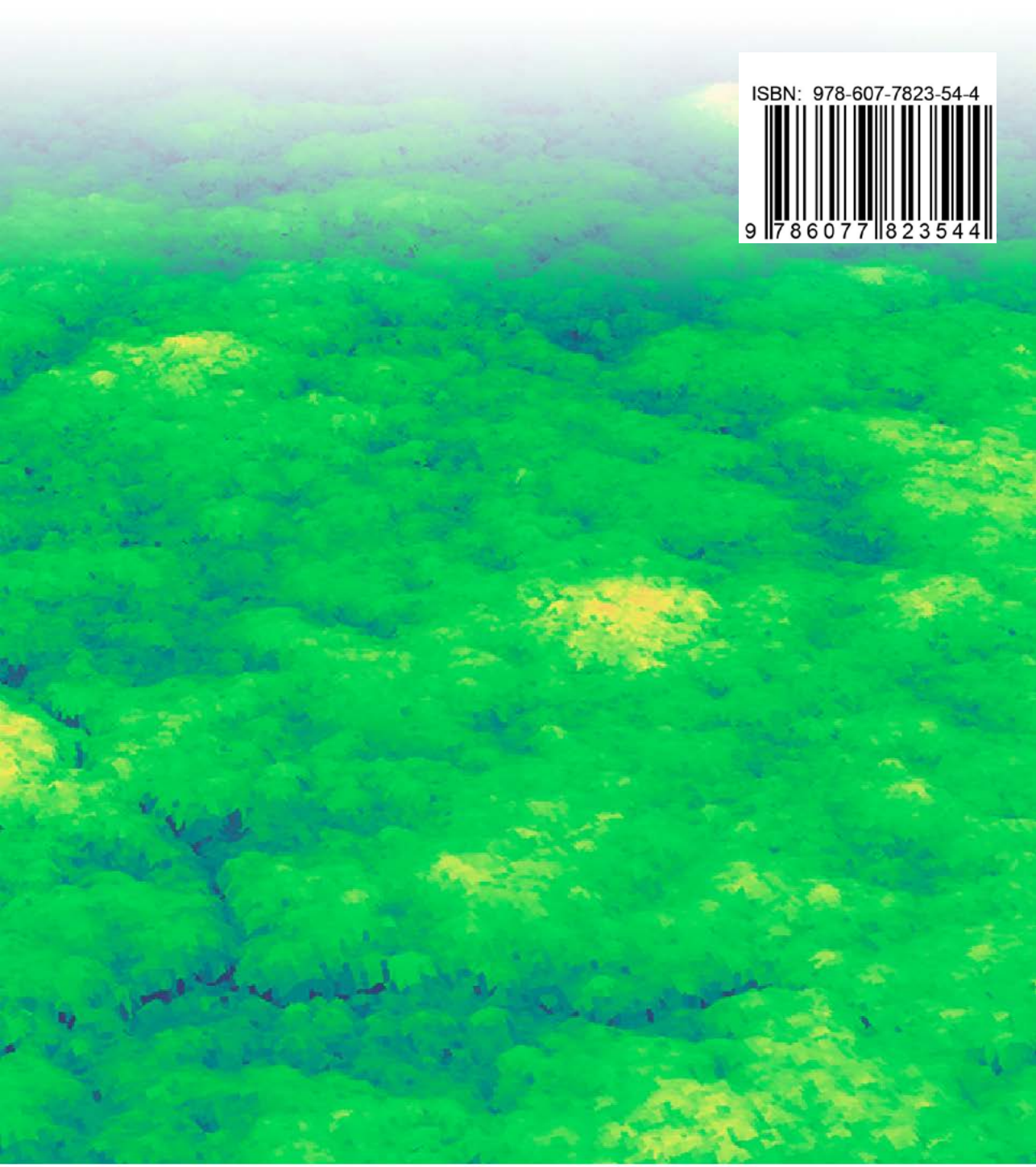
3. Archivo **sup_class_filter.tif**, el cual es un archivo de tipo *raster*, con formato **GeoTiFF** y una extensión ***.tif**. Este archivo contiene un mapa de coberturas del suelo obtenido de una clasificación supervisada de imágenes del satélite Landsat 7.

4. Carpeta **img (.../met_int/img)**, esta contiene archivos de tipo *raster*, en formato **GeoTIFF**, con extensión ***.tif**. Los archivos de esta carpeta fueron obtenidos de

imágenes del satélite Landsat 7 y corresponden a valores de reflectancia y texturas. Estas variables son utilizadas como variables explicativas en los métodos de interpolación de Regresión con Kriging. Los archivos son: (TM3.tif, TM4.tif, NDVI.tif, TM3_con.tif, TM3_dis.tif, TM3_med.tif, TM4_hom.tif, TM4_var.tif).

5. Carpeta **imgrf (.../met_int/imgrf)**, contiene archivos de tipo *raster* en formato **ERDAS IMAGINE** con extensión ***.img**. Los archivos de esta carpeta fueron obtenidos de imágenes del satélite Landsat 7 y corresponden a valores de reflectancia y texturas. Estas variables son utilizadas como variables explicativas en los métodos de interpolación de Random Forest con Kriging. Los archivos son:

1) NDVI.img
2) NDVI_con.img
3) NDVI_cor.img
4) NDVI_dis.img
5) NDVI_entr.img
6) NDVI_hom.img
7) NDVI_med.img
8) NDVI_smo.img
9) NDVI_var.img
10) TM3.img
11) TM3_con.img
12) TM3_cor.img
13) TM3_dis.img
14) TM3_entr.img
15) TM3_hom.img
16) TM3_med.img
17) TM3_smo.img
18) TM3_var.img
19) TM4.img
20) TM4_con.img
21) TM4_cor.img
22) TM4_dis.img
23) TM4_entr.img
24) TM4_hom.img
25) TM4_med.img
26) TM4_smo.img
27) TM4_var.img



ISBN: 978-607-7823-54-4

CONAHCYT
CONSEJO NACIONAL DE HUMANIDADES CIENCIAS Y TECNOLOGÍAS

CICY