Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación

Licenciatura en Ciencia de la Computación

TRABAJO DE DIPLOMA

Manipulación de series de datos Espacio-Temporales mediante el uso de formatos de datos científicos y geográficos en R

Autor: Yuniel Acosta Pérez

Tutor: Dr. Romel Vázquez Rodríguez

Santa Clara, 2017

Hago constar que el presente trabajo fue realizado en la Universidad Central Marta Abreu de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del tutor Firma del Jefe de Laboratorio

DEDICATORIA

Dedico los resultados de este trabajo a mi familia sin la cual no me hubiese sido posible llegar hasta aquí.

AGRADECIMIENTO

Quisiera agradecer a mi tutor Romel Vázquez Rodríguez, a mis compañeros de aula los cuales me han ayudado incondicionalmente y a todos los que de una forma u otra han apoyado este proyecto.

Resumen

En este trabajo se analizaron los formatos de datos científicos utilizados para manipular series de datos espacio-temporales y geoespaciales. Se estudió la conveniencia de desarrollar herramientas para la conversión de formatos de datos científicos que puedan almacenar datos espacio-temporales. Se desarrolló una nueva extensión (paquete) para R capaz de manipular y transformar, de forma eficiente, grandes volúmenes de datos científicos para series espacio-temporales y geoespaciales.

Para la validación y comprobación de los resultados se realizó un caso de estudio con datos globales de tipos de suelos, que contienen información sobre el uso del suelo a nivel mundial en el periodo 2001-2012. Los datos fueron obtenidos de la *Global Land Cover Facility* (GLCF), que pertenece al Departamento de Geografía de la Universidad de Maryland, y fueron creados con colaboración de la NASA. Este caso de estudio permitió crear archivos en formatos de datos científicos espacio-temporales que pueden ser utilizados por diversas instituciones que realicen investigaciones sobre el uso de los suelos. Las herramientas desarrolladas están basadas en software libre y pueden ser utilizadas en diferentes áreas de aplicación.

ABSTRACT

In this paper, we analyzed the scientific data formats used to manipulate spatial-temporal and geospatial data series. It was studied the convenience of developing tools for the conversion of scientific data formats that can store spatio-temporal data. A new extension (packet) was developed for R capable of efficiently manipulating and transforming large volumes of scientific data for spatio-temporal and geospatial series.

For the validation and verification of the results, a case study was carried out with global data of soil types, which contain information on land use worldwide in the period 2001-2012. The data were obtained from the Global Land Cover Facility (GLCF), which belongs to the Department of Geography of the University of Maryland, and were created with collaboration from NASA. This case study allowed the creation of archives in formats of spatiotemporal scientific data that can be used by several institutions that conduct research on the use of the soils. The tools developed are based on free software and can be used in different areas of application.

Tabla de Contenidos

Listado de Figuras

Listado de Tablas

INTRODUCCIÓN

En la actualidad se genera gran cantidad de información en disímiles formatos. Muchas de las instituciones científicas y centros de investigación han creado sus propios formatos para almacenar los grandes volúmenes de datos generados. El principal problema de estos datos radica en la estandarización y la interoperabilidad, pues a pesar de que existen múltiples estándares de almacenamiento de información, no todos son utilizados por las mismas aplicaciones informáticas, lo que puede generar problemas para la distribución y reutilización de la información (Roque 2014).

Los datos históricos se utilizan para realizar análisis de comportamiento de condiciones presentes y pasadas, así como para estimar proyecciones futuras. Los datos espacio-temporales se utilizan para realizar mapas, diagramas y modelos de superficies, para el mantenimiento y representación de condiciones reales (Iwamura et al. 2011). Se suele integrar la representación espacial con la representación del tiempo, lo que genera diferentes formatos (IBÁÑEZ & HOEHNE 2010). Algunos de estos formatos de datos espacio-temporales son: *XML, GML, KML, NetCDF* y *HDF* (Durango 2013). Estos formatos se utilizan, principalmente, para representar observaciones de oceanografía, calidad del agua, precipitación, ciencias atmosféricas y territoriales, entre otras. La selección del tipo de formato utilizado para los datos espacio-temporales depende del conocimiento que se tenga de ellos, del software utilizado y de la comunidad que los implemente.

En las últimas décadas se ha producido un desarrollo importante de la Estadística Espacio-Temporal, dada por la necesidad de analizar la evolución temporal del comportamiento espacial de magnitudes aleatorias que son de interés en estudios desarrollados en diversas áreas aplicadas tales como medioambiente, Geofísica, Biología y Medicina. En particular, en el contexto geo-estadístico, se ha producido un avance importante en la derivación de modelos flexibles para el procesamiento y análisis estadístico de datos espacio-temporales. Los enfoques adoptados se fundamentan esencialmente en la estadística espacial, dado su auge previo en este contexto (Salmerón Gómez 2008). R como lenguaje estadístico cuenta con diversos paquetes para manipulación de series de tiempo y datos espacio-temporales. Estos se encuentran organizados en vistas o temas en un repositorio oficial conocido como *The Comprehensive R Archive*

Network (CRAN). En el "CRAN TASK VIEWS" se encuentran las vistas "*SpatioTemporal*" y "*TimeSeries*" donde se resume un conjunto de paquetes para leer, escribir, visualizar y analizar series de tiempo y series de datos espacio-temporales.

Los datos geográficos son, por definición, información que varía en el espacio terrestre, es decir, que tienen asociada una coordenada geográfica que determina el lugar preciso que el dato representa en términos de elevación, temperatura, velocidad del viento, etc. A su vez, esta información suele registrarse en el tiempo con cierta regularidad. En algunas ocasiones se realiza con una resolución temporal definida y en otras se lleva a cabo eventualmente (Goodchild 2009).

Los datos espacio-temporales se gestionan principalmente con Sistemas de Información Geográfica (SIG). Un SIG puede reconocer y analizar las relaciones espaciales que existen en la información geográfica almacenada. Estas relaciones topológicas permiten crear modelos y análisis espaciales complejos. Adicionalmente, un SIG puede estar destinado al análisis de rutas, geo-estadística, álgebra de mapas, entre otros, con el uso de datos espacio-temporales (Goodchild 2009).

Desde los orígenes de los SIG, una de las preocupaciones principales ha sido la de representar de la mejor manera posible toda la información que podemos extraer de una zona geográfica dada, de tal modo que pueda almacenarse y analizarse en el entorno de un SIG (Olaya 2014). Entre los modelos lógicos principales para representar datos geográficos se encuentran el formato *raster* y vectorial que dan lugar a dos grandes tipos de capas de información espacial. En líneas generales podemos decir que el modelo *raster* se basa en una división sistemática del espacio, la cual cubre todo este, caracterizándolo como un conjunto de unidades elementales. El modelo vectorial, por su parte, no divide el espacio completamente, sino que lo define mediante una serie de elementos geométricos con valores asociados, siendo la disposición de estos no sistemática, sino guardando relación con los objetos geográficos presentes en la zona de estudio (Olaya 2014).

Antecedentes:

Este trabajo tiene como antecedentes proyectos desarrollados por el grupo de Computación Gráfica de nuestra universidad que han tenido como objetivo incorporar formatos de datos científicos en SIG libre. El trabajo ***Herramientas para la manipulación de formatos de datos***

científicos espacio-temporales en SIG, dotó a *gvSIG* de herramientas que permiten la manipulación de distintos tipos de formatos de datos científicos. También se puede mencionar la ***Herramienta de animación de NetCDF espacio-temporal*** realizado por el grupo de Computación Gráfica de nuestra universidad que permite a *gvSIG* hacer interesantes animaciones de archivos *NetCDF* igual a las realizadas por sistemas de información geográfica propietarios como *ArcGIS*.

En los últimos años ha surgido una tendencia a realizar investigaciones con el uso de paquetes matemáticos y estadísticos basados en *toolbox* especializados en diferentes materias o áreas de aplicación. *Matlab* y R son ejemplos de estos tipos de herramientas que permiten realizar fácilmente tareas de investigación en las más disimiles áreas.

Planteamiento del Problema

La manipulación de formatos de datos científicos para series espacio-temporales y geoespaciales en R se realiza de forma independiente por múltiples paquetes que permiten la Entrada/Salida de formatos de datos específicos. R no cuenta con una herramienta capaz de manipular y transformar múltiples formatos de datos científicos.

Objetivo general

Desarrollar un paquete en *R* capaz de manipular eficientemente los formatos de datos científicos y geoespaciales para series de datos espacio-temporales.

Objetivos Específicos

- Seleccionar de los principales formatos de datos científicos los más adecuados para la manipulación de series de datos espacio-temporales.
- Desarrollar un módulo para la manipulación de series espacio-temporales mediante el uso de formatos de datos científicos en R.
- Validar las herramientas implementadas mediante casos de estudio.

Para darle solución a la problemática planteada y responder a los Objetivos de la investigación fueron formuladas las preguntas de investigación.

Preguntas de Investigación

1. ¿Cuáles son los formatos de datos científicos más adecuados para la manipulación de series de datos espacio-temporales?
2. ¿Cómo se pueden integrar estos formatos de datos científicos en un lenguaje como R para manipular series de datos espacio-temporales?
3. ¿Qué beneficios brinda el uso de las herramientas desarrolladas para la manipulación de formatos de datos científicos en un lenguaje estadístico como R para la posterior animación de datos espacio-temporales?

Justificación de la investigación

Actualmente existe diversos tipos de formatos de datos científicos para almacenar grandes volúmenes de información, en este caso se justifica la realización de una investigación para determinar las ventajas y desventajas de estos formatos de datos científicos. Serán seleccionados los formatos más adecuados para la manipulación de datos espacio-temporales.

Entre los diferentes Sistemas de Información Geográfica existentes, A*rcGIS* y *gvSIG* por solo mencionar algunos, cuentan con herramientas para el trabajo con formatos de datos científicos, pero estos requieren de grandes capacidades de cómputo. En nuestro grupo de investigación se cuenta con el personal capacitado para realizar esta investigación donde se destaca la experiencia en el trabajo con estos formatos de datos. Se pretende determinar las bondades del empleo de lenguajes como *R* para manipular estos formatos de forma más óptima, rápida y con un consumo moderado de recursos computacionales.

1 FORMATOS DE DATOS CIENTÍFICOS PARA LA MANIPULACIÓN DE SERIES ESPACIO-TEMPORALES Y GEOESPACIALES.

En este capítulo se abordan los datos espacio-temporales, cuáles son sus características más importantes y la forma de representarlos. Haremos un estudio sobre algunos formatos de datos científicos espacio-temporales para conocer las potencialidades de cada uno de ellos, así como sus principales usos y aplicaciones. Además, se aborda sobre los sistemas de información geográfica (SIG) y su integración con los formatos de datos espacio-temporales. Por último, se tratan temas y aspectos importantes sobre el lenguaje de programación R.

1.1 Datos espacio-temporales

Los modelos espacio-temporales se construyen con datos que contiene la componente espacial, así como la temporal. Un ejemplo típico es el de una red de monitoreo (de un contaminante atmosférico, o una red de estaciones meteorológicas) en que los datos son reunidos a intervalos regulares, dígase todos los días, cada semana o en algún periodo de tiempo. Así, el análisis de los datos tiene que tomar cuenta la dependencia espacial de los lugares monitoreados, y que las observaciones de monitoreo normalmente no son independientes, pero conforman una serie de tiempo. En otras palabras, uno debe tomar en cuenta las correlaciones temporales, así como las correlaciones espaciales. Hasta el momento no había existido una teoría de los procesos espacio temporales separado de la ya bien establecida teoría de la estadística espacial y el análisis de las series de tiempo. Sin embargo, con el auge de los datos espacio-temporales en los últimos años se ha acrecentado el estudio y la investigación de los datos espacio-temporales unido al estudio de las series de tiempo (Durango 2013).

Diversos estudios permitieron incrementar las capacidades y desarrollar herramientas que contemplan el análisis de variables temporales. Estos estudios se fundamentan en el hecho de que los datos espaciales son una representación del mundo real y que la representación implica la incorporación de modelos dinámicos, dependientes de las variables espaciales y temporales. El tiempo es una dimensión fundamental para entender y modelar la evolución de los fenómenos geográficos, pues se considera que los fenómenos y actividades espacio-temporales dependen, directamente, de la transformación del espacio geográfico (Durango 2013).

Algunos estudios examinan el espacio y el tiempo, asumiendo que cada elemento geográfico desempeña múltiples roles en un momento específico en el tiempo; esto condujo a sugerir que no existe un área en el espacio geográfico que se pueda separar del flujo del tiempo y el espacio. Una entidad geográfica tiene una ruta espacio-temporal, que inicia en el momento de su nacimiento o creación y termina en el momento en que muere o se destruye. Durante diferentes etapas del ciclo de vida sufre una serie de cambios en su localización y en los eventos que la afectan en el tiempo. Por lo tanto, el tiempo y el espacio son inseparables. Las entidades geográficas presentan una ruta espacio-temporal, que inicia en el momento de la toma de los *geo-datos* y termina en el momento que se destruyen los *geo-datos*. Bajo la anterior descripción, los datos espacio-temporales se representan como una inserción de la dimensión tiempo en entidades geográficas concebidas, donde el espacio geográfico se organiza en capas temáticas que incluyen la información de captura en un tiempo determinado (Durango 2013).

Los datos espacio-temporales se almacenan en diferentes formatos digitales. Algunos ejemplos de los formatos que más se usan son: *NetCDF* (*Networked Common Data Form*), *HDF* (*Hierarchical Data Format*). Diferentes instituciones, dependiendo de su uso y necesidades, generaron estos formatos.

1.2 Formato de datos espacio temporales

Existen diversos formatos de datos espacio-temporales que son usados por diferentes comunidades e instituciones científicas para almacenar e intercambiar grandes volúmenes de información (McGrath 2003). Pero estos formatos tienen diferentes niveles de uso de acuerdo a las comunidades SIG que los posean y al dominio que estas comunidades tengan sobre un formato en específico. En este epígrafe se analizan algunos de estos formatos con el objetivo de conocer sus principales características y algunas de sus aplicaciones.

1.2.1 Hierarchical Data Format (HDF)

Hierarchical Data Format, más conocido por ***HDF,*** fue desarrollado por el Centro Nacional de Aplicaciones de Supercómputo (*National Center for Supercomputing Applications*, NCSA) en el año 1988. En la actualidad, su soporte corre a cargo de *HDF Group* de la universidad de Illinois. Es un formato de datos de propósito general, flexible, portable y eficiente para el almacenamiento y recuperación de datos científicos (Ullman & Denning 2012; Poinot 2010).

Dado su excelente desempeño y simplicidad ha sido ampliamente usada por muchas comunidades de científicos (Long et al. 2013). Este formato es muy usado por entidades que producen y gestionan información de carácter ambiental y otras entidades de observación territorial, como es el caso de la NASA (Durango 2013). *HDF* se encuentra disponible de forma libre. La distribución consiste en la biblioteca, utilidades de línea de comando, una suite de prueba, interfaces con java y *HDFView* un visor basado en java, mediante el cual los usuarios pueden observar fácilmente detalles interesantes de los datos. *HDF* presenta cuatro niveles de interacción. En su nivel más bajo es un archivo para el almacenamiento de datos científicos. En su nivel más alto es una colección de utilidades y aplicaciones para manipular, ver y analizar datos en los ficheros *HDF*, y en los niveles intermedios se encuentra una biblioteca de programas que provee *APIs* de alto nivel y una interfaz de datos de bajo nivel (Figura 1.1).

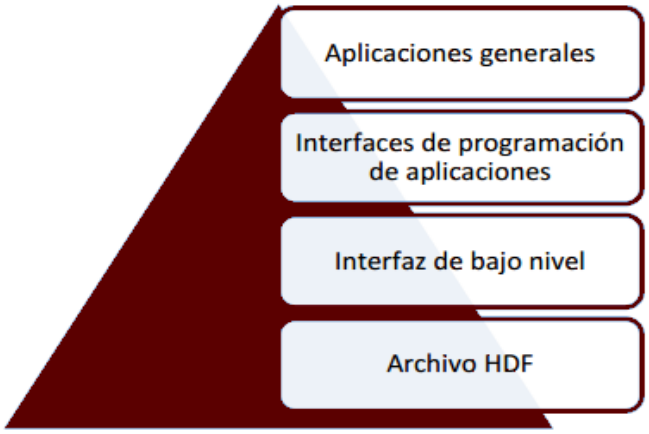

Figura 1.1 Niveles de interacción de *HDF*.

Aplicaciones generales

En el nivel más alto hay utilidades de línea de comandos de *HDF*, aplicaciones de NCSA que soportan visualización de datos y análisis, y variedad de aplicaciones de terceros desarrolladores.

Existen utilidades de línea de comandos de *HDF* para:

- Convertir de un formato a otro (por ejemplo, desde y hacia JPEG/*HDF*)

- Analizar y ver ficheros *HDF* (siendo *hdp* una de las herramientas más útiles)
- Manipular los ficheros *HDF*.

De las utilidades *HDF*, la "*hdp*" es una de las más importantes, su función está dada en que provee información rápida sobre los contenidos y objetos de datos en un archivo *HDF* (Long et al. 2013), puede listar los contenidos de los archivos *HDF* en varios niveles con diferentes detalles, además puede también vaciar los datos de uno o más archivos, en formato binario o ASCII.

Compresión de datos

HDF4 (y versiones posteriores) soportan una interfaz de compresión de bajo nivel, la cual permite que cualquier objeto de dato sea comprimido utilizando una variedad de algoritmos. Actualmente solo tres algoritmos de compresión están soportados: *Run-length Encoding* (RLE), *Huffman adaptativo*, y un codificador de diccionario *LZ-77* (el algoritmo de decodificación de gzip) (Yeh 2002). Los planes para algoritmos futuros incluyen un codificador de diccionario *Lempel/Ziv-78*, un codificador aritmético y un algoritmo rápido de Huffman. *HDF4* (y entregas posteriores) soportan compresión de n-bit para SDS, RLE (Run- Length Encoding), IMCOMP, y compresión JPEG para imágenes de puntos.

El archivo *HDF*

HDF puede almacenar varios tipos de objetos de datos dentro de un archivo, como imágenes de mapa de bits, paletas, texto y estilos de tablas de datos. Cada "objeto" en un archivo *HDF* tiene una etiqueta predefinida que indica el tipo de datos y un número de referencia que identifica la instancia. Hay un número de etiquetas disponibles para la definición que hace el usuario sobre los tipos de datos. Sin embargo, solamente las personas que tienen acceso al software del usuario pueden acceder adecuadamente a los datos. Los usuarios de *HDF* no necesitan conocer el formato físico, pues el único método práctico de acceso y manipulación de los datos es mediante interfaces de software. Existen diversas formas de acceder a los datos contenidos en un archivo *HDF*. Una de ellas es el acceso a nivel binario, conociendo la estructura del archivo como se

muestra en la Figura 1.2. A continuación, se describe la estructura básica del formato *HDF* (Group & others 2011):

- ***File Header***: Es el primer componente de un archivo *HDF* que incluye los primeros cuatro bytes en el archivo *HDF*. El *File Header* es una firma de los archivos *HDF*, con valores hexadecimales, por ejemplo, 0x0E, 0x03, 0x13, 0x01.
- ***RootGroup***: es una analogía de un directorio de sistema de archivos. Un grupo tiene cero o más objetos y cada objeto debe ser miembro de, al menos, un grupo. El grupo raíz es un caso especial, pues no puede ser miembro de ningún grupo. Contiene los siguientes campos: *Name* y *Attributes*. Los *Attributes* tienen asociados la siguiente información: *Name*; *Value*; *Type*, que puede tomar los siguientes valores: *String*, *byte* (8-bit), *short* (16-bit), *int* (32-bit), *unsigned byte* (8-bit), *unsigned byte* (16-bit), *unsigned byte* (32-bit), *long*(64-bit), *float*, *double*, *object reference*; tipo de almacenamiento: *Max string length* para tipos de datos *string* y *Array size* para todos los otros tipos de datos.
- ***Named Object***: Hay tres clases de *Named Objects*: *Group*, *Dataset* y *Named Datatype*. Cada uno de esos objetos es miembro de, al menos, un *Group* y un *Link.*
- ***Group***: similar al *Root Group*.
- ***Dataset***: es un arreglo multidimensional (rectangular) de *Data Elements*. El *Dataspace Object* define la forma de la matriz (número de dimensión y tamaño de cada dimensión). Se asocia con la siguiente información: *name* y *parent group*.
- ***Datatype***: describe las características de un único *Data Element*. Los *Datatype* son de dos tipos: *Atomic* y *Composite*. Un *Atomic Datatype* representa un objeto simple como: cadena de caracteres, carácter, tiempo, entero y doble, entre otros. Un *Composite Datatype* se compone de múltiples elementos de tipo *atomic datatype*, como: *array*, *enumeration*, *variable length* y *compound*. Se asocia con la siguiente información: *Datatype class*, *size*(*bits*) y *byte ordering*.
- ***Dataspace***: describe el diseño de los elementos de una matriz multidimensional. Un *Dataspace* describe el hiper-rectangulo como una lista de dimensiones con actuales y máximos (o ilimitado) tamaños. Se asocia con la dimensión y tamaño de la matriz.

- ***Attribute***: se usan para documentar los objetos. Los atributos de un objeto son el nombre y el dato. Un atributo es similar a un *Dataset*, pero con las siguientes limitaciones:

    - Sólo se puede acceder por medio del objeto.
    - Los nombres de los atributos tienen significado sólo sin el objeto.
    - Un atributo debe ser un objeto pequeño.
    - Un acceso simple debe leer y escribir los datos de un atributo.
    - Los atributos no tienen atributos.

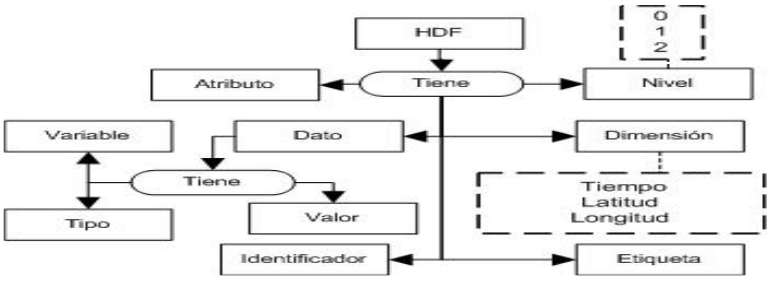

Figura 1.2 Formato de almacenamiento y transporte *HDF*. Fuente: (Soto-Durán et al. 2014)

1.2.2 HDF-EOS

Earth Observing System(EOS) es un programa iniciado por la *Earth Science Enterprise* (ESE) de la NASA para el estudio del cambio climático global. EOS usa principalmente instrumentos aerotransportados para obtener información sobre la tierra, la atmosfera, los océanos y los fenómenos físicos-químicos que ocurren sobre el sistema terrestre (Yang & Di 2004). En la actualidad el programa EOS genera más de 2 terabytes de datos por día. *HDF-EOS* es un formato estándar para los datos EOS producidos por la NASA. El formato *HDF* fue seleccionado como sistema base para almacenar los datos EOS, sin embargo, este carece de mecanismos para guardar información geo-localizada, que es vital para los datos geoespaciales. El proyecto EOS amplió *HDF* a *HDF-EOS* para añadirle tres nuevos modelos de datos (*point*, *swath* y *grid*). De los tres modelos de datos añadidos *swath* y *grid* pueden tener dos o múltiples dimensiones espaciales mientras que *point* es un modelo de dato discreto. El modelo *grid* es usado para datos geo-rectificados, donde los elementos de datos espaciales (pixeles) son regularmente organizados y todos los pixeles tiene la misma forma y tamaño. Las coordenadas espaciales de

cualquier pixel pueden ser obtenidas fácilmente, dando las coordenadas de un pixel de referencia, usualmente la esquina superior o inferior izquierda y el tamaño del pixel. Los datos *grid* pueden ser fácilmente añadidos a otros conjuntos de datos espaciales para ser analizados por Sistemas de Información Geográfica y sensores remotos (Yang & Di 2004) . El modelo de datos *swath* es para datos que van a ser geo-rectificados, donde diferentes pixeles tienen diferentes formas y tamaños. Sin embargo, sin ser geo-rectificados los datos *swath* no pueden ser combinados directamente con otros datos espaciales geo-rectificados. Actualmente muchos SIG y sensores remotos no son capaces de procesar los *HDF-EOS* con modelos de datos *swath* debido a no poder geo-rectificar estos datos. Esto sin duda ha significado una limitante para muchos usuarios de EOS, por lo que se están realizando estudios y trabajos con el objetivo de crear herramientas que ayuden a los usuarios a geo-rectificar sus datos.

1.2.3 Network Common Data Form (*NetCDF*)

El formato de archivo *NetCDF* (*Network Common Data Form*) fue implementado por el programa *Unidata*, uno de los ocho programas de la UCAR, y establecido como formato estándar para la comunidad científica para el almacenamiento de todo tipo de datos oceanográficos y atmosféricos desde el año 1989 (Borrell González 2012). La fortaleza de *NetCDF* está dada por la sencillez de su modelo de datos subyacente, su flexibilidad y su eficiente acceso a los datos (Hankin et al. 2010). *NetCDF* contiene información que ayuda a identificar qué clase de datos se encuentra en el archivo, como tipo de variable, unidad, dimensión y origen, entre otros. *NetCDF*, a diferencia de otros formatos, no requiere archivos adicionales para su interpretación. En relación con sus características, los datos *NetCDF* pueden ser (UNIDATA 2017):

- Auto-descriptivos: contienen información acerca de los datos.
- Portables: un archivo *NetCDF* se puede acceder de diferentes maneras.
- Escalables: se puede acceder a un pequeño conjunto de un gran conjunto de datos eficientemente.
- Agregables: se pueden añadir datos a estructuras existentes sin copiar ni redefinir todo el conjunto de datos.

- Compartibles: un proceso escritor y varios lectores pueden acceder a los datos de un *NetCDF* simultáneamente.
- Archivables: soporta el acceso a versiones antiguas de *NetCDF*.

Estándares que cumple *NetCDF*

- *OGC* (*Open Geospatial Consortium*) persigue acuerdos entre las diferentes empresas del sector que posibiliten la interoperación de sus sistemas de geo-procesamiento y facilitar el intercambio de la información geográfica en beneficio de los usuarios.
- *CDI* (*Common Data Index*) su objetivo es dar a los usuarios una visualización muy detallada y gran difusión geográfica de información oceanográfica a través de los datos obtenidos de diferentes instituciones.
- *ISO* 19115 (*International Organization for Standardization*), es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación. Su función principal es buscar la estandarización de normas de productos para organizaciones a nivel internacional. *NetCDF* tiene una interfaz de programación de aplicaciones (API) bien diseñada que permite el desarrollo de aplicaciones en varios lenguajes de programación. Esta interfaz es usada también para una biblioteca de funciones de acceso a datos, que se almacenan y recuperan en forma de matrices (Rew et al. 2010). Los valores de las matrices se pueden acceder directamente, sin conocer detalles del almacenamiento de los datos. El desarrollo del formato *NetCDF* mejora la accesibilidad y la reutilización de los datos en las matrices. Recientemente los desarrolladores de *NetCDF* y *HDF* han colaborado para crear un software que combina las mejores características de cada uno, permitiendo así, una mayor interoperabilidad entre las dos comunidades(Hankin et al. 2010). El software resultante fue *NetCDF-4* compatible con versiones anteriores de programas y datos para *NetCDF-3*, pero mejora el rendimiento e incrementa las capacidades de codificación de colecciones de datos.

El archivo *NetCDF*

Los archivos *NetCDF* tienen una estructura interna definida por el *CDM* (*Common Data Model*). El *CDM* es un modelo de datos abstractos para conjuntos de datos científicos, este modelo

fusiona los modelos *NetCDF*, *OPeNDAP* y *HDF5* para crear una API común para muchos tipos de datos científicos. El *CDM* consta de varias capas, donde cada una se va superponiendo en la parte superior de la anterior, para ir añadiendo cada vez una semántica más compleja:

- La capa de **acceso de datos**, se encarga de los datos de lectura y escritura.
- La capa de **sistemas de coordenadas**, identifica las coordenadas de las matrices de datos.
- La capa de **atributo estándar** es la que conoce algunos de los significados que los humanos utilizamos para hacer referencia a los datos científicos tales como unidades, sistemas de coordenadas, topología de datos, etc.…
- La capa de **características de tipo de dato científico**, identifica los tipos de datos añadiendo métodos especializados para cada tipo de datos.

Según (Rew et al. 2010) los tipos de datos abstractos más importantes que componen un archivo *NetCDF* son:

- **Conjunto de Datos (*Dataset*)**: Puede ser un *NetCDF*, *HDF5*, *OPeNDAP* o cualquier otro tipo de documento al que se pueda acceder a través de la API de *NetCDF*.
- **Grupo**: Un grupo contiene variables, dimensiones, atributos, etc.… En definitiva, los grupos son el contenedor de todo aquello que contiene nuestro *dataset* formando un árbol jerárquico. Por eso, como mínimo siempre habrá un grupo definido en nuestro archivo.
- ***ProtoVariable**:* En los archivos *NetCDF* una *ProtoVariable e*s un contenedor de datos. Está compuesta por varias variables que contienen su información, como por ejemplo el tipo de datos que contiene, su dimensión o dimensiones que definen su matriz y, opcionalmente, se pueden complementar añadiéndole un conjunto de atributos que la describan o definan aun con mayor exactitud.
- **Dimensión**: Una dimensión se utiliza para definir en función de qué van a variar los valores de nuestra variable. Puede ser compartida entre las demás variables lo que proporciona una manera simple y eficaz de asociarlas.
- **Atributo**: Un atributo tiene un nombre y un valor que se utilizan para asociar metadatos con una variable o grupo. Por ejemplo, si se trata de un atributo de una variable

estaríamos hablando de unidades o nombre estándar y solamente afectaría a esta variable. Si nos referimos a un atributo de un grupo, este estaría afectando a todo el conjunto de datos y el atributo sería, por ejemplo, para definir qué convención se ha utilizado o el nombre de la organización que ha generado estos datos.

- **Estructura**: Es un tipo de variable que contiene otras variables. De esta manera los datos almacenados en una estructura se encuentran físicamente muy juntos en el disco, lo que hace que sea eficaz la recuperación de estos datos al mismo tiempo.
- ***Array***: Un *array* (o matriz) contiene los datos reales de una variable obtenidos del disco, red, base de datos, etc. Podemos decir que un *array*es el contenedor de las series de datos.

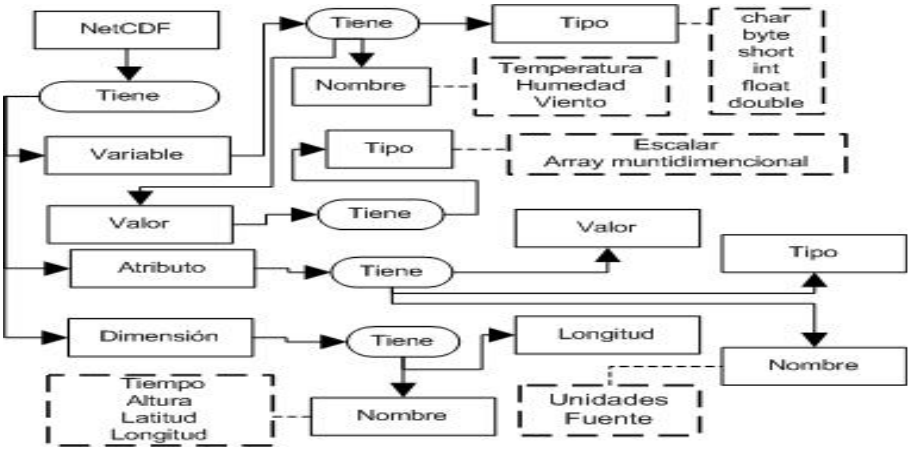

Figura 1.3 Formato de almacenamiento y transporte *NetCDF*. Fuente: (Soto-Durán et al. 2014)

1.2.4 Common Data Format (CDF)

CDF es un formato de datos para almacenar conjuntos de datos multidimensionales. Sus orígenes datan del desarrollo del Sistema de Datos Climáticos de la NASA en el *National Space Science Data Center* (NSSDC). Según (NASA 2013) una de las características más importante de *CDF* es que puede manipular conjuntos de datos inherentemente dimensionales además de los escalares. Para lograr esto, *CDF* agrupa los datos por variables cuyos valores son organizados conceptualmente en arreglos. Las variables *CDF* son nombres genéricos u objetos que representan datos. Por ejemplo, una variable puede ser datos representando una variable independiente, una variable dependiente, tiempo, un valor de fecha, etc. En otras palabras, las variables no contienen ningún significado aparte de los datos en sí mismos. La dimensionalidad

de una variable depende de cómo sean especificados los datos por los usuarios. Para un dato escalar por ejemplo el arreglo de valores debe ser 0-dimensional; mientras que para los datos de una imagen seria de 2 dimensiones. *CDF* permite a los usuarios especificar arreglos hasta de diez dimensiones. Los arreglos para una variable en particular son llamados "*variable record*". Una colección de arreglos, uno para cada variable es nombrada como un "*CDF record*". Un *CDF* puede contener múltiples "*CDF records*". Estos son útiles para los datos que son observados en diferentes instantes de tiempo. Mientras que las variables son un mecanismo para representar los datos, los atributos en un *CDF* son un mecanismo para describir el archivo *CDF* y sus variables. Existen dos tipos de atributos en un *CDF*: los atributos globales para describir el archivo *CDF* y los atributos de variables para describir cada variable. Ejemplos de atributos globales pueden ser: fecha de creación del archivo, autor, documentación de los datos, etc. Ejemplos de atributos de variables son: valor máximo y mínimo, unidades con la cual son almacenados los datos, etc. Es importante destacar que no existe una solo forma correcta de almacenar los datos en un *CDF*, dado que los usuarios tienen total control de como guardar los datos. Esta es una de las ventajas de *CDF*. Los datos pueden ser organizados de cualquier forma según crea el usuario. *CDF* es también una biblioteca flexible y extensible que brinda muchas facilidades para la creación y el acceso a un *CDF* (NASA 2013). Esta permite crear de dos formas los formatos de archivos para almacenar los datos y metadatos. La primer es el formato tradicional de multi-archivos *CDF*, de esta manera se crea un archivo "*.CDF*" con toda la información y metadatos, además un archivo para cada variable .v#, donde # es el número de la variable. La segunda opción es crear un único archivo ".cdf" que contiene toda la información, metadatos y los valores de los datos para cada variable en el *CDF*. Ambos formatos permiten el acceso directo a los datos. La ventaja de un único archivo es que minimiza el número de archivos a manejar y facilita su distribución a través de redes. Esta biblioteca también permite la compresión de datos, pero solo para un *CDF* creado a partir de un único archivo.

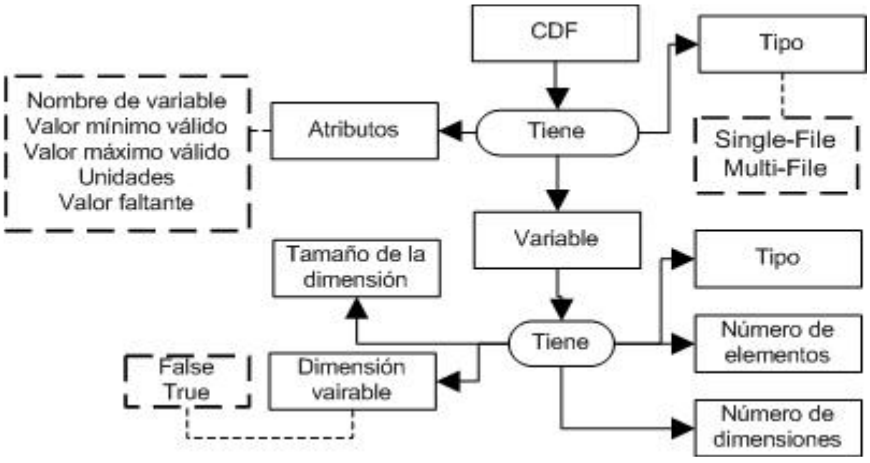

Figura 1.4 Formato de almacenamiento y transporte CDF Fuente: (Soto-Durán et al. 2014)

1.2.5 Generalidades de los formatos

Los formatos *CDF*, *HDF* y *NetCDF* están pensados para almacenar y transportar paralelepípedos de datos que contienen múltiples dimensiones, incluidos el tiempo y las coordenadas espaciales. De esta manera un corte transversal sobre los datos permite obtener un modelo *raster*, mientras que un corte longitudinal logra series de datos. La mayor fortaleza de estos tres formatos es la posibilidad de describir la información que contienen en términos de unidades de medida, fuente de obtención y todos los metadatos o atributos que el usuario requiera almacenar. Esto hace que los datos almacenados en *CDF*, *HDF* y *NetCDF* sean fáciles de interpretar (Soto-Durán et al. 2014).

Los formatos de datos científicos no son adecuados para almacenarlos en sistemas de bases de datos de propósito general pues estos no soportan objetos multidimensionales (*arrays*) o jerárquicos como unidad básica para el acceso a los datos. Se necesita un modelo de datos diferente para que se facilite la recuperación, modificación, manipulación matemática y visualización de estos datos (Ravishankar 2008).

Por lo tanto, estas tecnologías (*HDF*, *CDF*, *NetCDF*) han evolucionado para cumplir con los siguientes desafíos de almacenamiento de datos:

• Los datos son grandes

• Los datos son complejos

• Los datos son heterogéneos

• Los datos son esotéricos

• Las necesidades de acceso incluyen E / S paralelas

• Las necesidades de acceso incluyen acceso aleatorio

Después de conocer algunas de las principales características y aplicaciones de los formatos de datos espacio-temporales pasaremos a conocer algunos de los formatos de datos geoespaciales.

1.3 Formatos de datos geoespaciales

Los formatos de datos geoespaciales se crean, comparten y almacenan en muchos formatos diferentes. Los datos geoespaciales enlazan información alfanumérica con una localización específica. La información alfanumérica se muestra en la pantalla del ordenador acorde a la localización de los objetos. Esto es a lo que se le llama modelo de datos *"es un conjunto construido para la descripción y representación del aspecto de los objetos del mundo real en el ordenador"* (Longley, P., Maguire, D., Goodchild, M., Rhind 2005).

Los modelos de datos básicos de un SIG son el Modelo *Vectorial* y el Modelo *Raster*. La información espacial se muestra en un SIG utilizando capas que contienen esta información. Cada capa se puede basar tanto en un modelo de datos *vectorial* como *raster*. En este epígrafe se analizan estos modelos con el objetivo de conocer sus principales características y algunas de sus aplicaciones.

1.3.1 Modelo *Raster*

En el modelo *raster*, la zona de estudio se divide de forma sistemática en una serie de unidades mínimas o comúnmente denominadas celdas. Para cada celda se recoge la información adecuada que la describe. Un modelo de datos *raster* principalmente se usa para representar variables continuas en el espacio tales como la temperatura, elevación o imágenes tomadas por los satélites espaciales. Se puede apreciar con más detalles en la Figura 1.5 donde se muestra en una malla *raster* los diferentes objetos representados en ella (Olaya 2014).

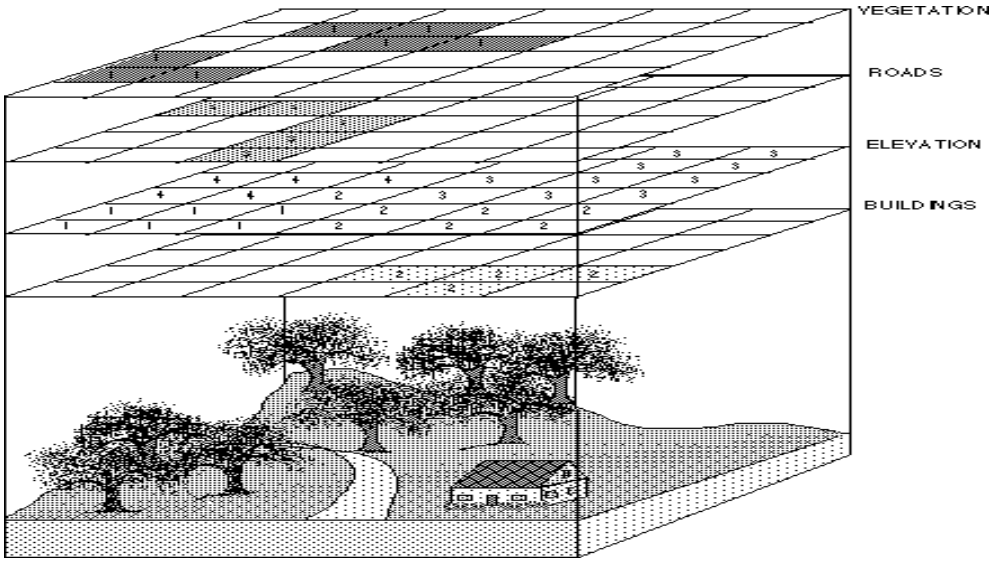

Figura 1.5 Representación de un paisaje mediante un modelo de datos *raster*. Fuente: (Chang 2008)

Aunque la malla de celdas puede contener información sobre varias variables, lo habitual es que trate una única variable. Es decir, que se tenga un único valor para cada una de las celdas.

La principal característica del modelo *raster* es su sistematicidad. La división del espacio en celdas se realiza de forma sistemática siguiendo determinado patrón, de tal modo que existe una relación implícita entre las celdas, pues estas son contiguas entre sí, cubren todo el espacio, y no se solapan. Por tanto, la posición de una celda depende de la posición de las restantes, para así conformar en conjunto toda la malla regular que cumple las anteriores características (Olaya 2014).

Los principales elementos que definen una malla *raster* de celdas cuadradas son:

- Una localización geográfica exacta de alguna celda y una distancia entre celdas, para en base a ellas, y en virtud de la regularidad de la malla, conocer las coordenadas de las restantes.
- Un conjunto de valores correspondientes a las celdas.

En el modelo *raster* no se recogen de forma explícita las coordenadas de cada una de las celdas, sino los valores de estas. No resulta necesario acompañar a dichos valores de un posicionamiento espacial concreto, pues hacen referencia a un elemento particular de la malla, la cual representa una estructura fija y regular. No obstante, se hace necesario posicionar dicha

malla en el espacio para después poder calcular las coordenadas particulares de cada celda (Olaya 2014).

Generalmente lo más común es definir la posición de una única celda (habitualmente la celda superior izquierda), una orientación fija, y una distancia entre celdas. Como se muestra en la Figura 1.6 mediante un sencillo cálculo se puede conocer las coordenadas de todas las celdas sin necesidad de almacenar estas (Olaya 2014).

La orientación de las capas *raster* habitualmente es Norte-Sur, de forma tal que al pasar de la primera a la segunda fila se desciende en latitud. Para entenderlo mejor, la parte superior de la imagen es el norte, mientras que la inferior es el sur. Esta convención simplifica el trabajo con capas *raster* dentro de un SIG y permite aplicar directamente la formula mostrada en la Figura 1.6 (Olaya 2014).

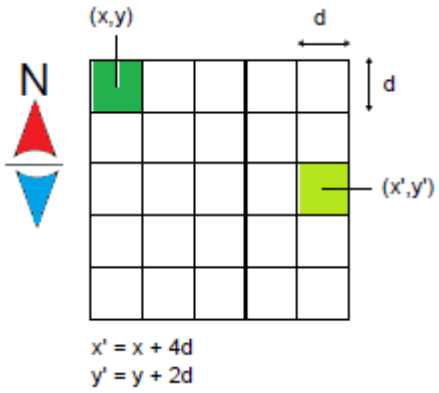

Figura 1.6 La estructura regular de la malla *raster*.

El otro parámetro necesario junto con la orientación de la malla y la situación geográfica de una de sus celdas es el denominado tamaño de celda o tamaño de píxel, también conocido como resolución, pues, en efecto, su magnitud define la resolución de la capa. Un tamaño de celda mayor implica una menor resolución, y viceversa. Además de servir para el cálculo de coordenadas de las celdas y definir la estructura de la malla, el tamaño de celda permite calcular áreas, ya que establece el área ocupada por cada celda. El aspecto más relevante es que el tamaño de celda determina la precisión con la que se recoge una variable dentro de una capa *raster*, que se considera como el equivalente conceptual a la escala de dicha capa. Por esta razón, es importante trabajar con capas *raster* de un tamaño de celda adecuado para el tipo de análisis o tarea que quiera desarrollarse (Olaya 2014).

Imágenes como capas *raster*

Un caso particular de capas *raster* son las imágenes, donde el concepto de celda en una malla *raster* es equivalente al de pixel (acrónimo de *picture element*) en las imágenes digitales. Una de las particularidades de las imágenes es la presencia de *bandas*. Los valores recogidos en las imágenes indican la reflectancia en una determinada longitud de onda. El espectro de radiación puede subdividirse en diferentes grupos por lo que, los sensores que toman estas imágenes recogen varias capas, una para cada uno de estos grupos. En lugar de almacenarse como un conjunto de capas separadas, es más usual que se almacenen en una única capa que contiene varias *bandas*, es decir, varios niveles distintos, donde cada uno podría constituir de por sí mismo una capa *raster* (Olaya 2014).

1.3.2 Modelo *Vectorial*

En el modelo *vectorial*, no existen unidades fundamentales que dividen la zona recogida, sino que se recoge la variabilidad y características de esta mediante entidades geométricas, para cada una de las cuales dichas características son constantes. Este modeliza el espacio geográfico mediante una serie de primitivas geométricas que contienen los elementos más destacados de dicho espacio. Estas primitivas son de tres tipos: puntos, líneas y polígonos (Figura 1.7) (Olaya 2014).

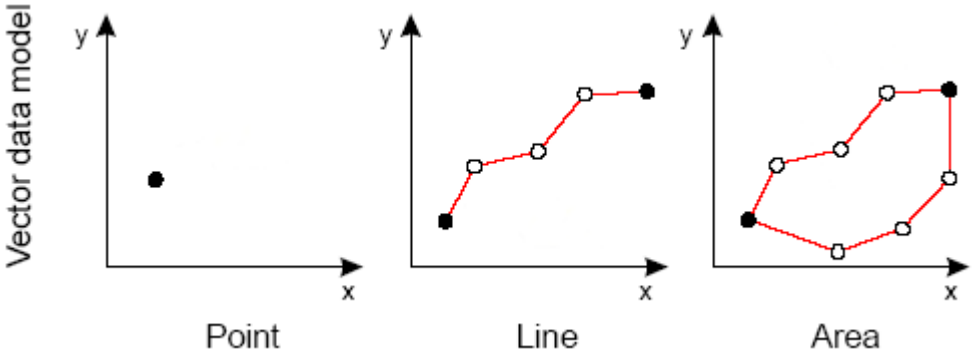

Figura 1.7 Primitivas geométricas en el modelo de representación *vectorial*.

Utilizando puntos, líneas o polígonos, puede modelarse el espacio geográfico si se asocia a estas geometrías una serie de valores definitorios. Al definir las formas geométricas básicas, todas ellas pueden en última instancia reducirse a puntos. Así, las líneas son un conjunto de puntos interconectados en un determinado orden, y los polígonos son líneas cerradas, también

expresables como una serie de puntos. Todo elemento del espacio geográfico queda definido por una serie de puntos que determinan sus propiedades espaciales y una serie de valores asociados. Una única entidad puede contener varias primitivas, esto permite representar territorios en los que no se puede mostrar toda su extensión mediante una sola primitiva. Una entidad que esté compuesta por más de una primitiva, existe un único conjunto de valores asociados a ella (Olaya 2014).

El modelo de representación *vectorial* posee como elemento particular la ***topología***. De manera informal, la topología se ocupa de aquellas propiedades y características de las figuras o cuerpos geométricos que permanecen invariantes, cuando dichas figuras son plegadas, dilatadas, contraídas o deformadas, de manera que no aparezcan nuevos puntos o se hagan coincidir puntos diferentes. La trasformación permitida presupone, en otras palabras, que hay una correspondencia biunívoca entre los puntos de la figura original y los de la transformada, y que la deformación hace corresponder puntos próximos a puntos próximos. Esta última propiedad se conoce como continuidad, donde se requiere que la transformación y su inversa sean continuas: lo que permite el trabajo con homeomorfismos (Stadler 2002).

La topología en el ámbito de los SIG se entiende desde un punto de vista menos estricto y más funcional. En general, una capa de información presenta topología si en ella se almacenan las relaciones entre los distintos elementos que la componen. De no ser así, la capa es puramente cartográfica, pues los elementos que contiene no presentan relación entre sí, o esta relación no está almacenada junto a la propia información de estos elementos (Olaya 2014).

La representación de puntos o líneas es inmediata, sin embargo, al representar polígonos aparecen dos situaciones diferentes (Figura 1.8):

- Si los polígonos aparecen aislados los unos de los otros, cada polígono se codifica como una línea cerrada, esta representación se denomina modelo *Orientado a Objeto*.

  Esta representación tiene como desventaja que, si los polígonos se yuxtaponen, codificar los polígonos como líneas cerradas tiene el problema de que habría que repetir cada una de las líneas interiores.

- El formato alternativo es el modelo Arco-Nodo donde se codifican las líneas por separado y posteriormente, se define cada uno de los polígonos a partir del conjunto de líneas que lo componen. Así en la Figura 1.8 el polígono B se codifica como la unión de los arcos 1,2 y 3. La codificación de polígonos con este modelo requiere por tanto de dos etapas:
    - Digitalización, durante la que se introducen lo arcos.
    - Reconstrucción de la topología, donde se definen los polígonos y se crea la tabla que relaciona polígonos con arcos. La reconstrucción de la topología exige que la disposición de los arcos sea topológicamente correcta, así en la Figura 1.8 en el polígono B los nodos iniciales y finales de los tres arcos deben coincidir exactamente.

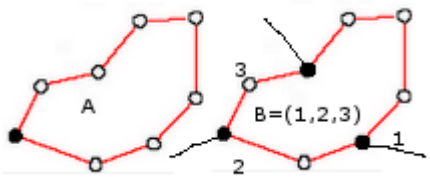

Figura 1.8 a)Polígono en formato Orientado a Objetos b) Polígono en formato Arco-Nodo.

La mayor virtud del modelo Arco-Nodo es ahorrar memoria, facilitar la digitalización y algunas de las operaciones de análisis SIG, siendo hoy día el modelo más utilizado (Sarria 2006).

Generalmente se considera al formato *vectorial* más adecuado para la representación de entidades o variables cualitativas, sin embargo, esto no es necesariamente así.

Para representar variables cualitativas y objetos en el modelo *vectorial* existen dos formas posibles (Figura 1.9):

- Formato *vectorial Arco-Nodo*
- Formato *vectorial Orientado a Objetos*, menos adecuado pues se introduce mucha información redundante.

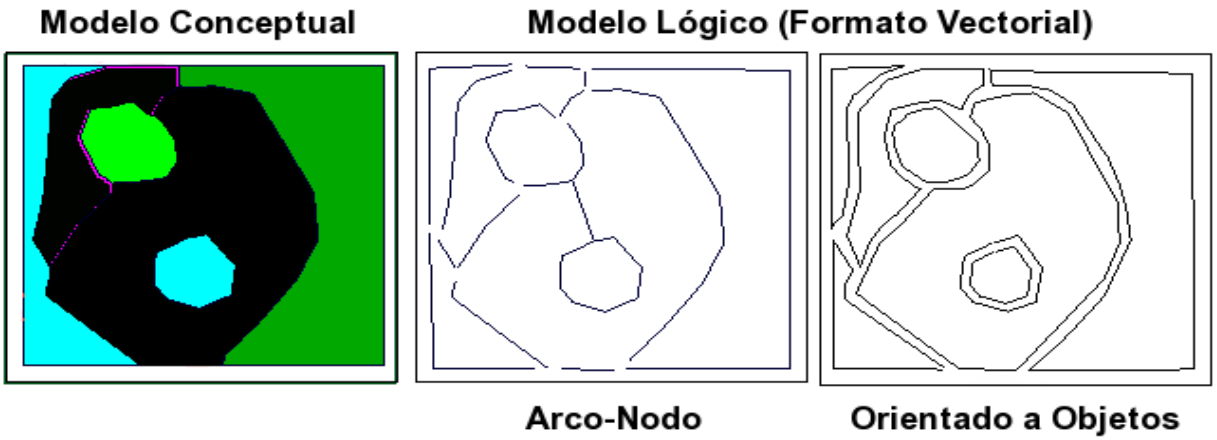

Figura 1.9 Modelos lógicos para representar variables cualitativas.

El modelo *vectorial* representa la superficie de tres formas posibles (Figura 1.10):

- **Malla regular de puntos**, a cada uno de ellos se asigna el valor de la variable medido en el punto.
- **TIN (Red Irregular de Triángulos)**, los puntos se concentran en aquellas zonas donde la variable representada tiene mayor variabilidad.
- **Isolíneas,** líneas en las que el identificador se sustituye por el valor de la variable**.**

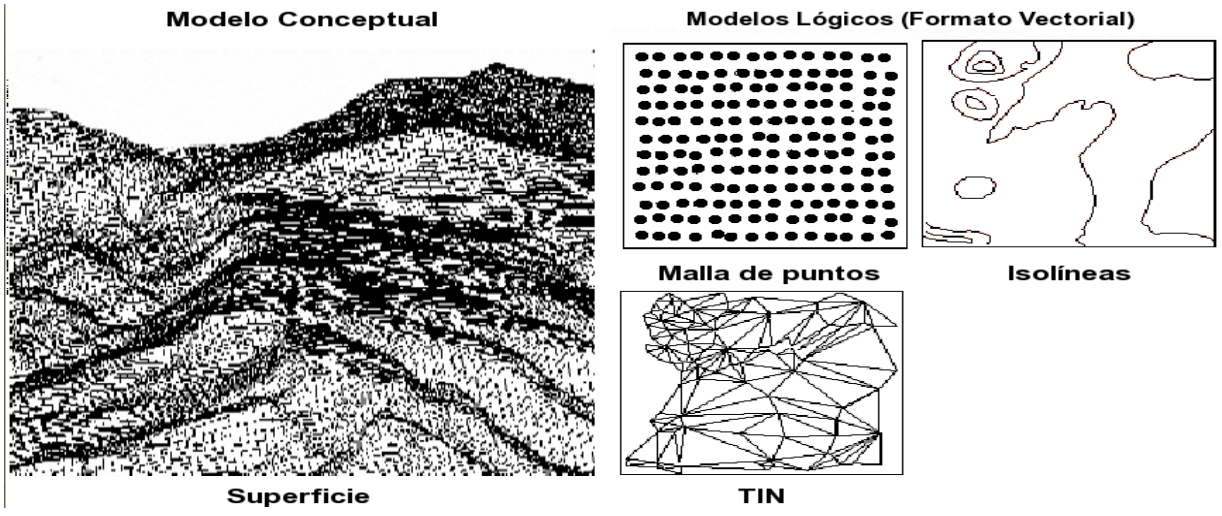

Figura 1.10 Modelos Lógicos para la representación de superficies en formato *Vectorial.*

Estos modelos lógicos de representar la superficie en el formato *vectorial* tienen como problema fundamental que no representan a la totalidad del espacio, por tanto, requieren una interpolación compleja para saber cuál es el valor de un punto en concreto.

1.3.3 Ventajas y desventajas del modelo *vectorial* y *raster*

Es obvio que las diferencias entre los modelos *vectorial* y *raster* son muy notables, y que cada uno de ellos ostenta de ventajas y desventajas. La decisión de un modelo u otro debe basarse en el tipo de estudio o enfoque que se pretenda realizar, así como del software y fuentes de datos disponibles.

Las superficies se representan más eficientemente en formato *raster* y solo pueden ser representadas en el formato *vectorial* mediante los modelos híbridos (mallas de puntos, TIN e Isolíneas) que no resultan convenientes para la realización de posteriores análisis pues las operaciones que permite el modelo *raster* resultan más lentas con el modelo *vectorial*.

Habitualmente resulta más eficiente la representación de los objetos utilizando el formato *vectorial* pues ocupa menos espacio en disco duro (aunque hoy en día este problema se puede suplir con varios sistemas de compresión, además de la cada vez mayor capacidad que presentan los discos duros) y permiten mayor rapidez para la visualización. Sin embargo, el formato *vectorial* es más lento que el *raster* para la utilización de herramientas de análisis espacial y consultar acerca de posiciones geográficas concretas (Sarria 2006).

En la Tabla 1.1 se resumen las ventajas e inconvenientes para el uso de los modelos *raster* y *vectorial* en el almacenamiento de datos espaciales:

Modelo *Raster*	Modelo *Vectorial*
Ventajas: 1. Estructura de datos sencilla 2. Eficiente para datos de detección remota o escaneados 3. Procedimientos sencillos de análisis espacial	Ventajas: 1. Los datos pueden representarse en su resolución original sin generalización 2. Requiere menos espacio de almacenamiento en disco 3. Las relaciones topológicas se mantienen fácilmente 4. La salida gráfica se asemeja más a los mapas dibujados a mano
Desventajas: 1. Requiere mayor espacio de almacenamiento en la computadora	Desventajas: 1. Estructura de datos más compleja 2. Ineficiente para datos de detección remota

2. Dependiendo del tamaño del pixel, la salida gráfica puede ser menos agradable	3. Algunos procedimientos de análisis espacial resultan procesos intensivos y muy complejos
3. Las transformaciones de proyecciones son más difíciles	4. La superposición de mapas vectoriales múltiples suele consumir mucho tiempo.
4. Más difícil la representación de relaciones topológicas	

Tabla 1.1 Comparación entre el modelo *raster* y *vectorial*

En estos epígrafes se ha abordado sobre las principales características de los formatos de datos espacio-temporales y geoespaciales, su definición y algunas de sus aplicaciones. Después de abordar estos temas es válida la pregunta ¿Por qué conviene convertir de un formato a otro?

1.4 Conveniencia de la conversión de formatos científicos para series de datos espacio-temporales y geoespaciales

La conversión de formatos científicos para el almacenamiento, transporte y visualización de datos geográficos es necesaria para lograr cierto nivel de interoperabilidad entre las aplicaciones o sistemas de información geográfica que manipulan estos tipos de formatos. Transformar un tipo de formato a otro puede ser útil en casos en los que se cuente con herramientas que solo permiten la manipulación de un tipo de formato específico.

El intercambio universal de los datos geoespaciales constituye uno de los temas más debatidos actualmente en el mundo de los SIG. Los usuarios disponen de servicios web como WMS (*Web Map Service*), WCS (*Web Coverage Service*), y WFS (*Web Feature Service*), que facilitan la interoperabilidad entre los datos geoespaciales.

Demandas de la interoperabilidad

Los usuarios esperan que el intercambio de datos reúna las siguientes características:

- **Sencillo:** no debe ser necesario que los usuarios entiendan mucho sobre los datos o su sistema fuente para importarlos y utilizarlos.
- **Transparente:** las complejidades asociadas con la transferencia de datos deben estar ocultas.
- **Abierto:** la interoperabilidad debe poder aplicarse a todos los sistemas, y el intercambio de datos ser independiente de la tecnología utilizada.

- **Efectivo**: la transferencia de datos debe ser fiable y los datos resultantes útiles para el fin perseguido.
- **Universal:** todas las bases de datos geoespaciales deben ser accesibles.

Todo esto no es fácil de conseguir. Una solución consistiría en una arquitectura única y un conjunto de estándares para datos geoespaciales. Sin embargo, parece imposible concebir que la comunidad global SIG adopte una única arquitectura geoespacial o estándares de datos a nivel global. Existen por lo menos media docena de estándares importantes, además de los productos de datos de propiedad comercial que ya están en el mercado. Esto significa que los esfuerzos de estandarización, no producen la interoperabilidad por sí solos. La interoperabilidad requiere consistencia a través de una amplia gama de parámetros técnicos, semánticos e institucionales (Levinsohn 2001).

Sin embargo, el alcance de este trabajo no es hacer una herramienta completamente interoperable entre los formatos de datos científicos y geográficos sino facilitar la transformación de datos entre estos formatos. Las principales ventajas de este trabajo son:

- Reunir en una sola herramienta la posibilidad de trasformar diversos tipos de formatos de datos científicos y geográficos.
- Facilitar el trabajo con formatos de datos científicos y geográficos.

Resulta conveniente seleccionar un lenguaje que se ajuste adecuadamente a las necesidades de la herramienta que se desea realizar y de los formatos científicos de datos espacio-temporales. A continuación, se aborda sobre el lenguaje estadístico y de programación *R.*

1.5 Lenguaje de Programación R

R es un lenguaje de programación interpretado, de distribución libre, bajo Licencia GNU, y se mantiene en un ambiente para el cómputo estadístico y gráfico. Este software corre en distintas plataformas Linux, Windows, MacOS, e incluso en PlayStation 3. El término ambiente pretende caracterizarlo como un sistema totalmente planificado y coherente, en lugar de una acumulación gradual de herramientas muy específicas y poco flexibles, como suele ser con otro software de análisis de datos (Santana & Farfán 2014). El hecho que R sea un lenguaje y un sistema, es

porque forma parte de la filosofía de creación 1, como lo explica John Chambers (Chambers & Hastie 1991), cito: *"Buscamos que los usuarios puedan iniciar en un entorno interactivo, en el que no se vean, conscientemente, a ellos mismos como programadores. Conforme sus necesidades sean más claras y su complejidad se incremente, deberían gradualmente poder profundizar en la programación, es cuando los aspectos del lenguaje y el sistema se vuelven más importantes."*

1.5.1 Historia de R

R fue creado en 1992 en Nueva Zelanda por Ross Ihaka y Robert Gentleman (Ihaka 1998). La intención inicial con R, era hacer un lenguaje didáctico, para ser utilizado en el curso de Introducción a la Estadística de la Universidad de Nueva Zelanda. Para ello decidieron adoptar la sintaxis del lenguaje S desarrollado por *Bell Laboratories*. Como consecuencia, la sintaxis es similar al lenguaje S, pero la semántica, que aparentemente es parecida a la de S, en realidad, es sensiblemente diferente, sobre todo en los detalles un poco más profundos de la programación (Santana & Farfán 2014).

A modo de broma Ross y Robert, comienzan a llamar "R" al lenguaje que implementaron, por las iniciales de sus nombres, y desde entonces así se le conoce en la muy extendida comunidad amante de dicho lenguaje.

1.5.2 Paquetes para la manipulación de formatos científicos de datos espacio-temporales y geoespaciales en R

R forma parte de un proyecto colaborativo y abierto que promueve la creación de paquetes con el fin de brindar la posibilidad a la comunidad de compartir entre los usuarios el trabajo y además contribuir con el crecimiento del propio lenguaje. Existe un repositorio oficial de paquetes que son organizados en vistas (o temas) que permiten agruparlos según su naturaleza y función. El lenguaje cuenta con paquetes para la manipulación de datos científicos y espacio-temporales como son:

1.5.2.1 Paquete *h5*

H5 proporciona una interfaz a la API *HDF5* a través de clases S4 para el almacenamiento eficiente de datos científicos en este formato. *HDF5* es un formato de datos binarios diseñado para la Entrada / Salida de forma flexible y eficiente de datos de gran volumen y complejidad.

Un archivo *HDF5* puede ser estructurado de forma jerárquica para almacenar conjuntos de datos (*datasets*) en grupos, muy similar a la estructura de carpetas en un sistema de archivos. Soporta almacenamiento rápido y recuperación de objetos R como *vectores*, *matrices* y *arrays* a archivos binarios en un formato independiente del idioma. Puesto que *h5* no es solo capaz de acceder a subconjuntos de datos almacenados, sino también puede manejar conjuntos de datos que estén en la memoria (Annau 2016).

1.5.2.2 Paquete *RNetCDF*

RNetCDF proporciona una interfaz para el formato de archivo *NetCDF* diseñado por *Unidata* para el almacenamiento eficiente de datos científicos en este formato. Este paquete es basado en la API C de la biblioteca *NetCDF*. La implementación actual soporta todas las operaciones sobre los conjuntos de datos (*datasets*) de los *NetCDF* en formatos de archivos clásicos y de 64 bits, y formato clásico *NetCDF4*, además nos permite la escritura de nuevos archivos con este formato y la lectura y modificación de archivos existentes (Michna 2016).

1.5.2.3 Paquete *Raster*

El paquete *Raster* dispone de clases y funciones para la manipulación, creación, lectura y escritura de datos geoespaciales en formato *raster*. El paquete también proporciona, entre otras cosas, funciones para la manipulación general de datos en formato *raster* que pueden ser usadas fácilmente para el desarrollo de funciones más específicas. Por ejemplo, hay funciones para leer porciones de los valores de un *raster* o para convertir números de celdas en coordenadas y viceversa. También implementa el álgebra *raster* y muchas otras funciones para la manipulación de datos *raster* que son muy comunes en Sistemas de Información Geográficas (SIG).

Una de las características más notables del paquete *Raster* es que puede trabajar con conjunto de datos *raster* almacenados en disco y que son muy grandes para cargarlos en memoria (RAM). Esta característica es posible porque los Objetos que crea solo contienen información sobre la estructura de los datos, como son el número de filas y columnas, la extensión espacial y el nombre del archivo, pero no intenta leer todos los valores de las celdas en memoria. Si no se especifica un fichero de salida y el *raster* es muy extenso para mantenerlo en memoria, los resultados son escritos en un fichero temporal (Greenberg et al. 2014; Hijmans 2013).

1.5.2.4 Paquete *rgdal*

El paquete *rgdal* permite la lectura y escritura de formatos de datos geoespaciales. Como biblioteca, presenta un único modelo abstracto de datos al uso que llama para todos los formatos soportados. También viene con una variedad de utilidades en línea de comando para la traducción y el procesamiento de datos geoespaciales (Pebesma et al. 2013).

1.6 Conclusiones parciales

Los estudios realizados exponen que, debido a la gran cantidad y diversidad de instituciones científicas y centros de investigación, que continuamente generan grandes volúmenes de datos e información en diversos formatos de datos científicos, existen problemas para su distribución y manipulación entre las comunidades de usuarios. Estos problemas residen principalmente en que no todos los formatos se rigen por los mismos estándares. Por tanto, algunas comunidades de usuarios tienen la necesidad de convertir y manipular datos a formatos que admitan sus aplicaciones.

De los formatos científicos para datos espacio-temporales analizados se selecciona *NetCDF* y *HDF* pues presentan una adecuada estructura para almacenar grandes volúmenes de datos espacio-temporales, además de que hay disimiles software que manipulan estos formatos. Para los datos geoespaciales se selecciona los modelos *raster* y *vectorial* al ser los más usados por las comunidades SIG.

Durante el estudio del lenguaje *R* se pudo constatar que posee grandes facilidades para la manipulación de formatos científicos de datos espacio-temporales y geográficos. *R* además de ser una alternativa de software libre cuenta con un gran repositorio de paquetes para facilitar y mejorar el trabajo con datos espacio-temporales y geoespaciales. De los paquetes de R para la manipulación de datos científicos y geográficos estudiados se selecciona *h5*, *RNetCDF*, *Raster* y *rgdal* pues facilitan la manipulación de estos tipos de formatos, además poseen una amplia documentación sobres sus principales funciones.

2 PAQUETE EN *R* PARA LA MANIPULACIÓN DE FORMATOS DE DATOS ESPACIO-TEMPORALES Y GEOESPACIALES.

En este capítulo se definen los principales aspectos a tener en cuenta en el diseño e implementación del paquete en *R.* Se presenta una descripción de las principales funcionalidades desarrolladas para dar solución al objetivo planteado.

2.1 Diseño del nuevo paquete realizado

El lenguaje UML (*Unified Modeling Language*) fue el utilizado para realizar el diseño del paquete, que tiene como objetivos principales la especificación, visualización, construcción y documentación de los productos de un sistema de software. Este lenguaje es usado por el RUP (*Rational Unified Process*) como lenguaje de modelado para lo cual se basa en todos sus tipos de diagramas, que constituyen diferentes vistas del modelo del producto.

Los diagramas UML empleados en el diseño del módulo fueron: diagrama de casos de uso y diagramas de actividades.

La herramienta empleada para el modelado de los casos de uso y los diagramas de actividades fue *Visual Paradigm* para UML versión 9.0.

2.1.1 Análisis de actores de casos de usos

Un caso de uso es una descripción de las actividades o pasos que se deben realizar para llevar a cabo un proceso. Un caso de uso es una secuencia de interacciones que se desarrolla entre un sistema y sus actores (Personaje o entidad que participa en un caso de uso) en respuesta a un evento que inicia un actor principal sobre el propio sistema.

El paquete implementado está destinado a un solo tipo de actor que es el especialista o investigador que desee trabajar con los formatos de datos científicos para series espacio-temporales y datos geográficos (Figura 2.1).

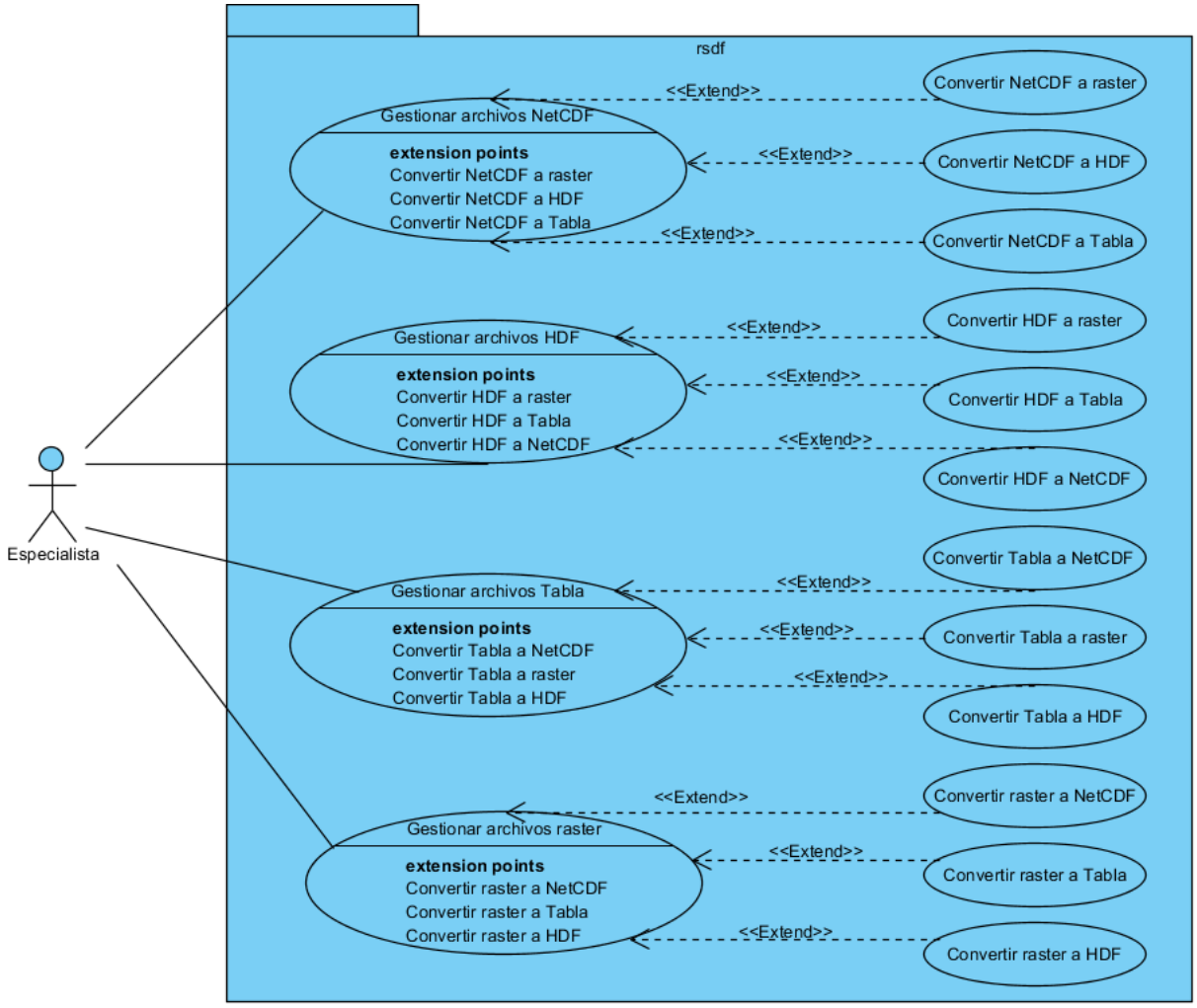

Figura 2.1 Diagrama de Casos de Usos.

El especialista cuanta con cuatro casos de usos generales: Gestionar archivos *NetCDF, HDF*, Tabla y *raster*, donde se puede trabajar con formatos para series espacio-temporales o para datos geoespaciales. El especialista puede contar con varias instancias del caso de uso Gestionar archivos *NetCDF*, que permite convertir de *NetCDF* a *raster*, a *HDF*, o a tabla. El caso de uso Gestionar archivos *HDF* permite convertir de *HDF* a *raster*, a *NetCDF*, o a tabla. El caso de uso Gestionar Tablas está dirigido a convertir tablas en formato *Excel* o *CSV* a formatos *HDF* o *NetCDF* (estándar o con la estructura permitida para su posterior análisis con la **Herramienta de animación de *NetCDF* espacio-temporal**). Por último, el caso de uso Gestionar archivos *raster* permite convertir uno o varios archivos *raster* al formato de datos *HDF o NetCDF*, con la opción de hacerlo de manera comprimida si lo que se desea es reducir su tamaño. Todos estos

casos de usos permiten obtener un resumen de las estadísticas para facilitar el análisis de los datos con los que se trabaja.

2.1.2 Diagramas de Actividades

Un diagrama de actividades muestra un proceso de negocio o un proceso de software como un flujo de trabajo a través de una serie de acciones. A continuación, se describen los diagramas de actividades de cada uno de los casos de usos del sistema vistos anteriormente (Figura 2.1).

Gestionar archivos *NetCDF*

El primer paso es seleccionar el archivo *NetCDF* que se desea convertir (Figura 2.2), una vez seleccionado, se tienen tres opciones de elección para el algoritmo de conversión en dependencia de a que formato se desea transformar.

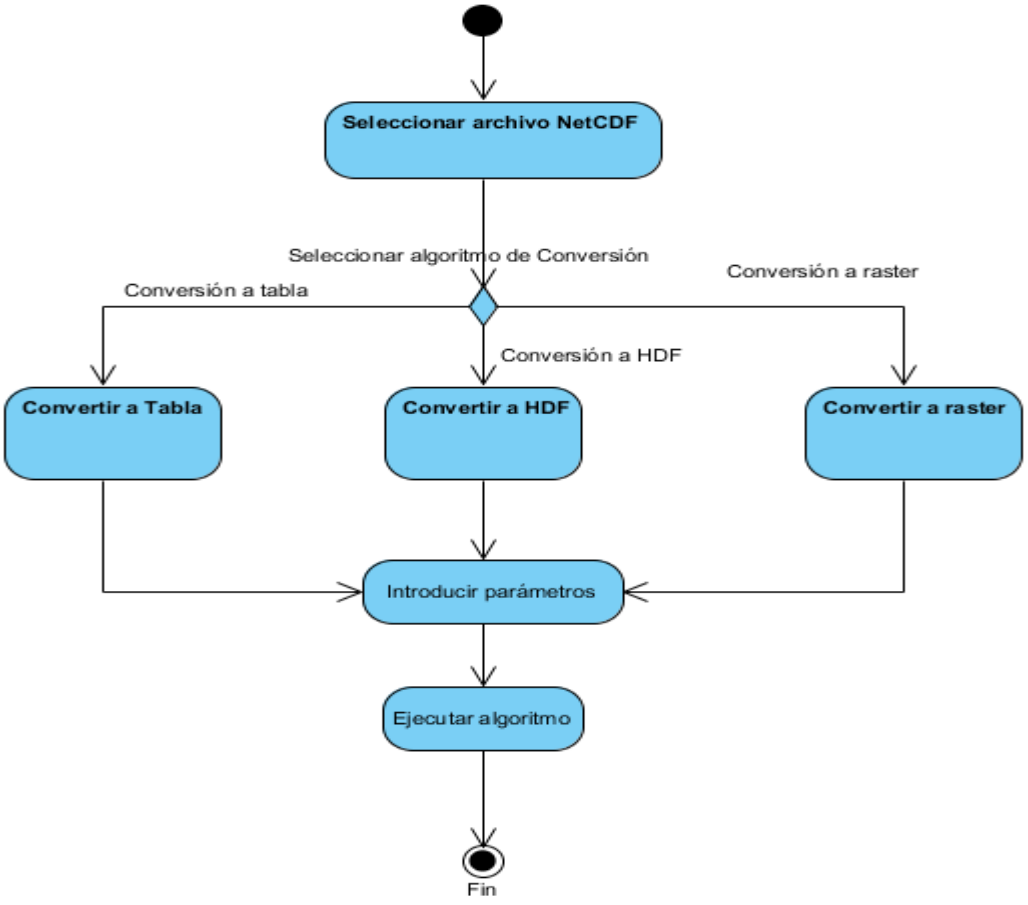

Figura 2.2 Diagrama de Actividad del caso de uso trabajo con archivos *NetCDF*.

Como se muestra en el diagrama una de las posibilidades es elegir convertir *NetCDF* a Tabla donde se pasa a introducir los parámetros:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.
- *Dataset* que se desea trasformar, este parámetro es opcional, pues en caso de que no se especifique se convierte todo el *NetCDF* a Tablas, esto quiere decir que cada *dataset* (incluyendo las dimensiones) del *NetCDF* se convierte a una hoja de la tabla Excel.

Otra posibilidad es convertir el *NetCDF* a *HDF* donde de igual forma se pasa a introducir los parámetros siguientes:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.
- *Dataset* que se desea trasformar, este parámetro es opcional, pues en caso de que no se especifique se convierte todo el *NetCDF* a *HDF* incluyendo las dimensiones.
- Nivel de compresión con la que se desea almacenar

Por último, se tiene la opción de convertir el *NetCDF* a *raster* donde como en los anteriores casos se pasa a introducir una serie de parámetros:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.
- Formato *raster* en el que se desea almacenar el nuevo archivo creado, este formato puede ser uno de los siguientes: *raster*, ascii, *SAGA*, *IDRISI*, *GTiff*, *ENVI*, *EHdr* y *HFA*. En caso de no especificar ninguno se guarda como *GTiff*.

Gestionar archivos *HDF*

En el trabajo con archivos *HDF* (Figura 2.3) primero se selecciona el archivo de entrada que se desea trasformar, una vez seleccionado, se tiene como en el caso anterior tres opciones de elección para el algoritmo de conversión.

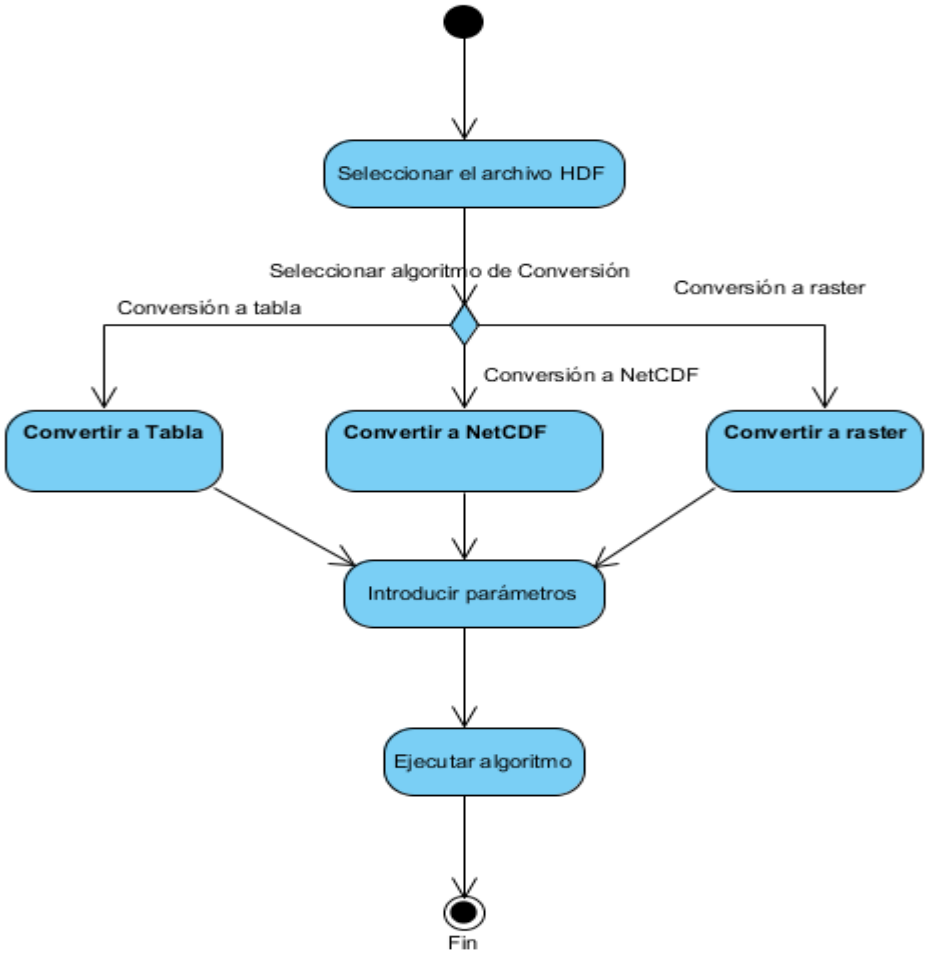

Figura 2.3 Diagrama de Actividades para el caso de Uso Trabajo con archivos *HDF*.

Una de las posibilidades de selección de algoritmo es convertir *HDF* a tabla donde se pasa a introducir los parámetros:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.

- *Dataset* que se desea trasformar, este parámetro es opcional, pues en caso de que no se especifique se convierte todo el *HDF* a Tablas, esto quiere decir que cada *dataset* del *HDF* se convierte a una hoja de la tabla *Excel*.

Otra posibilidad es convertir el *HDF* a *NetCDF* donde de igual forma se pasa a introducir los parámetros siguientes:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.
- *Dataset* que se desea trasformar, este parámetro es opcional, pues en caso de que no se especifique se convierte todo el *HDF* a *NetCDF*.
- Nivel de compresión con la que se desea almacenar

Por último, se tiene la opción de convertir el *HDF* a *raster* donde como en los anteriores casos se pasa a introducir una serie de parámetros:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.
- Formato *raster* en el que se desea almacenar el nuevo archivo creado, este formato puede ser uno de los siguientes: *raster*, ascii, *SAGA*, *IDRISI*, *GTiff*, *ENVI*, *EHdr* y *HFA*. En caso de no especificar ninguno se almacena como *GTiff*.

Gestionar archivos tablas

En el caso del trabajo con tablas como se observa en la Figura 2.4 primero se selecciona el archivo en formato tabla (*Excel* o *CSV*). Una vez que se haya seleccionado el archivo se puede mostrar una serie de estadísticas del mismo que le permite al especialista conocer la esencia de los datos. Por otra, existen tres opciones de elección del algoritmo de conversión.

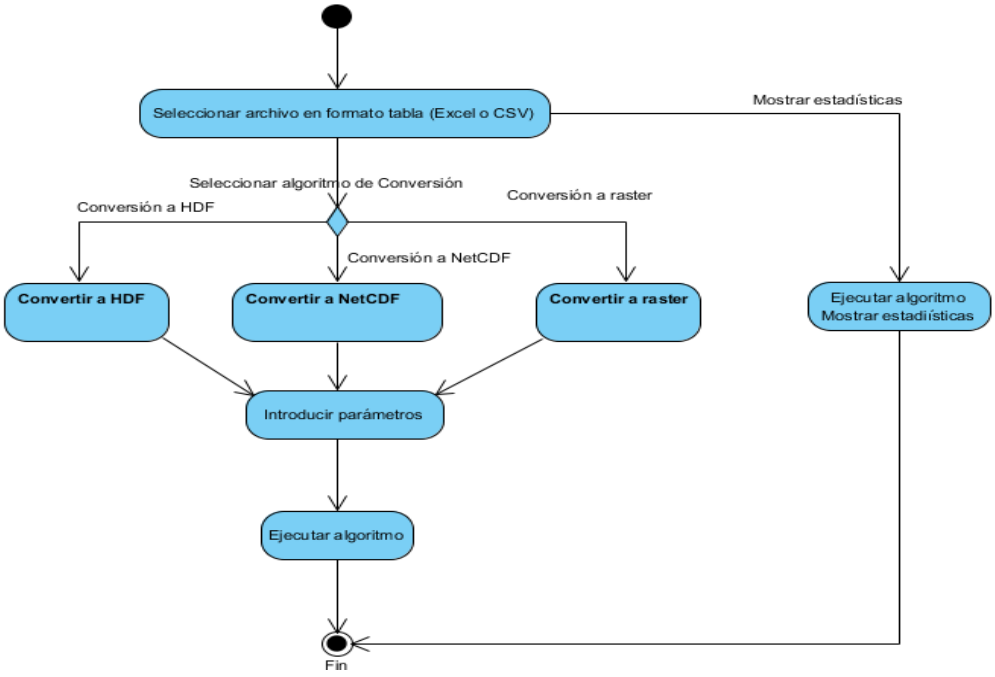

Figura 2.4 Diagrama de actividades para el Caso de Uso Trabajo con tablas.

De las posibilidades de conversión se tiene *Tabla a HDF* o *Tabla a NetCDF* donde se pasa a introducir los parámetros del algoritmo:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.
- Nivel de compresión con la que se desea almacenar

Por último, se tiene la opción de convertir el *Tabla* a *raster* donde como en los anteriores casos se pasa a introducir una serie de parámetros:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.
- Formato *raster* en el que se desea almacenar el nuevo archivo creado, este formato puede ser uno de los siguientes: *raster*, ascii, *SAGA*, *IDRISI*, *GTiff*, *ENVI*, *EHdr* y *HFA*. En caso de no especificar ninguno se almacena como *GTiff*.

Gestionar archivos *Raster*

En el caso del trabajo con tablas como se observa en la Figura 2.5 primero se selecciona el archivo *raster*. Una vez que se haya seleccionado el archivo se puede mostrar una serie de estadísticas del mismo que le permite al especialista conocer la esencia de los datos. Por otra, como en los anteriores casos, existen tres opciones de elección del algoritmo de conversión.

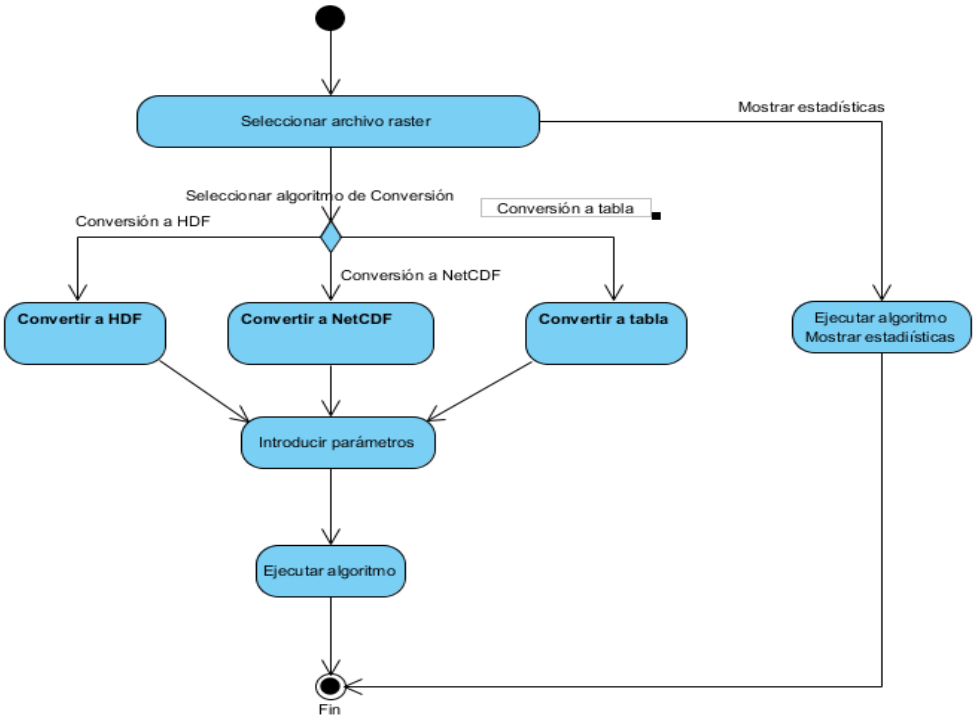

Figura 2.5 Diagrama de actividades para el Caso de Uso Trabajo con archivos *raster*.

De las posibilidades de conversión se tiene *raster a HDF* o *raster a NetCDF* donde se pasa a introducir los parámetros del algoritmo:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.
- Nivel de compresión con la que se desea almacenar

Por último, se tiene la opción de convertir el *raster* a *Tabla* donde como en los anteriores casos se pasa a introducir una serie de parámetros:

- Dirección para almacenar el nuevo archivo creado (se incluye el nombre del nuevo fichero a crear), este parámetro es opcional, pues en caso de que no se especifique una dirección se almacenara en la misma dirección y con el mismo nombre (excepto la extensión) que el fichero original.

2.2 Herramientas utilizadas en la implementación del paquete

Como lenguaje de programación se utilizó *R* que posee grandes facilidades para la manipulación de formatos científicos de datos espacio-temporales y geoespaciales. *R* es un lenguaje estadístico que permite analizar y manipular de forma eficiente grandes volúmenes de datos de diversos formatos, además de contar con múltiples paquetes para mejorar y facilitar el trabajo con datos espacio-temporales y geoespaciales.

Como Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) se utilizó *RStudio* en su versión (0.98.1103) por ser cómodo y fácil de usar para depurar programas hechos en *R*. Es software libre con licencia GPLv3 y se puede ejecutar sobre distintas plataformas (Windows, Mac, o Linux) o incluso desde la web usando *RStudio* Server. Incluye una consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo. Además, permite un análisis y desarrollo para que cualquiera pueda analizar los datos con *R* (Team 2015).

Algunas de sus principales características son:

- El resaltado de sintaxis, auto completado de código y sangría inteligente.
- Ejecutar código *R* directamente desde el editor de código fuente.
- Salto rápido a las funciones definidas.
- Documentación y soporte integrado.
- Administración sencilla de múltiples directorios de trabajo mediante proyectos.
- Navegación en espacios de trabajo y visor de datos.
- Depurador interactivo para diagnosticar y corregir los errores rápidamente.
- Herramientas de desarrollo extensas.

2.2.1 Paquetes usados en la implementación del nuevo módulo

El uso del paquete h5 proporciona una interfaz para la manipulación de archivos en formato *HDF5.* Entre los principales métodos de este paquete, que fueron utilizados, se encuentra:

- **h5file (name, mode = 'a'),** permite crear o abrir un archivo *HDF5.*
- **x [name] <- value**, esta sintaxis permite crear *grupos* y/o *datasets,* así como almacenar el valor (que puede ser vector, matriz o arreglo).
- **readDataSet (.Object, dspace = selectDataSpace (.Object))**, permite obtener el valor almacenado en el *dataset.*
- **h5attr (.Object, attributename, ...) <- value**, esta función permite crear y acceder a los atributos de los grupos, los *dataset* y/o el archivo.
- **h5close (.Object)**, la función permite cerrar un objeto de la clase *Dataset*, *H5Group* o *H5File*. Esta función es importante pues en caso de no cerrar uno de estos objetos no será posible trabajar con ellos hasta que no se haga esta operación.

Como se muestra en el manual de usuario *h5.pdf* (Annau 2016), el paquete h5 depende de una versión de R mayor o igual que la 3.2, además de que importa métodos del paquete *Rcpp* (>=0.11.5).

Para la manipulación de archivos *NetCDF* se usó el paquete *RNetCDF* que proporciona una interfaz para la manipulación eficiente de datos científicos en este formato. Entre los principales métodos de este paquete, que fueron utilizados, se encuentra:

- **create.nc(filename, clobber=TRUE, large=FALSE, share=FALSE, prefill=TRUE)**, permite crear un nuevo archivo *NetCDF* y nos devuelve un objeto de la clase "*NetCDF*".
- **open.nc(filename, write=FALSE, share=FALSE, prefill=TRUE, ...)**, con esta función se abre un archivo *NetCDF* para lectura y opcionalmente para escritura.
- **file.inq.nc(ncfile)**, esta función permite conocer el número de dimensiones, de variables, de atributos globales y el ID (identificador) de las dimensiones definidas de longitud ilimitada.

- **dim.inq.nc(ncfile, dimension)**, permite conocer información sobre las dimensiones del archivo *NetCDF*. La información que nos retorna es el nombre de las dimensiones, sus identificadores, la longitud de cada dimensión y una bandera para saber las dimensiones ilimitadas.
- **dim.def.nc(ncfile, dimname, dimlength=1, unlim=FALSE)**, permite crear nuevas dimensiones en el archivo *NetCDF*. El límite sugerido en el número de dimensiones es de 100. Después de creada una dimensión el nombre de esta puede ser cambiado, pero la longitud no puede ser cambiada, a menos que se copien todos los datados a una nueva dimensión con longitud redefinida.
- **var.def.nc(ncfile, varname, vartype, dimensions)**, permite crear nuevas variables en el archivo *NetCDF*.
- **var.inq.nc(ncfile, variable)**, esta función retorna información sobre las variables del archivo *NetCDF*. De la información que retorna permite conocer el nombre de la variable, su identificador, el tipo de la variable (NC_BYTE, NC_CHAR, NC_SHORT, NC_INT, NC_FLOAT y NC_DOUBLE), el número de dimensiones que contiene, un vector con los identificadores de las dimensiones de esta variable y el número de atributos que posee
  - **var.get.nc(ncfile, variable, start=NA, count=NA, na.mode=0, collapse=TRUE, unpack=FALSE, rawchar=FALSE)**, esta función devuelve el valor que contiene la variable. Variables numéricas siempre retornan el valor en *R* doble precisión (*2e-308* to *2e+308*), sin importar la precisión con la que este almacenada el *dataset* en el disco. Las variables del *NetCDF* de tipo NC_CHAR se devuelven como caracteres del tipo *R*.
- **var.put.nc(ncfile, variable, data, start=NA, count=NA, na.mode=0, pack=FALSE)**, permite escribir los valores en el archivo *NetCDF*.
- **att.inq.nc(ncfile, variable, attribute)**, permite conocer el identificador, nombre, tipo y longitud de los atributos.
- **att.put.nc(ncfile, variable, name, type, value)**, coloca un atributo dentro del *NetCDF*. El parámetro ***ncfile*** es un objeto de la clase *'NetCDF'* (valor que devuelve las funciones **open.nc** y **create.nc**) que apunta al archivo que se desea trabajar.
- **close.nc(con)**, esta función es de suma importancia pues permite cerrar un archivo o un *dataset* del *NetCDF*.

Como se puede apreciar en el manual de usuario *RNetCDF.pdf* (Michna 2016), el paquete *RNetCDF* depende de una versión de (*R* >=2.5.0).

Para la manipulación de archivos *raster* se usó el paquete *raster* que proporciona una interfaz para la manipulación eficiente de datos geográficos en este formato. Entre los principales métodos de este paquete, que fueron utilizados, se encuentra:

- **raster(x)**, permite crear una capa *raster*.
- **stack(x)**, permite crear varias capas *raster* con la misma extensión espacial y resolución.
- **writeRaster(raster, filename = raster.file, format = format, overwrite = T)**, permite escribir un objeto *raster* a un fichero usando uno de los formatos soportados. El parámetro ***raster*** es un objeto de *raster*.

Como se puede apreciar en el manual de usuario *raster.pdf* (Greenberg et al. 2014), el paquete *raster* depende de una versión de (*R* >=2.15.0) y de métodos del paquete *sp* (>=1.0-11).

Para la manipulación de archivos *Excel* se usó el paquete XLConnect que proporciona una interfaz para la escritura y lectura de datos en formato Excel. Entre los principales métodos de este paquete, que fueron utilizados, se encuentra:

- **writeWorksheetToFile (file, data, sheet, clearSheets = FALSE)**, permite escribir una hoja de cálculo en un fichero Excel.

Como se puede apreciar en el manual de usuario *XLConnect.pdf* (Mirai 2016), el paquete *XLConnect* depende de una versión de R (>= 2.10.0) y del paquete XLConnectJars (== 0.2-12). Además, importa funciones de *methods, rJava* y requiere tener instalado java (>= 1.6).

Para la manipulación de archivos *CSV* se usó la función *write.csv* y *read.csv* del paquete *utils* que se encuentra integrado a R como biblioteca del sistema.

- **write.csv(x, file)**, permite escribir un conjunto de datos a un fichero CSV.
- **read.csv(x)**, permite leer los datos de un fichero CSV.

Para la manipulación de archivos *dBase, SPSS, STATA* se usó el paquete *foreign* que proporciona una interfaz para la escritura y lectura de datos en formato DBF. Entre los principales métodos de este paquete, que fueron utilizados, se encuentra:

- **read.dbf(file)**, permite la lectura de ficheros en formato DBF (principalmente para *'shapefiles'*) y retorna un '*data frame'* con los datos provenientes del fichero DBF.
- **write.dbf(dataframe, file),** permite escribir un cuadro de datos (*data frame*) a un fichero en formato DBF.
- **write.foreign(df, datafile, codefile, package = c("SPSS", "Stata", "SAS"), ...),** permite escribir un cuadro de datos a un fichero en formato SPSS y STATA.

Como se puede apreciar en el manual de usuario *foreign.pdf* (Bivand et al. 2017), el paquete *foreign* depende de una versión de R (>= 3.0.0) y necesita ser compilado.

2.3 Método para añadir una nueva extensión a *R* desde *RStudio*

El proceso para escribir paquetes de R se describe detalladamente en el manual *Writing R extensions* (WRE) del R Core Team y que se puede ver en el *Comprehensive R Archive Network* (CRAN). En este documento se pretende concretar los pasos necesarios seguidos en R-Studio para la creación del paquete *Rsdf*. Aunque es posible añadir código en un lenguaje de bajo nivel (como C/C++/Fortran), vamos a centrarnos únicamente en el empaquetado de código y datos R, que fue lo utilizado en *Rsdf*.

¿Por qué crear un paquete de R?

Algunas de las principales razones son:

1. En primer lugar, crear un paquete nos obligar a pulir nuestras funciones, datos y código y, sobre todo, a documentar todo el trabajo y dar ejemplos claros. Las funciones son más fáciles de usar y podremos utilizar el comando '**?**' para ver detalles de los parámetros, los resultados y ejemplos.
2. Es el modo más elegante de compartir su trabajo. Los posibles usuarios del paquete agradecerán el esfuerzo de mejora y documentación que requiere la creación de un paquete. El intercambio con los usuarios también favorece la mejora del código.
3. Es la forma establecida para contribuir al crecimiento de R.

2.3.1 Pasos seguidos para la creación del paquete *Rsdf* con *RStudio*

La referencia fundamental para la construcción de un paquete está en los capítulos 1 y 2 del documento *Writing R extensions* (Team 2007) en los cuales se describen los detalles de cómo construir un nuevo paquete. Para la implementación del paquete *Rsdf* no fue necesario seguir todas las opciones que se detallan, por lo que es omitida la explicación de ellas.

Paso 1: Creación de la estructura de carpetas y archivos

Para crear un nuevo paquete con *RStudio* dar clic en el menú "File" y seleccionar la opción "*New Project*". Se muestra una ventana donde se debe seleccionar "*New Directory*" como se observa en la Figura 2.6.

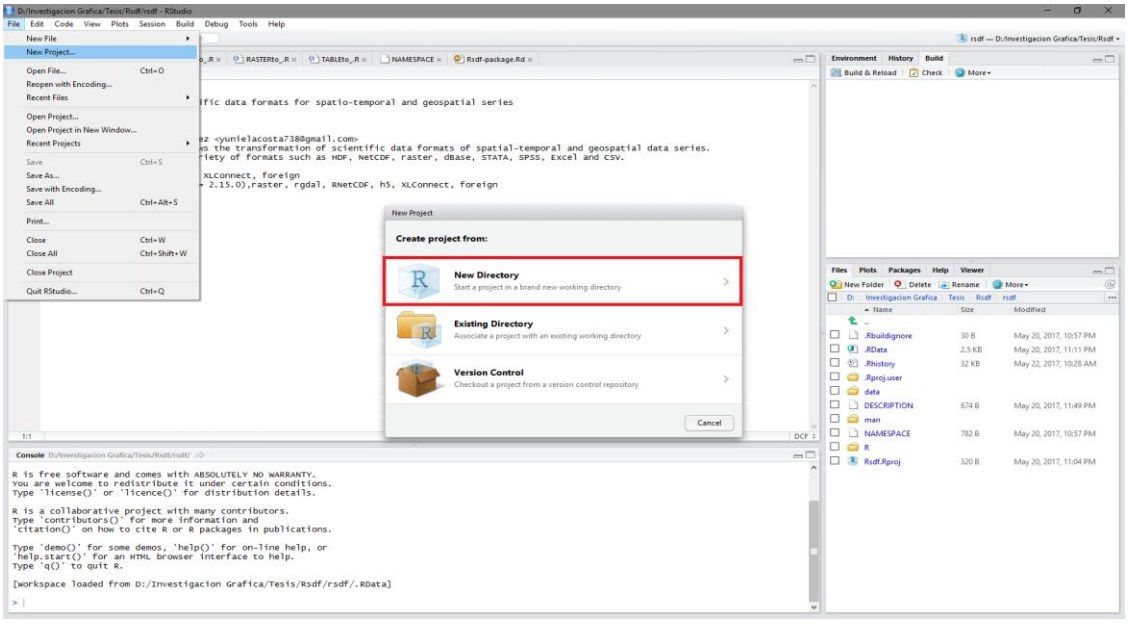

Figura 2.6 Creación de un nuevo paquete en *RStudio*.

Una vez selecciona la opción "*New Directory*", dar clic en "*R Package*".

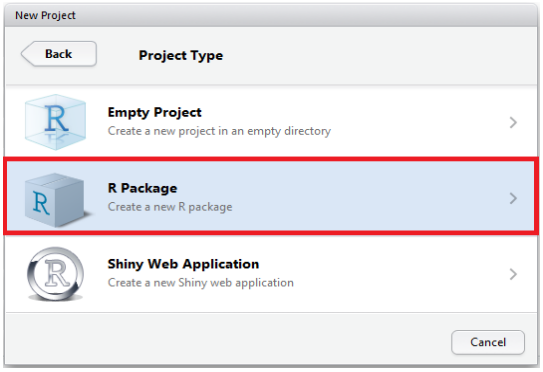

Figura 2.7 Seleccionar R Package.

Una vez seleccionado que se desea crear un paquete de R, se debe especificar el nombre del paquete, que es también el nombre del proyecto y de la carpeta que lo contiene. También hay que añadir los archivos fuentes con las funciones y datos que queremos incorporar al paquete como se muestra en la Figura 2.8. Por último, hay que especificar la carpeta donde se crea el paquete, así cuando se pulse el botón "*Create Project*", se crea toda la estructura de archivos necesaria dentro de la carpeta especificada y estaremos en disposición de rellenar los archivos de configuración y documentación del paquete.

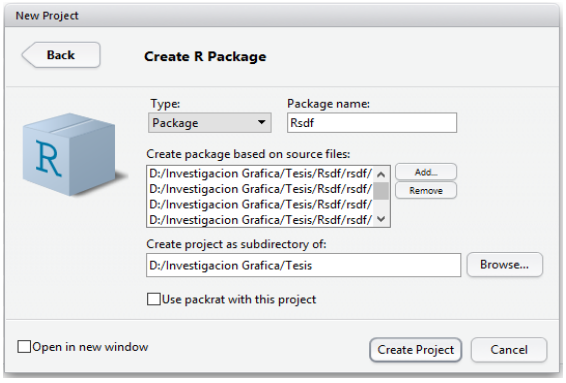

Figura 2.8 Crear paquete en R.

La estructura de carpetas y archivos del paquete *Rsdf* como se muestra en la Figura 2.9 consta del archivo DESCRIPTION y NAMESPACE fundamentales a la hora de construir el paquete.

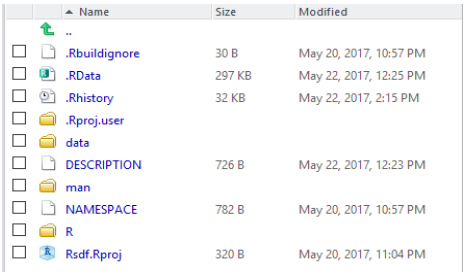

Figura 2.9 Estructura de carpetas y archivos del paquete *Rsdf*.

Además de estos dos archivos se encuentra la carpeta *R* que contiene todo el código fuente del paquete, la carpeta *data* con los datos y, por último, la carpeta *man* que posee los ficheros de ayuda y documentación del paquete.

Paso 2: Edición de los archivos de descripción

El archivo DESCRIPTION es fundamental pues le indica a R las características básicas del paquete y un breve resumen de su funcionalidad. En la Figura 2.10 se muestra el archivo DESCRIPTION del paquete *Rsdf* después de ser modificado.

```
Package: Rsdf
Type: Package
Title: Manipulation of scientific data formats for spatio-temporal and geospatial series
Version: 1.0
Date: 2017-06-21
Authors@R: c(person("Yuniel", "Acosta", "Pérez", role = c("aut", "cre", "cph"), email = "yunielacosta738@gmail.com"),
           person("Romel", "Vázquez ", "Rodríguez", role = c("cre", "cph"), email = "romelvazquez@gmail.com"),
           person("Andy", "Morfa", "Hernández", role = "ctb", email = "andym@uclv.cu"),
           person("Carlos", "Pérez", "Risquet", role = "ctb", email = "cperez@uclv.edu.cu"))
Author: Yuniel Acosta Pérez
Maintainer: Yuniel Acosta Pérez <yunielacosta738@gmail.com>, Romel Vázquez Rodríguez <romelvazquez@gmail.com>
Description: The package allows the transformation of scientific data formats of spatial-temporal and geospatial data series.
             It supports a variety of formats such as HDF, NetCDF, raster, dBase, STATA, SPSS, Excel and CSV.
URL: https://gitcoholic.uclv.edu.cu/Acosta/rsdf
Encoding:  UTF-8
BuildManual:yes
Imports: raster, RNetCDF, h5, XLConnect, foreign
Depends: R (>= 2.15.0), raster (>= 2.1-49), rgdal (>= 0.7-22), RNetCDF (>= 1.8-2), h5 (>= 0.9.8), XLConnect (>= 0.2-12), foreign (>= 0.8-66)
License: GPL (>= 3)
```

Figura 2.10 Archivo DESCRIPTION del paquete *Rsdf*.

Este es el conjunto mínimo de campos que tiene que tener un paquete de R. Existe más campos que son opcionales como se puede apreciar en el manual *Writing R extensions* (Team 2007), pero hay que tener mucho cuidado al rellenarlos ya que este archivo se consulta durante el proceso de creación del paquete. Importante resaltar el campo '*Depends'* proporciona una lista separada por comas con los nombres de los paquetes necesarios para que *Rsdf* trabaje correctamente. El campo '*Imports'* hace referencia a una lista separada por comas con los nombres de los paquetes de los que *Rsdf* importa clases y funciones. La fecha es al estilo yyyy-mm-dd. Si algún campo ocupa más de una línea, como el campo '*Description'*, la segunda y siguientes deben empezar por un tabulador o espacio en blanco. Aunque no es recomendable, hemos añadido el campo '*Encoding'* para utilizar el idioma propio (castellano, catalán) en los archivos de ayuda.

Paso 3: Documentación de las funciones

La subcarpeta *man* contiene los archivos de documentación (*Rd*) de cada uno de los objetos utilizables del paquete. Para cada función de la carpeta *R*, debemos crear un archivo de documentación (con la extensión *Rd*) en la carpeta *man*. R utiliza un lenguaje especial para crear estos archivos de documentación "*Rd*". La instalación de *R* se encarga de trasladar este lenguaje genérico *Rd* a archivos HTML, de texto o *LaTeX*, según se necesite. Este lenguaje de documentación tiene una semejanza evidente con el *LaTeX*. En todo caso no es necesario saber *LaTeX* para escribir la documentación de nuestro paquete.

El lenguaje "Rd" consiste en una serie de instrucciones con sus correspondientes argumentos. Cada instrucción es de la forma:

\command{arg}

El archivo "Rd" para una función contiene los puntos obligatorios '*name'*, '*alias'*, '*title'*, '*description'*, '*usage'* en la cabecera y '*keyword'* en el pie, además de los puntos opcionales como '*arguments'*, '*value'*, '*details'*, '*references'*, '*seealso'*, '*examples'* en el cuerpo del archivo.

Además de los puntos obligatorios, se recomienda añadir las secciones '*arguments'*, '*value'*, '*author'* y '*examples'*. Luego un archivo "Rd" típico tiene un mínimo de 10 secciones, cada una de las cuales empieza por su correspondiente instrucción:

```
\name{ ... }
\alias{ ... }
\title{ ... }
\description{ ... }
\usage{ ... }
\arguments{ ... }
\value{ ... }
\author{ ... }
\examples{ ... }
\keyword{ ... }
```

Figura 2.11 Estructura de un archivo Rd.

En todo caso la recomendación es empezar con un mínimo de puntos y añadir los que deseemos después de depurar de errores el archivo.

Instrucciones de formato "Rd"

En el capítulo 2 de WRE existe un extenso conjunto de instrucciones para dar formato. En el paquete *Rsdf* solo fueron utilizadas:

Para utilizar la fuente "*typewriter*" (especialmente para código R): \code{HDFtoNetCDF}

Insertar URL: \url{ http://www.r-project.org/ }

Insertar una dirección de correo: \email {yunielacosta738@gmail.com}

Paso 4: Chequear y crear el paquete

RStudio dispone de una pestaña específica para chequear el paquete.

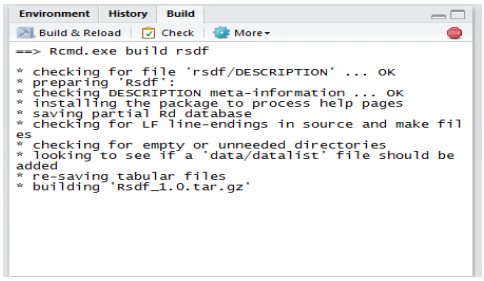

Figura 2.12 Pestaña *Build* en *RStudio*.

Si pulsamos el icono *'Check'* se chequea el paquete y veremos los errores o advertencias. Corregidos los errores y todas las advertencias que podamos, el siguiente paso lo podemos hacer con el icono *'Build & Reload'* que construye el paquete y lo carga para que lo probemos.

Paso 6: Distribuir el paquete

Con el icono *'More'* de la Figura 2.12 se despliega un menú que permite crear el paquete en la versión "*zip*" en Windows con la opción *'Build Binary Package'*.

2.4 Arquitectura general del paquete *Rsdf*

Con el paquete *Rsdf* se puede trabajar desde *RStudio*, *R GUI* o *RKWard* lo que facilita la interacción con los usuarios de este paquete.

La arquitectura general del paquete *Rsdf* parte de la interacción de *RStudio*, *R GUI* o *RKWard* (en dependencia de con que IDE se desee trabajar) con el motor de trabajo R el cual se encarga de trabajar con los datos y de ejecutar todas las instrucciones de las funciones del paquete *Rsdf*. Como se puede apreciar en la Figura 2.13, *Rsdf* para que funcione correctamente depende de funciones de otros paquetes como son: *RNetCDF*, *raster*, *h5*, *XLConnect* y *foreign.* Estos paquetes a su vez dependen de otros que pueden servir de enlace con bibliotecas externas como es el caso *rgeos* y *rgdal* que interactúan con *GEOS*, *GDAL*, *OGR* y *proj*, importantes bibliotecas para la manipulación de datos Geoespaciales.

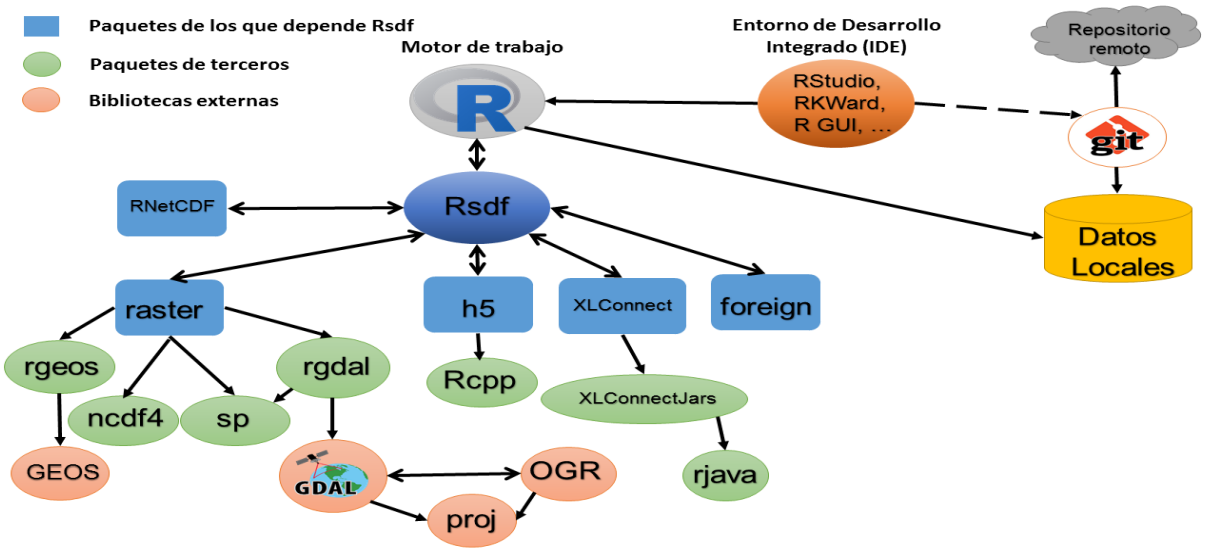

Figura 2.13 Arquitectura general del paquete *Rsdf*.

2.5 Principales funcionalidades del paquete

El paquete *Rsdf* permite la transformación de formatos científicos para series de datos espacio-temporales y geoespaciales. Actualmente soporta los siguientes formatos: *HDF*, *NetCDF*, *Raster*, *Vectoriales (Shapefiles)*, *CSV*, *Excel*, *DBF*, *SPSS*, *STATA*.

El paquete *Rsdf* en su versión 1.0 cuenta con 38 funciones, en la Tabla 2.1 aparece cada una de ellas separadas por el tipo de formato de datos que transforman.

HDF	NetCDF	Raster	Tabla
HDFtoNetCDF	NetCDFtoHDF	rasterToNetCDF	CSVtoHDF
HDFtoRaster	NetCDFtoRaster	rasterToHDF	CSVtoNetCDF
HDFtoCSV	NetCDFtoCSV	rasterToCSV	CSVtoRaster
HDFtoExcel	NetCDFtoExcel	rasterToExcel	ExceltoHDF
HDFtoDBF	NetCDFtoDBF	rasterToDBF	ExceltoNetCDF
HDFtoSPSS	NetCDFtoSPSS	rasterToSPSS	ExceltoRaster
HDFtoSTATA	NetCDFtoSTATA	rasterToSTATA	DBFtoHDF
		rasterToShape	DBFtoNetCDF
		shapeToRaster	DBFtoRaster
			SPSStoHDF
			SPSStoNetCDF

	SPSStoRaster
	STATAtoHDF
	STATAtoNetCDF
	STATAtoRaster

Tabla 2.1 Funciones que componen el paquete *Rsdf*

Para una mejor comprensión de cada una de las funciones se realiza un análisis detallado sobre algunas cuestiones importantes que no se pueden pasar por alto.

HDFtoNetCDF, HDFtoRaster, HDFtoCSV, HDFtoExcel, HDFtoDBF, HDFtoSPSS y HDFtoSTATA

Las funciones de la Tabla 2.2 permiten convertir un fichero en formato *HDF* a uno en formato *NetCDF, raster, csv, Excel, dbf, SPSS y STATA* respectivamente. Como elemento esencial hay que introducir el nombre absoluto del archivo *HDF* a transformar y en dependencia de la función que se desee utilizar el resto de los parámetros varía como muestra la Tabla 2.3. Se debe tener en cuenta que la trasformación de un archivo *HDF* a uno en formato *NetCDF* hay que especificar los *datasets* del fichero *HDF* que constituyen dimensiones, en caso de que este parámetro no se especifique cuando se ejecute la función, esta solicita los *datasets* que representan dimensiones. Si no se tienen conocimientos sobre cuales *datasets* pueden constituir dimensiones se debe escribir NA. La conversión de un *HDF* a un *raster* permite especificar el formato (*raster, ascii, SAGA, IDRISI, GTiff, ENVI, EHdr y HFA*) (Figura 2.14) con el que se desea guardar el nuevo archivo *raster*. Se tiene la opción de puntualizar la proyección del *raster*, además si no se desea convertir todo el *HDF* a un *raster* se puede especificar que *datasets* del *HDF* se desean transformar. Cuando se pretende transformar un *HDF* a un *raster* hay que tener en cuenta que si se escoge un formato *raster* que soporte multicapas (Figura 2.14) y los *datasets* del *HDF* son de distintos tamaños esto proporciona un error, pues todas las capas del *raster* tienen que ser del mismo tamaño. Una posible solución es escoger con el parámetro *datasets* cuales valores transformar o permitir que se escriba por cada conjunto de datos un fichero *raster* poniendo el parámetro *bylayer* en VERDADERO. El parámetro *dim* permite definir que dimensiones tomar en caso de que el *dataset* sea multidimensional. Por ejemplo, un *dataset* con dimensiones 120x360x1444 se puede escoger que dimensiones guardar dim = c (120,360). Convertir un fichero *HDF* a un fichero Excel solo se considera que se puede transformar cada *dataset* o un

conjunto de *datasets* a una hoja de cálculo en el archivo Excel. Cuando se transforma un archivo en formato *HDF* a DBF, CSV, SPSS o STATA esta operación se realiza convirtiendo cada *dataset* en un fichero donde el nombre está compuesto por el nombre especificado en el segundo parámetro y el nombre del *dataset*.

File type	Long name	default extension	Multiband support
raster	'Native' raster package format	.grd	Yes
ascii	ESRI Ascii	.asc	No
SAGA	SAGA GIS	.sdat	No
IDRISI	IDRISI	.rst	No
GTiff	GeoTiff (requires rgdal)	.tif	Yes
ENVI	ENVI .hdr Labelled	.envi	Yes
EHdr	ESRI .hdr Labelled	.bil	Yes
HFA	Erdas Imagine Images (.img)	.img	Yes

Figura 2.14 Formatos *raster* soportados

La Tabla 2.2 muestra la **signatura** de cómo usar las funciones:

Funciones del fichero HDFto_
`HDFtoNetCDF(hdf.file, nc.file = hdf.file, dimensions, pack = TRUE, verbose = FALSE)`
`HDFtoRaster(hdf.file, raster.file = hdf.file, format = "raster", projection = NA, datasets, bylayer = FALSE, dim, verbose = FALSE)`
`HDFtoCSV(hdf.file, csv.file = hdf.file, verbose = FALSE)`
`HDFtoExcel(hdf.file, excel.file = hdf.file, datasets,verbose = FALSE)`
`HDFtoDBF(hdf.file, dbf.file = hdf.file, verbose = FALSE)`
`HDFtoSPSS(hdf.file, spss.file = hdf.file, verbose = FALSE)`
`HDFtoSTATA(hdf.file, stata.file = hdf.file, verbose = FALSE)`

Tabla 2.2 Encabezado de las funciones que permiten convertir un fichero *HDF* a otro formato

La Tabla 2.3 muestra los argumentos pertenecientes a las funciones y su descripción:

Parámetro	**Descripción**
`hdf.file`	Nombre del fichero *HDF* que se desea transformar.

`nc.file`	Nombre del fichero de salida *NetCDF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`raster.file`	Nombre del fichero de salida *Raster*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`csv.file`	Nombre del fichero de salida *CSV*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`excel.file`	Nombre del fichero de salida *Excel*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`dbf.file`	Nombre del fichero de salida *DBF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`spss.file`	Nombre del fichero de salida *SPSS*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`stata.file`	Nombre del fichero de salida *STATA*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`format`	El formato de salida debe ser uno de los siguientes: (*ascii, SAGA, IDRISI, GTiff, ENVI, EHdr y HFA*). Si no se proporciona este argumento, se intenta inferirlo desde la extensión del nombre de archivo. Si falla, se utiliza el formato predeterminado. El formato predeterminado es *'raster'*.
`projection`	Proyecciones se realizan con la biblioteca PROJ.4 expuesta por rgdal. Ejemplo: "+proj=lcc +lat_1=48 +lat_2=33 +lon_0=-100 +ellps=WGS84".
`datasets`	Vector o lista con el nombre de los *datasets* que se desean transformar en caso de que no se especifique se transforma cada *dataset* del fichero *HDF* en una capa del fichero de salida *raster* (si el formato en el que se desea guardar soporta multicapas) o en una hoja de cálculo del fichero de salida Excel.
`bylayer`	Si es verdadero, se escribe cada capa en ficheros separados.
`dim`	Define las dimensiones que se almacenaran en caso que el *dataset* posea más de dos dimensiones. Si este parámetro no se define se toman las dos primeras dimensiones.
`dimensions`	Vector o lista que contiene el nombre de los *dataset* del *HDF* que constituyen dimensiones que determinan a los otros *datasets*. En caso de no conocer que *dataset* puede representar una dimensión no especificar este parámetro.

`pack`	Las variables se empaquetan si pack = TRUE.
`verbose`	Reportar el progreso y un resumen de los datos.

Tabla 2.3 Argumentos de las funciones pertenecientes a los archivos *HDF*

NetCDFtoHDF, NetCDFtoRaster, NetCDFtoCSV, NetCDFtoExcel, NetCDFtoDBF, NetCDFtoSPSS y NetCDFtoSTATA

Las funciones de la Tabla 2.4 permiten convertir un fichero en formato *NetCDF* a uno en formato *HDF, raster, csv, Excel, dbf, SPSS y STATA* respectivamente. Como elemento esencial hay que introducir el nombre absoluto del archivo *NetCDF* a transformar y en dependencia de la función que se desee utilizar el resto de los parámetros varía como muestra la Tabla 2.5. La transformación de un archivo *NetCDF* a uno en formato *HDF* o Excel permite especificar que *datasets* del *NetCDF* que se desean transformar. En caso de que el *NetCDF* posea dimensiones degeneradas (longitud = 1), estas pueden ser omitidas con el parámetro *collapse* en VERDADERO. La conversión de un *NetCDF* a un *raster* permite especificar el formato con el que se desea guardar. El formato del archivo *raster* puede ser (*raster, ascii, SAGA, IDRISI, GTiff, ENVI, EHdr y HFA*) como se muestra en la Figura 2.14. Se tiene la opción de puntualizar la proyección del *raster*, además si no se desea convertir todo el NetCDF a un *raster* se puede especificar que *datasets* del NetCDF se desean transformar. Cuando se pretende transformar un *HDF* a un raster hay que tener en cuenta que si se escoge un formato *raster* que soporte multicapas (Figura 2.14) y los *datasets* del *HDF* son de distintos tamaños esto proporciona un error, pues todas las capas del *raster* tienen que ser del mismo tamaño. Una posible solución es escoger con el parámetro *datasets* cuales valores transformar o permitir que se escriba por cada conjunto de datos un fichero raster poniendo el parámetro *bylayer* en VERDADERO. Cuando se transforma un archivo en formato NetCDF a DBF, CSV, SPSS o STATA esta operación se realiza convirtiendo cada *dataset* en un fichero donde el nombre está compuesto por el nombre especificado en el segundo parámetro y el nombre del *dataset*.

La Tabla 2.4 muestra la **signatura** de cómo usar las funciones:

Funciones del fichero NetCDFto_
`NetCDFtoHDF (ncfile, h5file = ncfile, unpack = TRUE, collapse = FALSE, datasets, verbose = FALSE)`

NetCDFtoRaster(ncfile, raster.file = ncfile, format ='raster', projection = NA, bylayer = FALSE, verbose = FALSE)
NetCDFtoCSV(ncfile, csv = ncfile, collapse = FALSE, verbose = FALSE)
NetCDFtoExcel(ncfile, excel.file = ncfile, unpack = TRUE, collapse = FALSE, datasets, verbose = FALSE)
NetCDFtoDBF(ncfile, dbf.file = ncfile, collapse = FALSE, verbose = FALSE)
NetCDFtoSPSS(ncfile, spss.file = ncfile, collapse = FALSE, verbose = FALSE)
NetCDFtoSTATA(ncfile, stata.file = ncfile, collapse = FALSE, verbose = FALSE)

Tabla 2.4 Encabezado de las funciones que permiten convertir un fichero *NetCDF* a otro formato

La Tabla 2.5 muestra los argumentos pertenecientes a las funciones y su descripción:

Parámetro	Descripción
`ncfile`	Nombre del fichero *NetCDF* que se desea transformar.
`h5file`	Nombre del fichero de salida *HDF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`raster.file`	Nombre del fichero de salida *Raster*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`csv`	Nombre del fichero de salida *CSV*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`excel.file`	Nombre del fichero de salida *Excel*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`dbf.file`	Nombre del fichero de salida *DBF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`spss.file`	Nombre del fichero de salida *SPSS*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`stata.file`	Nombre del fichero de salida STATA. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`unpack`	Las variables empaquetadas se desempaquetan si se descomprime = TRUE y se definen los atributos *add_offset* y *scale_factor*. El valor predeterminado es FALSE.
`collapse`	VERDADERO si se deben omitir las dimensiones degeneradas (longitud = 1).

`datasets`	Vector o lista con el nombre de los *datasets* que se desean transformar en caso de que no se especifique se transforma cada *dataset* del fichero *NetCDF* en uno del fichero de salida *HDF*.
`format`	El formato de salida debe ser uno de los siguientes: (*ascii, SAGA, IDRISI, GTiff, ENVI, EHdr y HFA*). Si no se proporciona este argumento, se intenta inferirlo desde la extensión del nombre de archivo. Si falla, se utiliza el formato predeterminado. El formato predeterminado es 'raster'.
`projection`	Proyecciones se realizan con la biblioteca PROJ.4 expuesta por rgdal. Ejemplo: "+proj=lcc +lat_1=48 +lat_2=33 +lon_0=-100 +ellps=WGS84".
`datasets`	Vector con el nombre de los *datasets* que se desean transformar en caso de que no se especifique se transforma cada *dataset* del fichero *NetCDF* en una hoja de cálculo del fichero de salida Excel.
`verbose`	Reportar el progreso y un resumen de los datos.

Tabla 2.5 Argumentos de las funciones pertenecientes a los archivos *NetCDF*

rasterToNetCDF, rasterToHDF, rasterToCSV, rasterToExcel, rasterToDBF, rasterToSPSS, rasterToSTATA, rasterToShape y shapeToRaster

Las funciones de la Tabla 2.6 permiten convertir un fichero en formato *raster* a uno en formato *NetCDF, HDF, csv, Excel, dbf, SPSS y STATA* respectivamente. Como elemento esencial hay que introducir el nombre absoluto del archivo *raster* a transformar y en dependencia de la función que se desee utilizar el resto de los parámetros varia como muestra la Tabla 2.7. Transformar de un *raster* a *NetCDF*, *HDF* y Excel convierte cada capa del *raster* (si es multicapa) en un *dataset* (*NetCDF*, *HDF*) o en una hoja de cálculo del Excel. Convertir un *raster* a un *Shapefile* permite seleccionar el tipo de formato vectorial a guardar (Puntos, Contorno, Polígonos), así como la función para seleccionar un subconjunto de valores *raster* (sólo permitido si fichero *raster* tiene una sola capa). Transformar un *Shapefile* a un *raster* permite especificar el formato que se desea guardar como se muestra la Figura 2.14. Nombre de la columna a utilizar para la dimensión z en el *raster*.

La Tabla 2.6 muestra la **signatura** de cómo usar las funciones:

Funciones del fichero RASTERto_
`rasterToNetCDF(raster.file, ncfile = raster.file, verbose = FALSE)`
`rasterToHDF(raster.file, h5file = raster.file, verbose = FALSE)`
`rasterToCSV(raster.file, csv.file = raster.file, verbose = TRUE)`
`rasterToExcel(raster.file, excel.file = raster.file, verbose = FALSE)`
`rasterToDBF(raster.file, dbf.file = raster.file, verbose = TRUE)`
`rasterToSPSS(raster.file, spss.file = raster.file, verbose = FALSE)`
`rasterToSTATA(raster.file, stata.file = raster.file, verbose = FALSE)`
`rasterToShape(raster.file, out.file = raster.file, shape.type = 'Points', fun=NULL, verbose = FALSE, ...)`
`shapeToRaster(shpfile, raster.out = shpfile, format = "raster", ncells = 99, cellsize = NA, ncellwarn = 1000, column = "", verbose = TRUE, ...)`

Tabla 2.6 Encabezado de las funciones que permiten convertir un fichero *raster* a otro formato

La Tabla 2.7 muestra los argumentos pertenecientes a las funciones y su descripción:

Parámetro	**Descripción**
`raster.file`	Nombre del fichero *Raster* que se desea transformar.
`shpfile`	Nombre del fichero a transformar como "coolstuff.shp" o un Objeto Spatial*DataFrame.
`ncfile`	Nombre del fichero de salida *NetCDF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`h5file`	Nombre del fichero de salida *HDF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`csv.file`	Nombre del fichero de salida *CSV*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`excel.file`	Nombre del fichero de salida *Excel*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`dbf.file`	Nombre del fichero de salida *DBF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.

`spss.file`	Nombre del fichero de salida *SPSS*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`stata.file`	Nombre del fichero de salida *STATA*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`out.file`	Nombre del fichero de salida *Shapefile*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`raster.out`	Nombre del fichero de salida *Raster*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`shape.type`	Forma en la que se guarda el fichero en formato vectorial. Puede ser ('Points', 'Contour', 'Polygons'). Predeterminado 'Points'.
`fun`	Función para seleccionar un subconjunto de valores *raster* (sólo permitido si raster.file tiene una sola capa).
`format`	El formato de salida debe ser uno de los siguientes: (ascii, SAGA, IDRISI, GTiff, ENVI, EHdr y HFA). Si no se proporciona este argumento, se intenta inferirlo desde la extensión del nombre de archivo. Si falla, se utiliza el formato predeterminado. El formato predeterminado es 'raster'.
`ncells`	Si no se da, se infiere de la extensión del *Shapefile* y el tamaño de la celda.
`cellsize`	Tamaño de la celda en unidades de coordenadas (generalmente grados o m). Se calcula a partir de *ncell* si es NA.
`ncellwarn`	Advierta si habrá más celdas que esto. Para evitar por ejemplo grados accidentales en lugar de km.
`column`	Nombre de la columna a utilizar para la dimensión z en el *raster*. Cadena vacía para la selección interactiva
`verbose`	Reportar el progreso y un resumen de los datos.

Tabla 2.7 Argumentos de las funciones pertenecientes a los archivos *raster*

CSVtoHDF, CSVtoNetCDF, CSVtoRaster, ExceltoHDF, ExceltoNetCDF, ExceltoRaster, DBFtoHDF, DBFtoNetCDF, DBFtoRaster, SPSStoHDF, SPSStoNetCDF, SPSStoRaster, STATAtoHDF, STATAtoNetCDF y STATAtoRaster

Las funciones de la Tabla 2.8 permiten convertir un fichero en formato CSV, Excel, DBF, SPSS, STATA a uno en formato *NetCDF*, *HDF* y *raster* respectivamente. Transformar un archivo en formato *CSV, DBF, SPSS, STATA* a un fichero en formato *NetCDF, HDF* o *raster* hay que tener en cuenta el nombre con el que se desea guardar el *dataset*, en caso de que no se especifique el nombre del *dataset* se toma del nombre del fichero de entrada. Transformar un archivo en formato Excel a *NetCDF, HDF* o *raster* se puede especificar la región que se desea convertir en un *dataset* del archivo *HDF, NetCDF* o en una capa del archivo *raster*, así como las hojas que sean de interés transformar.

La Tabla 2.8 muestra la signatura de cómo usar las funciones:

Funciones del fichero TABLEto_
`CSVtoHDF(csv.file, hdf.file = csv.file, dataset.name = dataset_name(filename = csv.file), verbose = FALSE)`
`CSVtoNetCDF(csv.file, nc.file = csv.file, dataset.name = dataset_name(filename = csv.file), verbose = FALSE)`
`CSVtoRaster(csv.file, raster.file = csv.file, format ='raster',projection = NA, verbose = FALSE)`
`ExceltoHDF(excel.file, hdf.file = excel.file, region, sheets, verbose = FALSE)`
`ExceltoNetCDF(excel.file, nc.file = excel.file, region, sheets, verbose = TRUE)`
`ExceltoRaster(excel.file, raster.file = excel.file, format = "raster", region, bylayer = FALSE, projection = NA, sheets, verbose = FALSE)`
`DBFtoHDF(dbf.file, hdf.file = dbf.file, datasets, verbose = FALSE)`
`DBFtoNetCDF(dbf.file, nc.file = dbf.file, datasets, verbose = FALSE)`
`DBFtoRaster(dbf.file, raster.file = dbf.file, format ='raster', datasets, projection = NA verbose = FALSE)`
`SPSStoHDF(spss.file, hdf.file = spss.file, dataset.name = dataset_name(filename = spss.file), verbose = FALSE)`

SPSStoNetCDF(spss.file, nc.file = spss.file, dataset.name = dataset_name(filename = spss.file), verbose = FALSE)
SPSStoRaster(spss.file, raster.file = spss.file, format ='raster', projection = NA, verbose = FALSE)
STATAtoHDF(stata.file, hdf.file = stata.file, dataset.name = dataset_name(filename = stata.file), verbose = FALSE)
STATAtoNetCDF(stata.file, nc.file = stata.file, dataset.name = dataset_name(filename = stata.file), verbose = FALSE)
STATAtoRaster (stata.file, raster.file = stata.file, format ='raster', projection = NA, verbose = FALSE)

Tabla 2.8 Encabezado de las funciones que permiten convertir un fichero Tabla a otro formato

La Tabla 2.9 muestra los argumentos pertenecientes a las funciones y su descripción:

Parámetro	Descripción
`csv.file`	Nombre del fichero *CSV* que se desea transformar.
`excel.file`	Nombre del fichero *Excel* que se desea transformar.
`dbf.file`	Nombre del fichero *DBF* (dBase) que se desea transformar.
`spss.file`	Nombre del fichero *SPSS* que se desea transformar.
`stata.file`	Nombre del fichero STATA que se desea transformar.
`hdf.file`	Nombre del fichero de salida *HDF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`nc.file`	Nombre del fichero de salida *NetCDF*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`raster.file`	Nombre del fichero de salida *raster*. Por defecto se guarda en la misma carpeta y con el mismo nombre que el fichero de entrada.
`dataset.name`	Nombre con el que se desea guardar el *dataset*. Por defecto se usa el nombre del fichero *CSV, SPSS o STATA* (en dependencia de la función que se use).
`format`	El formato de salida debe ser uno de los siguientes: (*ascii, SAGA, IDRISI, GTiff, ENVI, EHdr y HFA*). Si no se proporciona este argumento,

	se intenta inferirlo desde la extensión del nombre de archivo. Si falla, se utiliza el formato predeterminado. El formato predeterminado es 'raster'.
`projection`	Proyecciones se realizan con la biblioteca PROJ.4 expuesta por rgdal. Ejemplo: "+proj=lcc +lat_1=48 +lat_2=33 +lon_0=-100 +ellps=WGS84".
`region`	Un especificador de rango en la forma 'A10: B18'. También puede ser un vector de las siguiente forma c(startRow, startCol, endRow, endCol) donde el vector puede contener un solo valor en el caso que solo se especifique la fila inicial, dos en el caso que se especifique la fila y columna inicial; tres en el caso que se especifique la fila inicial, columna inicial y la fila final. Por último, puede tener cuatro valores en el caso que se especifique la fila inicial, columna inicial, fila final y columna final.
`sheets`	Las hojas del archivo *Excel* que se desean transformar. En caso de no especificarse se transforma cada hoja en un *dataset* del nuevo archivo *HDF* o del archivo *NetCDF*, o del fichero *raster*.
`bylayer`	Si es verdadero, se escribe cada capa en ficheros separados.
`datasets`	Conjunto de datos del fichero DBF (en caso de conocer el nombre de los *datasets* que contiene el fichero DBF).
`verbose`	Reportar el progreso y un resumen de los datos.

Tabla 2.9 Argumentos de las funciones pertenecientes a los archivos Tabla

2.6 Conclusiones parciales

La implementación de un paquete para la manipulación y transformación con archivos en formatos de datos científicos para series espacio temporales y geoespaciales contribuye al análisis y visualización de grandes volúmenes de información mediante la conversión de estos archivos a *HDF*, *NetCDF*, *raster*, vectorial y Tablas. Además, se presentó el análisis de los actores y casos de uso del sistema. Se mostró la arquitectura general del paquete que se realizó. Se mostraron las acciones que son necesarias realizar para la incorporación de una nueva extensión (paquete) a *R* y las principales funcionalidades del paquete *Rsdf*.

3 USO Y VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA.

En este capítulo se expone mediante un caso de estudio como utilizar los algoritmos implementados en el paquete con el objetivo de validar y chequear los resultados obtenidos. Así como la posterior visualización de los datos de los resultados.

3.1 Instalación del paquete desarrollado

El paquete *Rsdf* presenta como requerimientos tener instalado R en su versión (>= 2.15.0), además necesita importar métodos de los paquetes raster (>= 2.1-49), rgdal (>= 0.7-22), RNetCDF (>= 1.8-2), h5 (>= 0.9.8), XLConnect (>= 0.2-12), foreign (>= 0.8-66). También es necesario tener instalado en el sistema java en su versión (>= 1.6).

3.1.1 Pasos a seguir para instalar el paquete *Rsdf* en *RStudio*

La instalación del paquete en *RStudio* se puede realizar desde el repositorio CRAN o desde el archivo binario del paquete. En este epígrafe se explica cómo instalar el paquete en *RStudio* desde el archivo binario.

Para instalar un nuevo paquete se puede utilizar el comando `install.packages` donde se pasa por parámetro el archivo comprimido que contiene la instalación del paquete y como es local la instalación el repositorio es *null*. A continuación, se muestra como se usa el comando:

```
install.packages("D:/Investigacion Grafica/Tesis/Build package/Rsdf_1.0.zip", repos = NULL)
```

Otra vía es seleccionar en el menú *Tools* la opción *Install Package* o desde la paleta *Package* el botón *Install* como se muestra en la Figura 3.1.

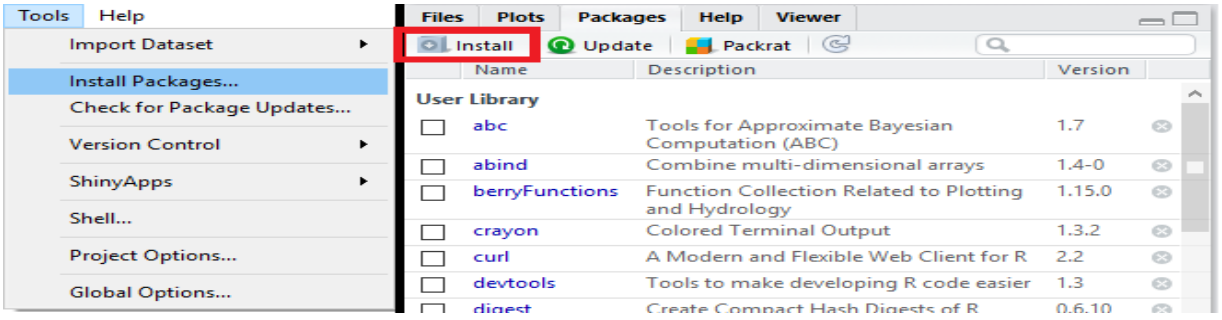

Figura 3.1 *RStudio* instalación de un nuevo paquete

La ventana *Install Packages* (Figura 3.2) permite instalar paquetes desde el CRAN o desde un fichero comprimido.

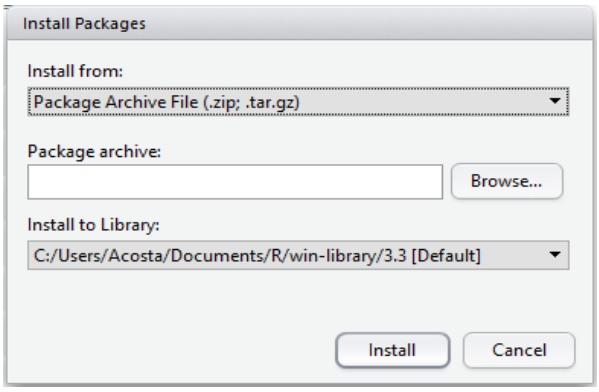

Figura 3.2 Ventana instalar paquete

Seleccionar la opción *Package Archive File (.zip;.tar.gz)* y luego navegar hasta donde se encuentre el paquete *Rsdf* como se observa en la Figura 3.3.

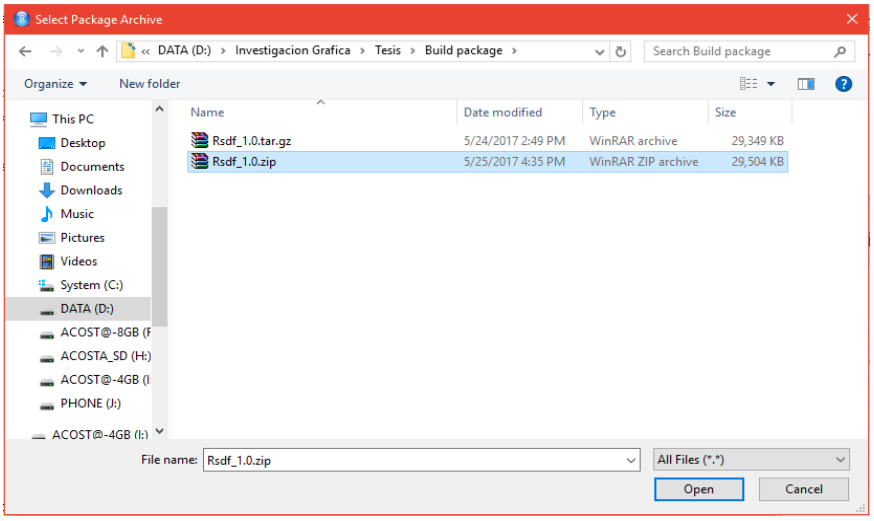

Figura 3.3 Ventana seleccionar paquete

Como se mencionó al inicio de epígrafe *Rsdf* para trabajar correctamente depende de otros paquetes que deben de estar instalados en nuestro sistema.

3.2 Caso de estudio: Datos globales del uso del suelo

En este epígrafe se hace uso de las funciones implementadas para elaborar un archivo *NetCDF* y *HDF* que permite que sus datos sean analizados y visualizados por sistemas de Información Geográfica como *gvSIG*.

3.2.1 Origen de los datos

Los datos utilizados para este caso de estudio han sido tomados por *Global Land Cover Facility* (GLCF), perteneciente a la Universidad de Maryland, Departamento de Geografía y con

colaboración de la NASA. El GLCF es un centro que se enfoca en la investigación de la dinámica de la cobertura terrestre, desarrollo y distribución de productos que explican aspectos y cambios en la cobertura terrestre. Los datos son provienen de Satélites teledirigidos de detección remota. Los productos desarrollados se centran en el cambio de acceso de sistemas locales de coberturas terrestres a sistemas globales (Anon n.d.). Los datos MODIS se encuentran disponibles de forma *on-line* y se pueden acceder desde la página oficial de GLCF o por vía ftp.

3.2.2 Características de los datos

El archivo **LC_5min_global_2012.tif** es un mapa global de uso del suelo, periodo 2001-2012. Este archivo se encuentra en http://glcf.umd.edu/data/lc/ y tiene un tamaño en disco de 7.3 MB. Posee 5 'x 5' de resolución que comprende 1776 filas x 4320 columnas para un total de 7672320 celdas con un tamaño de píxel geográfico de aproximadamente 0,083333º. Su extensión es de -180, 180, -64, 84 (xmin, xmax, ymin, ymax) y la proyección es +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0. Los valores como se muestra en la Tabla 3.1 van de 0 a 255 donde cada valor representa una codificación.

Valor	Etiqueta
0	Agua
1	Bosque de hoja perenne
2	Bosque de hoja ancha de hoja perenne
3	Deciduous Needleleaf forest
4	Bosque de hoja caduca de hoja caduca
5	Bosque mixto
6	Matorrales cerrados
7	Matorrales abiertos
8	Sabanas arboladas
9	Sabanas
10	Pastizales
11	Humedales permanentes
12	Tierras de cultivo
13	Urbano y urbanizado
14	Cropland / Mosaico de vegetación natural
15	Nieve y hielo
16	Estéril o escasamente vegetado
254	Sin clasificar
255	Valor de relleno

Tabla 3.1 Valores de codificación

La Figura 3.4 muestra el fichero **LC_5min_global_2012.tif**:

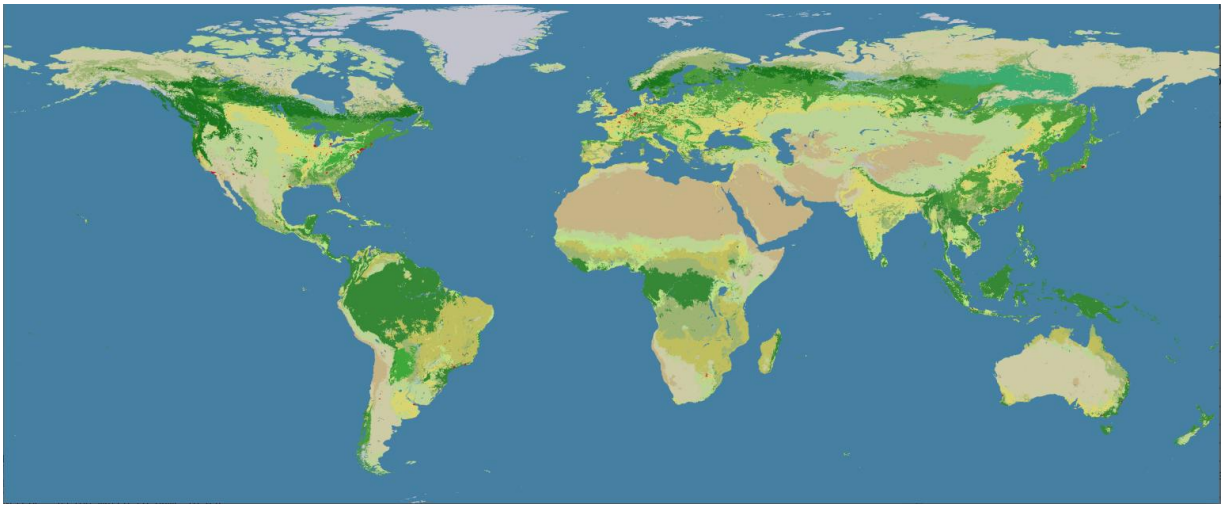

Figura 3.4 Mapa global de uso del suelo. Fuente: http://glcf.umd.edu/data/lc/

3.2.3 Ejecución de las funciones de Trasformación

Una vez que se han descrito los datos se utiliza la función *rasterToNetCDF, rasterToHDF* para transformar el raster a un archivo en formato *NetCDF* y *HDF* respectivamente. En la Figura 3.5 se observa la ejecución de la función *system.time* pasando como parámetro nuestras funciones, esta permite conocer el tiempo que se demora el algoritmo para transformar un *raster* a un *NetCDF*.

```
> system.time(rasterToNetCDF("data/LC_5min_global_2012.tif",ncfile = "LC_5min.nc"))
[1] "finish"
   user  system elapsed
   6.95    1.69    9.95
Warning message:
package 'ncdf4' was built under R version 3.3.2
> system.time(rasterToHDF("data/LC_5min_global_2012.tif",h5file = "LC_5min.h5"))
[1] "finish"
   user  system elapsed
   4.83    0.69    6.11
```

Figura 3.5 Ejecución de las funciones *rasterToNetCDF y rasterToHDF*

La ejecución del proceso se realizó en una computadora con las siguientes características: procesador Intel(R) Core(TM)2 Duo CPU E7300 @ 2.66GHz y memoria de sistema de 4096 MB. El tiempo de ejecución para transformar el *raster* a un *NetCDF* fue de aproximadamente 9.95 segundos como se observa en la Figura 3.5 y para convertir el *raster* a *HDF* transcurrió un tiempo de 6.11 segundos. Se aprecia como buenos resultados si se tiene en cuenta la cantidad de información procesada. En cuanto a la memoria ocupada en disco duro se redujo

considerablemente en el nuevo archivo *HDF* de 7.3 MB a 515.2 KB como se aprecia en la Figura 3.6, no sucedió igual con el fichero *NetCDF* que paso a aumentar su tamaño en 29.3 MB.

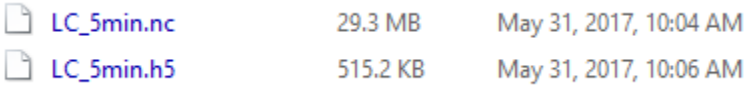

Figura 3.6 Archivos generados a partir del raster LC_5min_global_2012.tif

La Figura 3.7 muestra la estructura de cada fichero creado, así como los datos almacenados y una comparación entre ellos. Se puede observar que estructura de ambos archivos difieren, pero los datos almacenados son los mismos. El nombre del *dataset* que contiene los datos almacenos es el mismo que el nombre de la capa del *raster*, en caso de ser un *raster* multicapa se crea un *dataset* por cada capa. Como los archivos resultantes provienen del *raster* LC_5min_global_2012 los *datasets* creados poseen dos dimensiones que comprende 1776 filas x 4320 columnas y en el caso del *NetCDF* cuenta con dos variables más que representan la dimensiones longitud (4320) y latitud (1776).

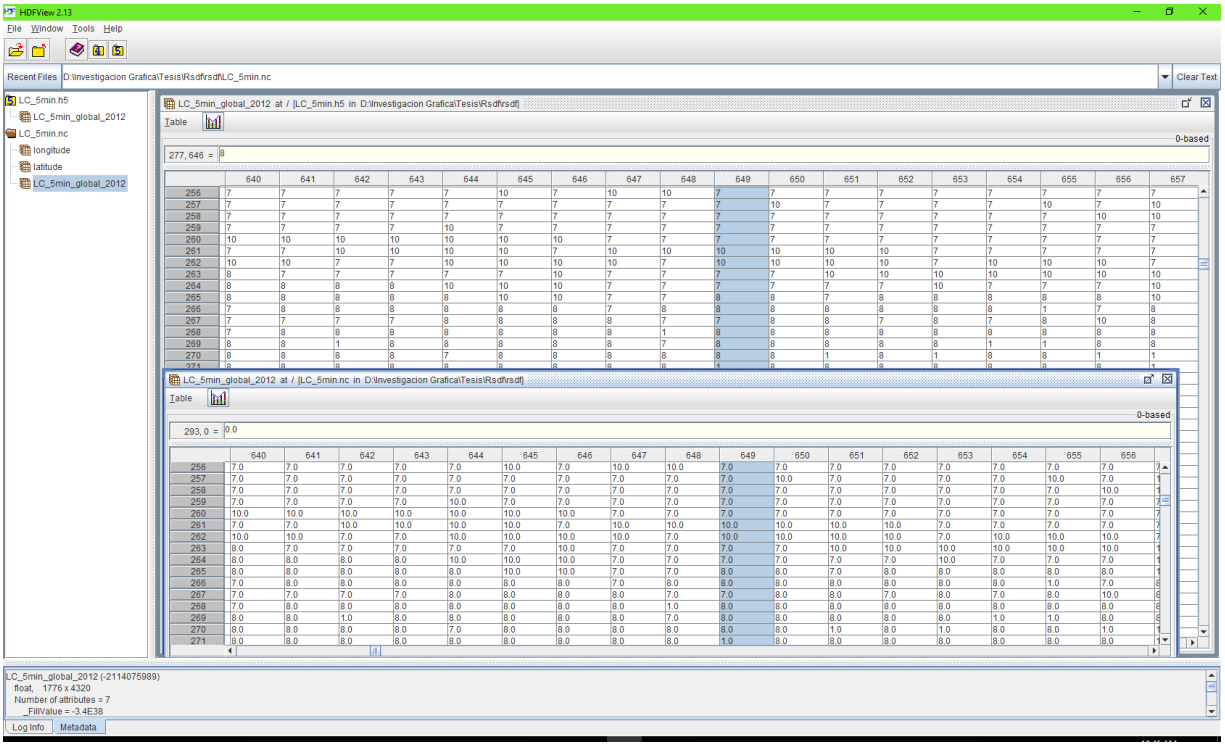

Figura 3.7 Vista de los archivos generados a partir del *raster* LC_5min_global_2012.tif

3.3 Uso de la herramienta desarrollada

El paquete *Rsdf* permite transformar de un formato de datos científicos a otro, lo que facilita a usuarios de estos formatos obtener datos de fuentes ajenas a la institución donde trabajan.

3.3.1 Uso de algunas funciones del paquete *Rsdf*

Como se muestra en la Tabla 2.1 *Rsdf* cuenta con 38 funciones, de las cuales se realiza pruebas a varias de ellas. Para las pruebas se utiliza el archivo cru_ts3.21.1901.1910.tmn.dat.nc tomado del instituto *Climatic Research Unit* (CRU) de la universidad del Este de Anglia en Reino Unido y el archivo 3B-HHR.MS.MRG.3IMERG.20141001-S090000-E092959.0540.V03D.HDF5 tomado de la NASA.

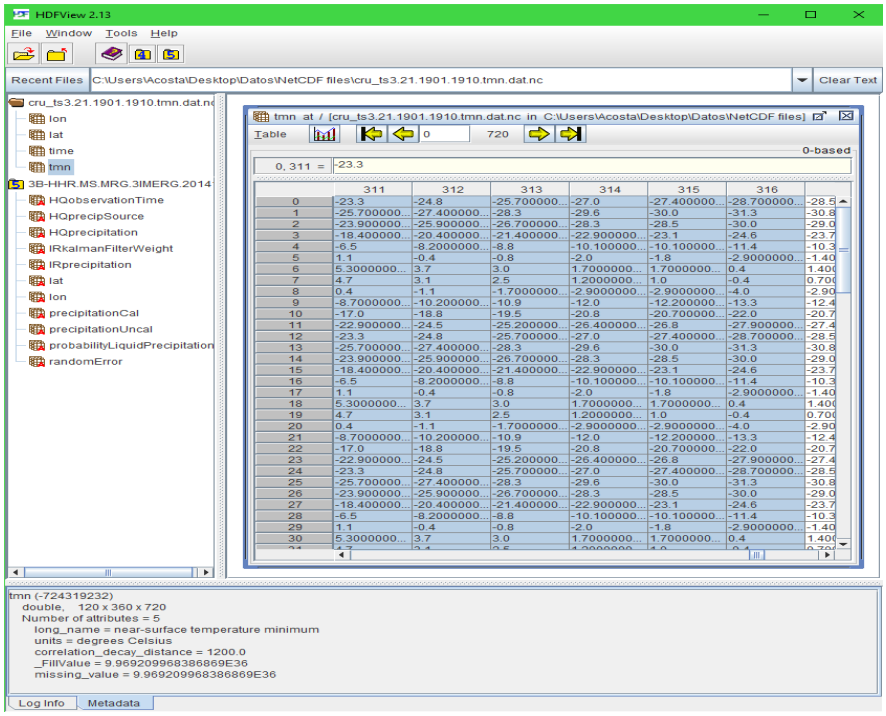

Figura 3.8 Estructura de los archivos de prueba

En la Figura 3.8 se muestra la estructura general del archivo cru_ts3.21.1901.1910.tmn.dat.nc que cuenta con tres dimensiones (longitud, latitud y tiempo) y la variable *tmn* que representa la temperatura mínima en grados Celsius.

Al aplicar la función `NetCDFtoHDF("data/cru_ts3.21.1901.1910.tmn.dat.nc","CRU.`

`TS3.tmn.h5")` se transforma el archivo *NetCDF* a un *HDF* que mantiene la misma estructura que el original. En la Figura 3.9 se muestra una comparación entre el archivo de entrada y el fichero resultante de aplicar la transformación.

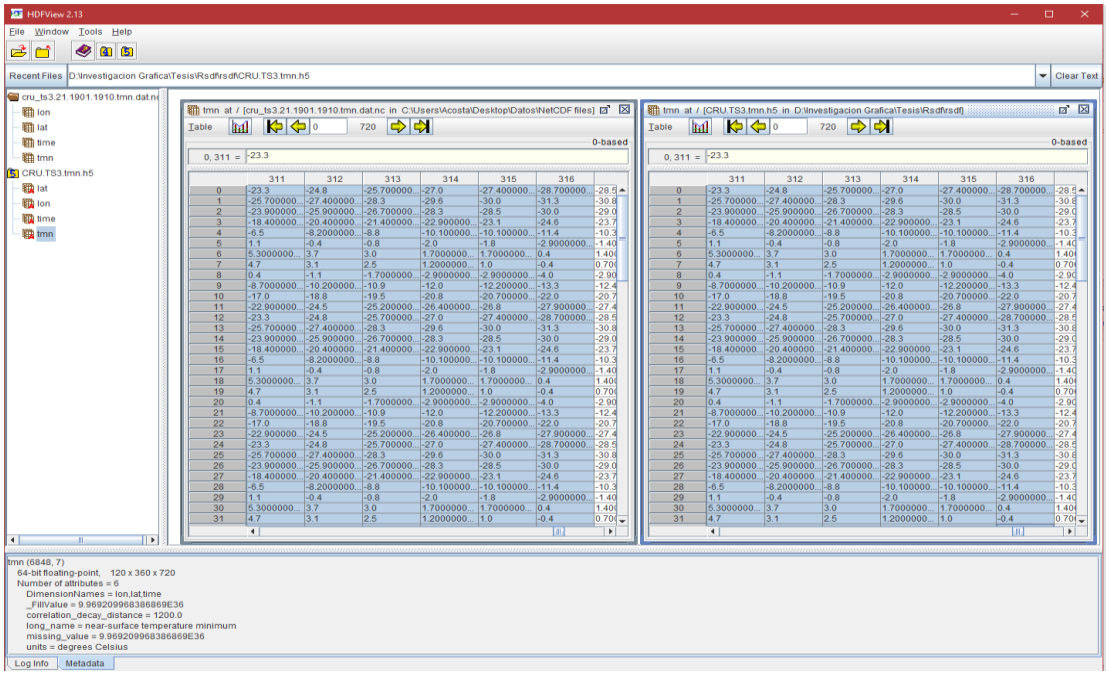

Figura 3.9 Comparación del *NetCDF* de entrada y el *HDF* de salida

A continuación, se transforma el *NetCDF* a un archivo en formato raster. Después de aplicar la función:`NetCDFtoRaster("data/cru_ts3.21.1901.1910.tmn.dat.nc","CRUraster.tmn.tif", format = 'GTiff')` se aprecia que no ocurrió perdida al transformarse los datos a la estructura del fichero raster. Como se mencionó anteriormente el archivo *NetCDF* posee tres dimensiones que son la latitud, longitud y el tiempo en que varían los datos de la variable *tmn*. La Figura 3.10 muestra la estructura del nuevo archivo *raster* obtenido a partir del proceso de transformación. Se transformaron los 120 instantes de tiempos a 120 bandas que contienen los datos representados por las dimensiones latitud y longitud.

```
> raster("CRUraster.tmn.tif")
class       : RasterLayer
band        : 1  (of  120  bands)
dimensions  : 360, 720, 259200  (nrow, ncol, ncell)
resolution  : 0.5, 0.5  (x, y)
extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
coord. ref. : NA
data source : D:\Investigacion Grafica\Tesis\Rsdf\rsdf\CRUraster.tmn.tif
names       : CRUraster.tmn
values      : -59.1, 27.2  (min, max)
```

Figura 3.10 Estructura del archivo *raster* obtenido de aplicar la función de transformación

Los datos de la variable *tmn* se pueden apreciar mediante la visualización del nuevo *raster* creado como se muestra en la Figura 3.11.

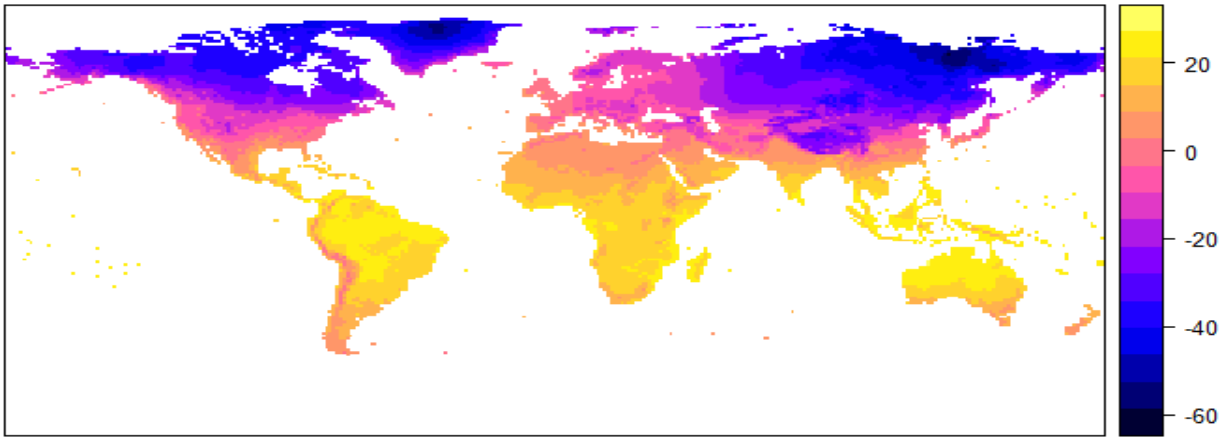

Figura 3.11 Visualización del nuevo *raster* creado

El archivo HHR.MS.MRG.3IMERG.20141001-S090000-E092959.0540.V03D.HDF5 cuenta con una estructura de dos dimensiones (lat y lon) y 9 variables que representan las precipitaciones como se muestra en la Figura 3.8.

Al aplicar la función `HDFtoNetCDF("HHR.MS.MRG.3IMERG.20141001-S090000-E092959.0540.V03D.HDF5","HHR.nc")` se transforma el archivo *HDF* a un *NetCDF* que mantiene la misma estructura que el original. En la Figura 3.12 se muestra una comparación entre el archivo de entrada y el fichero resultante de aplicar la transformación. No existe perdida de datos después del proceso de transformación.

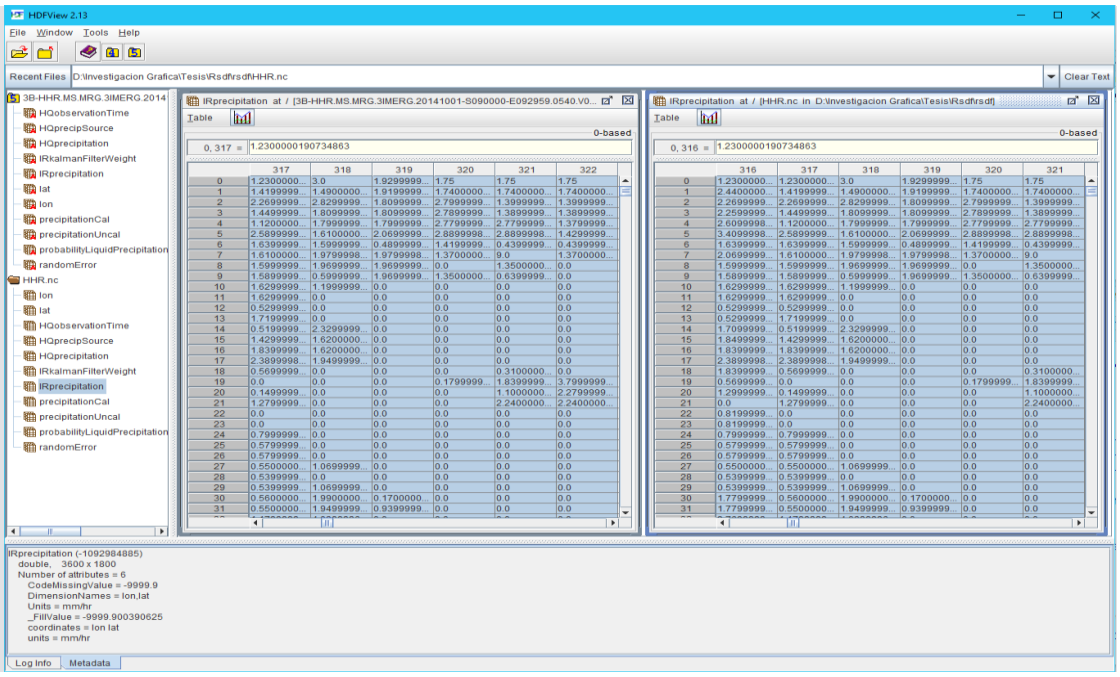

Figura 3.12 Comparación del *HDF* de entrada y el *NetCDF* de salida

3.4 Conclusiones parciales.

A lo largo de este capítulo se han probado diferentes algoritmos en la realización del caso de estudio planteado, corroborando de esta manera la eficiencia y exactitud de los algoritmos utilizados. También ejemplifica cual puede ser el uso y la utilidad del nuevo módulo realizado. Además, todos los archivos creados pueden ser objeto de estudio de entes especializados en el comportamiento de los suelos.

4 CONCLUSIONES

Como resultado de esta investigación se desarrolló un paquete en *R* para la manipulación de grandes volúmenes de series de datos espacio-temporales y geoespaciales, cumpliéndose el objetivo general planteado y una serie de tareas que se desglosan a continuación:

- Se realizó un profundo análisis sobre los principales formatos de datos científicos para series espacio-temporales utilizados en el área de las ciencias geográficas, se describieron y mostraron algunas de sus propiedades más importantes y se seleccionaron *HDF* y *NetCDF* como los más adecuados para la representación de las series espacio-temporales. Para los datos geoespaciales se selecciona los modelos *raster* y *vectorial* al ser los más usados por las comunidades SIG.
- Se desarrolló el paquete *Rsdf* capaz de manipular eficientemente series de datos espacio-temporales y geoespaciales en diversos formatos de datos científicos, se mostró la arquitectura general del mismo, así como los pasos a seguir para la incorporación de una nueva extensión (paquete) en R.
- Se presentó un caso de estudio donde se transformaron datos del comportamiento de los suelos, publicados por la Universidad de Maryland, donde se evidencio la utilidad de los algoritmos desarrollados para la manipulación de grandes volúmenes de datos espacio-temporales y geoespaciales.

5 RECOMENDACIONES

Luego de concluir esta investigación, solo queda hacer algunas recomendaciones, tales como:

- Implementar nuevos algoritmos que amplíen las funcionalidades de la herramienta.
- Incorporar a *Rsdf* nuevos formatos de datos científicos para series espacio-temporales y geoespaciales.
- Probar la efectividad de la herramienta con varios conjuntos de datos en otro dominio de aplicación.

6 REFERENCIAS BIBLIOGRÁFICAS

Annau, M., 2016. Package “h5.” , p.19.

Anon, GLCF: Welcome. Available at: http://www.landcover.org/ [Accessed May 26, 2017].

Bivand, R. et al., 2017. full-text.

Borrell González, A., 2012. Aplicación informática para la integración de los datos generados por el observatorio OBSEA en las redes de sistemes de observación.

Chambers, J.M. & Hastie, T.J., 1991. *Statistical models in S*, CRC Press, Inc.

Chang, K.-T., 2008. *Introduction to Geographic Information Systems*, Available at: http://courses.washington.edu/gis250/lessons/introduction_gis/ [Accessed March 23, 2017].

Durango, C., 2013. Caracterización de Datos Espacio - Temporales en Sistemas de Información Geográfica.

Goodchild, M.F., 2009. Geographic information systems and science: today and tomorrow. *Procedia Earth and Planetary Science*, 1(1), pp.1037–1043. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1878522009001611 [Accessed March 14, 2017].

Greenberg, J.A., Perpinan, O. & Be-, A., 2014. Package “ raster .”

Group, H.D.F. & others, 2011. HDF5 User’s Guide.

Hankin, S.C. et al., 2010. NetCDF-CF-OPeNDAP: Standards for ocean data interoperability and object lessons for community data standards processes. In *Oceanobs 2009, Venice Convention Centre, 21-25 septembre 2009, Venise*.

Hijmans, R.J., 2013. Introduction to the ’ raster ’ package (version 2 . 1-49). , (2008), pp.1–27.

IBÁÑEZ, A.M. & HOEHNE, A.V., 2010. Diseño de primitivas geométricas espacio-temporales para describir fenómenos dinámicos. *Geofocus.Rediris.Es*, pp.232–251. Available at: http://geofocus.rediris.es/2010/Articulo10_2010.pdf.

Ihaka, R., 1998. R: Past and future history. *Computing Science and Statistics*, pp.392–396.

Iwamura, K. et al., 2011. 4D-GIS (4 dimensional GIS) as spatial-temporal data mining platform and its application to managementand monitoring of large-scale infrastructures. In *Proceedings 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services*. IEEE, pp. 38–43. Available at: http://ieeexplore.ieee.org/document/5969001/ [Accessed March 14, 2017].

Levinsohn, A., 2001. LA INTEROPERABILIDAD GEOESPACIAL: EL SANTO GRIAL DEL CAMPO SIG. *Mapping*. Available at: http://redgeomatica.rediris.es/metadatos/publica/articulo04.htm.

Long, Q. et al., 2013. JAWAMix5: an out-of-core HDF5-based java implementation of whole-genome association studies using mixed models. *Bioinformatics*, 29(9), pp.1220–1222.

Longley, P., Maguire, D., Goodchild, M., Rhind, D., 2005. *Geographical Information Systems and Science*, Wiley. Available at: https://books.google.com.cu/books?hl=es&lr=&id=toobg6OwFPEC&oi=fnd&pg=PR9&dq=Geographic+information+systems+&+science.+third.+Hoboken&ots=yj-ivPEitm&sig=Ui_7F7JdwBgZETPG5J5NzubfP_I&redir_esc=y#v=onepage&q=Geographic information systems & science. thir [Accessed March 15, 2017].

McGrath, R.E., 2003. XML and Scientific File Formats. In *2003 Seattle Annual Meeting*.

Michna, A.P., 2016. Package " RNetCDF ."

Mirai, S.G., 2016. Package "XLConnect" : Excel Connector for R. Available at: https://cran.r-project.org/web/packages/XLConnect/XLConnect.pdf.

NASA, 2013. CDF User's Guide.

Olaya, V., 2014. Sistemas de Información Geográfica. *Journal of Chemical Information and Modeling*, 53(9), pp.1689–1699.

Pebesma, E., Michael, S. & Robert, H., 2013. Package " rgdal ." , 01.

Poinot, M., 2010. Five good reasons to use the hierarchical data format. *Computing in Science*

& Engineering, 12(5), pp.84–90.

Ravishankar, M.S., 2008. HDF_NetCDF_Report. , pp.1–7. Available at: papers2://publication/uuid/362F8A2A-33B7-419A-A584-B1C0459F951A.

Rew, R. et al., 2010. The NetCDF users guide-data model, programming interfaces, and format for self-describing, portable data-NetCDF version 4.1. *Unidata Program Center*.

Roque, D.C., 2014. Herramientas para la manipulación de formatos de datos científicos espacio-temporales en SIGs.

Salmerón Gómez, R., 2008. *Análisis estadístico de datos espacio-temporales mediante modelos funcionales de series temporales*, Editorial de la Universidad de Granada. Available at: https://dialnet.unirioja.es/servlet/tesis?codigo=21049 [Accessed March 14, 2017].

Santana, J. & Farfán, E., 2014. El Arte de programar en R: un lenguaje para la estadística. , p.182.

Sarria, F.A., 2006. Modelos y estructuras de datos. *Sistemas de Información Geográfica de la licenciatura de Ciencias Ambientales e Introducción a los Sistemas de Información Geográfica de la licenciatura de Geografía*, (1), pp.53–70. Available at: http://www.um.es/geograf/sigmur/sigpdf/temario_3.pdf.

Soto-Durán, D.E., Marín-Morales, M.I. & Vargas-Agudelo, F.A., 2014. CARACTERIZACIÓN DE FORMATOS DE ALMACENAMIENTO, TRANSPORTE Y VISUALIZACIÓN DE DATOS GEOGRÁFICOS. , pp.23–33.

Stadler, M.M., 2002. ¿Qué Es La Topología ? *Documento Web*, pp.63–78. Available at: http://personales.ya.com/casanchi/mat/topologia01.htm\nhttp://doqpelganger.wordpress.com/2010/12/09/topologia-la-ciencia-de-los-topos/.

Team, R., 2015. RStudio: integrated development for R. *RStudio, Inc., Boston, MA URL http://www. rstudio. com*.

Team, R.C.D.C., 2007. Writing R Extensions. *Development*, 0, p.401.

Ullman, R. & Denning, M., 2012. HDF5 for NPP sensor and environmental data records. In *2012 IEEE International Geoscience and Remote Sensing Symposium*. pp. 1100–1103.

UNIDATA, 2017. UNIDATA. Available at: http://www.unidata.ucar.edu.

Yang, W. & Di, L., 2004. An accurate and automated approach to georectification of HDF-EOS swath data. *Photogrammetric Engineering & Remote Sensing*, 70(4), pp.397–404.

Yeh, P., 2002. Implementation of CCSDS lossless data compression for space and data archive applications. In *SpaceOps 2002 Conference*. p. 12.