Implementación de un Índice de Posibilidad de Incendio para las Sierras de Córdoba.

Por Luis Gonzalo Quiroga.

Presentado ante la Facultad de Matemática, Astronomía y Física y el Instituto de Altos Estudios Espaciales Mario Gulich como parte de los requerimientos para la obtención del grado de

MAGISTER EN APLICACIONES ESPACIALES DE ALERTA Y RESPUESTA TEMPRANA A EMERGENCIAS

UNIVERSIDAD NACIONAL DE CÓRDOBA

Abril, 2015

Director: Lanfri, Sofía. Codirector: Cecilia Estrabou.

Resumen

Los incendios forestales son uno de los desastres naturales más destructivos, representando un peligro para la conservación de la naturaleza, la preservación de vidas humanas y recursos económicos. El ciclo climático de la provincia de Córdoba se distingue por presentar gran precipitación estival, que lleva a un gran crecimiento de plantas, y una posterior sequía en la época invernal, transformado la vegetación en combustible. Esa característica junto a estaciones anuales más marcadas, como producto del cambio climático y un aumento de la población en zonas rurales, conlleva a un incremento en la frecuencia de incendios forestales. Debido a esto, en el presente trabajo se plantea la implementación de un índice de posibilidad de incendio de manera automatizada para la zona central de Argentina (Sierras de Córdoba), como una herramienta para evitar potenciales daños ecológicos y económicos producidos por el fuego. El lenguaje de programación elegido es Python por su robustez y que es fácil de aprender. Además se pretende mejorar el índice adjuntando factores de riesgo antrópicos que pueden afectar la frecuencia de los incendios forestales. Como resultado se obtuvo un software que está compuesto por siete sub-programas que pueden funcionar en conjunto o independientemente. Los archivos de salida de dicho software además del mapa de posibilidad de incendio, incluyen los mapas temáticos de los sub-procesos necesarios para llegar al resultado final.

Palabras clave: fuego, incendio forestal, bosque, índice, teledetección.

Abstract

Forest fires are one of the most destructive natural disasters, representing a threat to nature conservation, preservation of human lives, and economic resources. The climate cycle in Córdoba province is distinguished by, a summer rainy season, leading to high growth of plants, and a subsequent drought in the winter, thus vegetation is transformed into fuel. That feature alongside more marked annual seasons - as a result of climate change - and population growth in rural areas, leads to an increase in the frequency of forest fires. This study intends to implement an automated fire probability index for the central part of Argentina (Córdoba Sierras), which can be used as a tool to avoid potential ecological and financial damage. The programing software used is Python, and it was chosen due to its robustness and easy access. Besides, we intend to improve this rate by adding anthropogenic risk factors that can affect the forest fire frequency. As a result, we obtained a software program made up of seven sub-programs that can work together or independently. This software output files offer a map with fire probability index, along with thematic maps of sub-processes necessary to arrive at the final result.

Keywords: fire, wildfire, forest, index, remote sensing.

Resumen 2

Abstract 3

Capítulo I: Introducción 3

I.I Introducción 3

I.II Objetivo 5

I.III Estructura del trabajo 5

Capítulo II: El Índice 6

II.I Origen del Índice 6

II.II Conformación del índice. 7

Capítulo III: Reflectancia de superficie 13

III.I Disponibilidad y Selección de sensores. 13

III.II Características de los productos MODIS utilizados 14

III.III Procesamiento de productos MODIS 14

Capítulo IV: Características Del Área de Estudio 17

IV.I Ambiente y Geografía física 17

IV.II Combustibles 18

IV.III Zonas de amortiguación 19

Capítulo V: Correcciones topográficas de la evapotranspiración 21

V.I Información meteorológica 21

V.II Evapotranspiración 21

V.III Cálculos de la iluminación sobre planos inclinados. 23

Capítulo VI: Construcción del Software 26

VI.I Elección del lenguaje de programación 26

VI.II Requerimientos del programa y ejecución. 27

Capítulo VII: Resultados y Conclusiones 30

VII.I Resultados en mapas 30

VII.II Conclusión 35

VII.III Consideraciones finales. 36

Bibliografía 37

Anexo I 40

Etapa 0 40

Etapa I 43

Etapa II 45
Etapa III 49
Etapa IV 57
Etapa V 59

Capítulo I: Introducción

I.I Introducción

En las últimas décadas, la ocurrencia repetida de incendios forestales de gran magnitud en varias partes del planeta (con 2.9 millones de hectáreas afectadas entre 1986 y 2004 en Sudamérica solamente) ha remarcado la necesidad de desarrollar herramientas efectivas de monitoreo para evaluar y mitigar este problema (Ceccato *et al.* 2002a;b, FAO 2006).

En una gran variedad de ecosistemas, los incendios forestales son un fuerte factor de transformación del ambiente y en algunos casos son eventos fundamentales para el funcionamiento del sistema (Anderson 1981, Pucheta *et al*. 1997, Reich *et al.* 2001); sin embargo, cuando los incendios forestales aumenta su ocurrencia o intensidad, las modificaciones que producen pueden traer como consecuencia la pérdida de servicios ecológicos que brindan los bosques afectando de forma drástica la calidad de vida del hombre. Sus efectos negativos incluyen la pérdida de recursos forestales, paisajísticos, alimentarios, la desaparición de la biodiversidad asociada, pérdida de vidas y de bienes (Martínez *et al*. 2009, Gavier *et al.* 2004). Esta pérdida de bosques mediada por fuego también se suma a la deforestación, causada por el avance de la frontera agrícola y urbanización, produciendo cambios en el ambiente físico, aumento de erosión, pérdida de suelo y disminución de retención de agua (Renison *et al*. 2002).

Los incendios tienen tanto impactos locales como globales, afectando inmensas áreas, liberando gases de invernadero a la atmósfera, modificando el ciclado de nutrientes y los regímenes hidrológicos (Chuvieco *et al*. 2010).

En las Sierras de Córdoba, Argentina, el fuego fue una herramienta de manejo importante que se usaba asociada a las actividades ganaderas, presentes en la zona a partir del siglo XVII (Díaz *et al*. 1994). Si bien esta práctica está hoy prohibida los habitantes locales continúan utilizando esta herramienta. Los fuegos de origen natural, en cambio, son virtualmente escasos (Miglietta 1994), pero cuando ocurren, son causados generalmente por rayos sobre todo en la región denominada “Sierras Grandes” (Comunicación personal con Guardaparques del Parque Nacional Quebrada del Condorito) .

El aumento del calentamiento global, produce cambios en los regímenes climáticos, lo cual conlleva a un incremento en la ocurrencia de incendios forestales y con mayor severidad, ya que la época del año en que ocurre un incendio podría afectar la intensidad que tiene el mismo sobre la superficie y sobre las copas. El estado fenológico (estación-dependiente) de las plantas quemadas determinara la respuesta reproductiva y lo cual tiene un efecto

pronunciado en la recuperación de los ecosistemas post fuego (Flannigana *et al.* 2000). Por ello se tendría que destinar más recursos y mejores herramientas para la prevención de incendios, logrando la protección de recursos los naturales y humanos. La obtención de Información abocada a la amenaza de incendio, permitirá un adecuado manejo de medios para planificar eficazmente y tomar medidas preventivas (reducción de niveles de combustible, respuesta temprana a incendios) en ambientes propensos al fuego donde valiosos recursos seguramente serían dañados (Loboda *et al.* 2007).

El desarrollo de mapas permite mostrar el riesgo de incendio basado en observaciones de que existe una fuerte relación entre el fuego, las características del combustible (tipo de vegetación, densidad, contenido de humedad), la topografía (pendiente, altitud, exposición solar) y las condiciones meteorológicas (lluvia, dirección y velocidad del viento, humedad del aire, temperatura de la superficie y del aire). Estos parámetros impactan directamente en la propensión de un área dada a incendiarse y propagar el fuego. Debido a que estas cantidades pueden ser medidas, sin importar la causa de ignición del incendio, el comportamiento del mismo puede ser considerado estrictamente dependiente de estos parámetros y por lo tanto puede ser previsto cuando son conocidos (Laneve *et al*. 2011).

Por lo tanto generar mapas de riesgo de incendio es clave en la prevención, ya que el planeamiento pre-fuego necesita herramientas operativas para monitorear cuándo y dónde un incendio ocurrirá, o cuándo los daños potenciales serán mayores de los normales (Loboda *et al.* 2007, Chuvieco *et al*. 2010).

Uno de los principales factores para el desarrollo de un incendio es el estado de la vegetación (contenido de agua, verdor, distribución), que pueden ser medidas a campo usando diversas técnicas, pero dichos datos sólo representan condiciones específicas locales limitadas a un momento dado en el tiempo y el espacio.

El censado remoto junto con los sistemas de información geográfica son importantes componentes para la realización de mapas y la disminución de la vulnerabilidad. La disponibilidad actual de datos satelitales de mediana y alta resolución espacial permite generar herramientas con nuevos métodos para la extracción de la información (Mallinis *et al*. 2008). Datos obtenidos mediante teledetección proveen información a escalas locales, regionales y globales. Pueden también generar información de modo secuencial en el tiempo (por ejemplo cada 30 minutos, todos los días, o cada varios días), cuya recopilación a campo sería excesivamente costosa, esta información con una alta tasa de actualización permite una mayor exactitud en los valores obtenidos para realizar índices de peligro de incendio (Ceccato 2001a;b).

Se han propuesto muchos índices de amenaza de incendio, como por ejemplo, el Índice Portugués (INMG. 1988; Gonçalves *et al.* 1990), el método ICONA (ICONA, 1993), el *Canadian Fire Weather Index* (Van Wagner *et al.* 1985) y el *Fire Potential Index* (FPI) (Burgan *et al*. 1998; Burgan *et al.* 1993; San-Miguel-Ayanz *et al*. 2003).

Excepto en el FPI, estos índices no utilizan datos obtenidos de imágenes satelitales. En 2011

Laneve *et al.* desarrollan un FPI modificado (MFPI) en donde se agrega un nuevo paso en el cálculo del índice. Dicho índice considera los efectos de la iluminación solar en el terreno sobre la evapotranspiración, lo cual permite (sin afectar su eficiencia en zonas planas) obtener resultados que se ajustan más a la realidad de las áreas ubicadas en zonas de montaña, donde la radiación solar recibida puede variar ampliamente entre laderas, dependiendo de su pendiente y orientación.

I.II Objetivo

Dada la frecuente ocurrencia de incendios forestales y las grandes pérdidas, tanto ecológicas como económicas que se producen en la provincia de Córdoba, en el presente trabajo se pretende realizar una implementación a nivel local del trabajo realizado por Laneve *et al.* en 2011, ajustando su técnica a las características locales. A su vez se propone añadir al índice modificado de posibilidad de incendio la presencia de asentamientos humanos (pueblos y ciudades), cuerpos de agua y redes viales como factores de riesgo.

I.III Estructura del trabajo

El presente trabajo está desarrollado en siete capítulos. Cada uno trata una temática distinta de todas las que intervienen en la construcción de un índice de posibilidad de incendio. En el capítulo II se desarrolla el origen del índice y cuáles son los datos de entrada y procesos necesarios para obtenerlo. A continuación en el capítulo III se exponen el por qué de la selección del sensor utilizado para obtener los datos de reflectancia de la vegetación y sus características.

En el capítulo IV se desarrolla brevemente la descripción del ambiente en la zona de estudio, para posteriormente ahondar sobre las capas de combustibles y zonas de amortiguación. Luego en el capítulo V se explica como se obtiene la evapotranspiración y la importancia de corregir la cantidad de energía incidente en zonas de montaña.

La elección del lenguaje de programación, funcionamiento y requerimientos del programa que genera el índice automáticamente se explica en el capítulo VI. Finalmente en el capítulo VII se expone la conclusión y consideraciones a futuro.

Capítulo II: El Índice

II.I Origen del Índice

El índice de posibilidad de incendio (FPI en inglés) tiene su origen como parte del sistema de prevención y respuesta a incendios forestales dentro del servicio forestal estadounidense (USFS por sus siglas en inglés). Fue presentado en el año 1998 por Burgan *et al.* para resolver la falta de un sistema que proveyera rápidamente información a los encargados de tomar decisiones para el manejo de incendios, decisiones que en el caso de comenzado el incendio deben tomarse en una escala temporal de 24 horas o menos. Hasta aquel entonces solo se disponía de información obtenida a campo, siendo sumamente puntual, de gran costo y de actualización lenta, resumida en el sistema nacional de nivel de peligro de incendio (*U.S. National Fire Danger Rating System)*.

El modelo del índice de posibilidad de incendio fue desarrollado teniendo en cuenta incorporar, tanto información provista por sensores remotos como observaciones a campo. La resolución espacial del índice en el momento de su desarrollo fue de 1 km.

Su formula es:

$$FPI = (1 - TN_f) \cdot (1 - LR) \cdot 100$$

donde TN_f es el porcentaje del tiempo de retraso de ignición a 10 horas y LR es la fracción de verdor relativo. Dicha formula será desarrollada específicamente en la sección II.II.

Los supuestos del modelo FPI son: 1) La posibilidad de ocurrencia de un incendio puede ser determinada si, el nivel de humedad en la vegetación viva y muerta puede ser estimado fehacientemente, 2) El verdor de la vegetación provee un estimador útil de la vegetación viva con gran contenido de humedad, 3) La humedad de combustible para un tiempo de retraso de ignición a 10 horas (*10 hour timelag fuel moisture*) debe ser usada para representar la vegetación muerta, ya que el contenido de humedad en combustible de pequeño tamaño es crítico para la propagación de un incendio y 4) la información sobre condiciones de viento no son incluidas por su existencia transitoria.

Los datos de entrada para el FPI constan en un mapa de modelos de combustibles de resolución de 1 km, un mapa de verdor relativo (Burgan *et al.* 1993), que indica el estado actual de verdor de la vegetación comparado con los valores históricos máximos y mínimos de verdor y un mapa de humedad de combustible muerto de 10 horas de retraso de ignición (fm10hs) (Fosberg *et al.* 1971). Los combustibles del fm10hs están definidos por piezas de

vegetación leñosa de entre 0,6 y 2,5 cm de diámetro. Todos estos datos de entrada deben estar en formato imagen de mapa de bits con resolución de 1 km por pixel. El dato de salida es el FPI con valores que varían entre 1 y 100, siendo las cifras más altas las que indican mayor posibilidad de incendio.

Originalmente el FPI estima correctamente (es decir, que no sobreestima o subestima) los valores de posibilidad de incendio en un área dada mientras esta sea una zona plana, en zonas de montaña estas estimaciones se ven afectadas por las diferencias de cantidad de energía solar recibida que existe en las laderas, según su pendiente y orientación (Burgan *et al.* 1998, Laneve *et al.* 2011).

En 2011 Laneve *et al.* desarrollan una versión modificada del FPI, el MFPI (*Modified Fire Potential Index*), en el cual posteriormente al cálculo y clasificación (en cinco categorías) del FPI, se realiza una reclasificación de cada categoría teniendo en cuenta la evapotranspiración del terreno. Con el agregado de esta nueva variable se solucionan los defectos de sobre o subestimación del FPI en zonas montañosas. Además de la corrección topográfica se agrega una mejora en la resolución del índice a 250 m.

El cálculo de la evapotranspiración se realiza siguiendo la fórmula de Penman-Monteith modificada por FAO (Allen *et al*.,2006). En la etapa del cálculo de radiación solar total incidente se modifica este valor con los cálculos específicos para obtener la radiación total incidente en un plano inclinado (Se hablará más extensamente de este paso en el capítulo V).

II.II Conformación del índice.

Basado en el FPI original y específicamente en la sección de corrección topográfica del MFPI, se realizó el mapa de índice de riesgo de incendio. La resolución espacial de salida elegida para el mapa es de aproximadamente 250 m (dicha elección se explica en el capítulo III) y se requieren los siguientes datos de entrada: imágenes diarias de reflectancia de superficie, mapa de cobertura de combustible, Modelo de Elevación Digital (DEM), imágenes diarias de predicción de humedad relativa, temperatura máxima, temperatura mínima y velocidad del viento. A estos datos se agregan las variables de distancia a caminos, ciudades y cursos de agua, como fuentes atenuantes o agravantes según el caso, constituyendo así el Índice de Riesgo de Incendio (IRI).

El cálculo del IRI puede subdividirse en seis secciones: información del estado de la vegetación, contenido de humedad, cálculo del FPI, cálculo de evapotranspiración y corrección del FPI, determinación de zonas de riesgo e índice final.

La información del estado de la vegetación se genera a partir de las imágenes de reflectancia de superficie. Como primera medida es necesario generar los valores históricos de Índice de Vegetación de Diferencia Normalizada (NDVI), es decir, el valor máximo y mínimo posible de NDVI para cada pixel del área de estudio. Con los valores históricos de NDVI y el valor de

NDVI correspondiente al día de emisión del mapa, se obtiene el índice de verdor relativo (RG). Para calcular el NDVI se utiliza la fórmula propuesta por Tucker (1979)

$$NDVI = \frac{Infrarrojo\ Cercano - Rojo\ Visible}{Infrarrojo\ Cercano + Rojo\ Visible}$$

y para calcular el verdor relativo se utiliza la fórmula descripta por Burgan (1998)

$$RG = \frac{NDVI_0 - NDVI_{mn}}{NDVI_{mx} - NDVI_{mn}} \cdot 100$$

donde $NDVI_0$ es el valor del NDVI para el día de cálculo del índice, $NDVI_{mn}$ es el valor mínimo histórico y $NDVI_{mx}$ es el valor máximo histórico y RG es verdor relativo que está expresado en una escala de 0 a 100.

Para estimar el contenido de humedad de la vegetación/combustible se calcula el fm10hs a partir de datos de humedad relativa y temperatura, utilizando la formula descripta por Laneve

$$fm10hs = 1{,}28 \cdot emc$$

(2011)

donde emc se computa mediante la fórmula empírica

$$emc = 0{,}03229 + 0{,}281073 \cdot h - 0{,}000578 \cdot h \cdot T \qquad si\ h < 10\%$$

$$emc = 2{,}22749 + 0{,}160107 \cdot h - 0{,}014784 \cdot T \qquad si\ 10\% \leq h \leq 50\%$$

$$emc = 21{,}0606 + 0{,}005565 \cdot h^2 - 0{,}00035 \cdot h \cdot T - 0{,}483199 \cdot h \qquad si\ h > 50\%$$

donde h es igual a la humedad relativa y T es igual a la temperatura.

Una vez calculado el fm10hs, cuyos valores están expresados en porcentaje, se obtiene la versión fraccional del fm10hs (TN_f) a partir de su normalización utilizando los valores de la humedad de extinción del combustible muerto (MX_d), utilizando la siguiente fórmula (Burgan, 1998),

$$TN_f = \frac{fm10hs - 2}{MX_d - 2}$$

El valor MX_d es un dato preexistente para cada tipo de combustible definido por Scott *et al.* (2005). En la Tabla 1 se muestran las equivalencias elegidas entre los modelos propuestos por Scott *et al.* (2005) y las capas de combustible determinadas. Con estas equivalencias se obtiene el MX_d más proximo a cada tipo de cobertura local. La elección del modelo equivalente se realizó teniendo en cuenta las características ecológicas de cada tipo de combustible.

Tabla 1

Modelo de Scott *et al.* (2005)	Cobertura de combustible
91	Urbano
98	Cuerpos de Agua
99	Salinas / Roca Desnuda
102	Cultivos
103	Vegetación Inundable
104	Pastizal de Llanura
105	Pastizal de Altura
123	Pastizal y Bosque de Altura
141	Vegetación Halófita
147	Romerillal
165	Bosque Serrano
189	Bosque de Llanura

Equivalencias entre los modelos de cobertura propuesto Scott et al. (2005) y las coberturas presentes en el área de estudio.

Posteriormente se obtienen la proporción viva de un dado pixel cuando la vegetación está su pico de verdor (LR_{mx}) que se define de la siguiente manera,

$$LR_{mx} = 35 + 40 * (NDVI_{mx} - 100)/ 80$$

Con el LR_{mx} y el verdor relativo (reescalado entre 0 y 1, RG_f) se calcula la fracción de verdor relativo LR que determina la proporción actual de combustible vivo para el pixel,

$$LR = RG_f * LRmx / 100$$

Una vez obtenido el fm10hs fraccional y la fracción de verdor relativo y se procede a calcular el FPI utilizando la formula según Burgan *et al.* (1998),

$$FPI = (1 - TN_f) \cdot (1 - LR) \cdot 100$$

Los valores del FPI varían entre 0 y 100. Siguiendo lo propuesto por Laneve *et al.* 2011 (MFPI), los valores del FPI son clasificados en cinco clases de riesgo (más una sexta post revaluación con evapotranspiración) para generar un mapa temático, utilizando la siguiente escala (Tabla 2):

Tabla 2

Valor de FPI	Valor en MFPI
0 - 20	1
21 - 40	2
41 - 55	3
56 - 70	4
71 - 100	5

Equivalencia de valores entre la escala del índice FPI y el índice MFPI.

Para calcular la evapotranspiración siguiendo a Laneve *et al.* (2011) se utilizan las directrices propuestas por la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO) en Allen *et al.* 2006, para obtener los valores de evaporación de referencia (ET_o) mediante el método FAO Penman-Monteith. La ET_o expresa el poder evaporante de la atmósfera en una localidad y época del año específicas utilizando únicamente la información climática disponible, y no considera ni las características del cultivo, ni los factores del suelo (Allen *et al.* 2006).

Las ecuaciones para calcular el ET_o no serán descriptas en este manuscrito, exceptuando las correcciones topográficas de la incidencia de energía solar recibida que serán descriptas en el capítulo V.

Una vez obtenidos los valores de ET_o y realizada la clasificación del FPI se procede a ejecutar la reevaluación de las clases del índice mediante el siguiente procedimiento.

Se calculan los estadísticos Media ($\bar{X}$) y Desvío Estándar (σ) de ET_o para las clases 4 y 5. Una vez obtenidos esos valores, se procede de la siguiente manera.

- Para cada pixel perteneciente a la clase 5,
  - si el valor de ET_o del pixel es mayor que $\bar{X} + \sigma$ entonces el pixel es reclasificado a la nueva clase 6.
  - si el valor de ET_o del pixel es menor que $\bar{X} - \sigma$ entonces el pixel es reclasificado a la clase 4.
- Para cada pixel perteneciente a la clase 4,
  - si el valor de ET_o del pixel es mayor que $\bar{X} + \sigma$ entonces el pixel es reclasificado a la clase 5.

- si el valor de ET_o del pixel es menor que $\bar{X}$ - σ entonces el pixel es reclasificado a la clase 3.

Una vez aplicada estas reglas, se obtiene el MFPI (Laneve *et al.* 2011), un mapa temático con 6 clases de riesgo, siendo 6 la clase de mayor posibilidad de incendio. Posteriormente se procede al agregado de las zonas de riesgo.

Las zonas de riesgo, son aquellas donde la posibilidad de incendio (vulnerabilidad) por acumulación de biomasa seca y condiciones ambientales, entra en contacto con la probabilidad de ignición (amenaza) relacionada a la actividad humana.

Para poder determinar dichas áreas se utilizan capas vectoriales de rutas, ríos y poblados y partir de estas capas se generan archivos de mapa de bits de zona de amortiguación para cada una de las capas vectoriales. En este escrito se utiliza zona de amortiguación no como un area de mitigación de algún efecto externo, como comúnmente se utiliza en conservación, sino como estratos entre un pixel con combustible A a un pixel B con capacidad de afectar positiva o negativamente la propensión de A de incendiarse.

Para generar los mapas temáticos con zonas de amortiguación se utiliza una división espacial similar a la propuesta por Loboda *at al.* 2007 donde, creó 11 zonas de amortiguación para caminos y 20 para poblados. En ese trabajo tanto para caminos como para poblados las primeras seis subdivisiones eran las que poseían mayor carga de ignición (que representa una estimación cuantitativa de la frecuencia de ignición).

Considerando lo antes dicho inicialmente se crearon únicamente seis subdivisiones de un km de ancho, pero teniendo en cuenta que la resolución de las imágenes MODIS es de 250 m se decidió modificar las subdivisiones a un total de 24, pero abarcando el espacio de seis km.

En una distancia lineal desde el punto de interés (pixel con río, poblado o ruta) hasta los seis km, hay un total de 30 puntos riesgo. El valor de 30 fue asignado teniendo en cuenta la cantidad de puntos promedio necesarios para pasar de una clase a otra del MFPI (~19) más diez puntos más para evitar una posible subestimación, la elección de dicha cifra fue totalmente arbitraria ya que no se disponían de cuantificaciones reales de la carga de ignición para los puntos de interés utilizados.

Para el caso de poblados y rutas (teniendo como origen el límite del camino o poblado) los valores de riesgo están organizados de manera descendente a medida que se alejan del punto de origen. Para los ríos el nivel de riesgo va en orden ascendente desde el punto de origen.

Una vez creadas las zonas de amortiguación son sumados (en el caso de caminos y poblados) o restados (en el caso de ríos) a los valores del FPI original. Como los valores del FPI van de 0 a 100, es posible que se obtengan valores de más de 100, en esos casos el pixel cuando es transformado en la escala del MFPI, se lo califica automáticamente como un pixel clase 6, aquellos pixeles que obtengan valores menores a 0 estos son clasificados

como clase 1. De esta forma se obtiene finalmente un índice de riesgo de incendio.

En el siguiente gráfico (Figura 1) se resume el proceso de cálculo del índice.

Figura 1

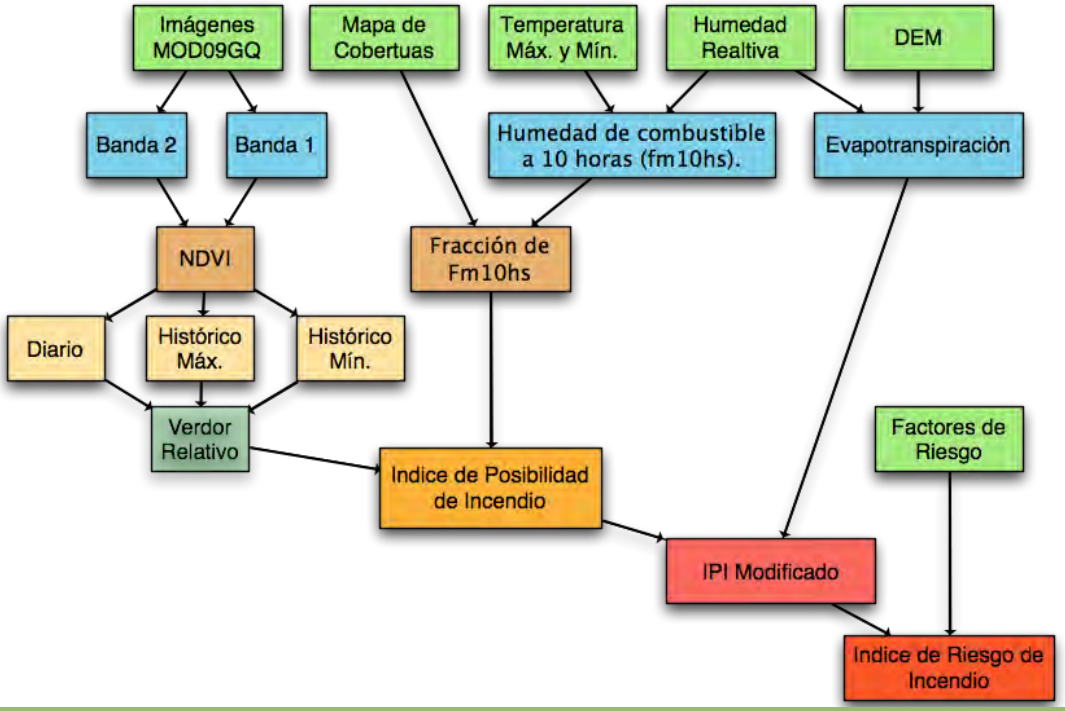


Diagrama de construcción del Índice de Riesgo de Incendio.

Capítulo III: Reflectancia de superficie

III.I Disponibilidad y Selección de sensores.

Para realizar el índice modificado de posibilidad de incendio es necesario contar con imágenes satelitales del área de estudio, tanto actuales como históricas, la información de archivo es indispensable para poder obtener los valores de NDVI, máximos y mínimos, posibles para cada pixel de la zona elegida y las imágenes actuales son utilizadas para los cómputos del MFPI de cada día.

Al momento de realizar este trabajo los satélites, cuyas imágenes estaban disponibles de manera libre y con un archivo histórico, eran los satélites del programa LandSat con una resolución temporal de 16 días y una resolución espacial promedio de 30 m con los sensores *Thematic Mapper* (TM)*, Enhanced Thematic Mapper* (ETM), *Enhanced Thematic Mapper Plus* (ETM+) o *Operational Land Imager* (OLI), y los satélites Terra y Aqua con resolución temporal de 24 h y 250 m de resolución espacial con el sensor *Moderate-Resolution Imaging Spectroradiometer* (MODIS)

Evaluando los productos disponibles por ambos sensores se decidió utilizar los generados por el sensor MODIS, ya que si bien este tiene menor resolución espacial, la resolución temporal que provee es más alta que la de los sensores a bordo de los satélites LandSat. Esa característica es clave a la hora de evaluar el estado de la vegetación, ya que permite conocer los cambios en el contenido de agua y verdor con una actualización diaria, dando al índice mayor exactitud.

Otra característica que llevó a elegir los productos del sensor MODIS es la continuidad y disponibilidad de los productos. El archivo de imágenes de MODIS es amplio, poseyendo un registro continuo para el área de estudio desde el 2000, manteniendo siempre las mismas condiciones de calibración y preprocesamiento. Esa situación no ocurre con los productos LandSat, ya que el archivo está fragmentando, compuesto por los productos de los sensores a bordo de los distintos satélites LandSat, los cuales además necesitan de un preprocesamiento y calibración específica para cada sensor.

III.II Características de los productos MODIS utilizados

Se seleccionó la base de datos MODIS Surface Reflectance como origen de los datos de entrada. Este es un conjunto de productos de resolución espacial variable entre 250 m y 1 km, con siete bandas, banda 1 (620-670 nm), banda 2 (841-876 nm), banda 3 (459-479), banda 4 (545-565 nm), banda5 (1230-1250 nm), banda 6 (1628-1652 nm), y banda 7 (2105-2155 nm). Todas las imágenes provistas en estos productos ya tienen valores de reflectancia de superficie con corrección de los efectos de gases y aerosoles atmosféricos, también están provistas de georeferenciación con proyección sinusoidal.

Los productos descargados fueron dos: MOD09GA que provee raster diarios de las bandas 1 a 7 de reflectancia de superficie con 500 m de resolución espacial y MOD09GQ que provee raster diarios de las bandas 1 y 2 de reflectancia de superficie con 250 m de resolución espacial.

Del producto MOD09GQ se obtienen la banda 1 que captura longitudes de onda entre 620-670 nm (Rojo) y la banda 2 que captura longitudes de onda entre 841-876 nm (Infrarrojo cercano). Del producto MOD09GA se obtiene la banda 6 que captura longitudes de onda entre 1628-1652 nm (Infrarrojo de onda corta)

Junto con las bandas de reflectancia se obtiene de ambos productos su respectiva banda de calidad (*Reflectance Band Quality*) la cual debería contener información respecto al estado de cobertura nubosa, tipo de corrección atmosférica realizado y valores del pixel dentro del rango aceptable, para cada pixel y banda, pero la información sobre cobertura nubosa no fue encontrada.

Las imágenes de reflectancia de superficie utilizadas pueden ser separadas en dos grandes grupos, uno aquel compuesto por imágenes del archivo de MODIS, de las cuales fueron utilizadas imágenes a partir de enero de año 2010 hasta marzo de 2013. Entre los años 2010-2011 se descargaron 6 imágenes por año las cuales son representativas de la fluctuación del verdor de la vegetación a lo largo del año y entre 2012 y 2013 se obtuvieron todas las imágenes disponibles. Estas imágenes se utilizaron para obtener los valores máximos y mínimos del Índice de Vegetación de Diferencia Normalizada (NDVI) para cada pixel del área de estudio. El segundo grupo es aquel compuesto por las imágenes correspondiente al día de producción del IRI, las cuales se utilizan para obtener NDVI y Verdor Relativo (comparando el NDVI actual con los valores históricos).

III.III Procesamiento de productos MODIS

Para generar el índice de NDVI del área de estudio fue necesario utilizar 2 baldosas (el archivo de imágenes MODIS está organizado en baldosas) correspondientes a la columna 12 (h12) y filas 11 y 12 (v11, v12). Las imágenes fueron procesadas mediante el software

MRTools provisto por el servicio de MODIS. Durante este procesamiento ser procedió a unir ambas baldosas, recortar el área de estudio, re-proyectarlas a coordenadas geográficas (datum WGS84) y transformadas a formato GeoTIFF.

Los archivos de salida de MRTools corresponden a una imagen por cada banda de reflectancia, más otra imagen también GeoTIFF correspondiente a la banda de calidad. Cabe destacar que durante este proceso las bandas correspondientes a imágenes MOD09GA son redimensionadas a 250 m, para que posean idéntica resolución que las imágenes resultantes de procesar los archivos MOD09GQ. Tanto los algoritmos para union y como para redimensión de las imágenes son procesos que exceden el objetivo de esta tesis y se encuentran disponibles en el MRT User Manual o consultando directamente al *Land Processes Distributed Active Archive Center* (LP DAAC) de *National Aeronautics and Space Administration* (NASA)/*United States Geological Survey* (USGS).

Una vez obtenidas las imágenes de mapa de bits del área de interés en formato GeoTIFF y durante la generación de los índices de NDVI, se analizó cada imagen una para enmascarar aquellos pixeles que contuvieran nubes, sombra de nubes, cuerpos de agua y aquellos que tuvieran información dañada o falte la información.

Para detectar los pixeles con información faltante o dañada se utilizó la información disponible en las bandas de calidad (*Reflectance Band Quality).* Los datos en estas bandas están clasificados con un código binario (Vermote *et al.*, 2011) el cual es utilizado para enmascarar automáticamente los pixeles.

En el caso de la máscara de nubes, cuerpos de agua y sombras de nube, se marcaron regiones de interés para obtener el valor medio y desvío estándar de la reflectancia para cada una de las clases a enmascarar (Tabla 3), con un total de 150 pixeles por región de interés y por banda. Posteriormente se utilizó un árbol de decisión para realizar la máscara. Se eligió realizar este método rápido de máscara, ya que profundizar en algoritmos para ese proceso sería una tesis completamente aparte.

En el caso de pixeles faltantes o dañados las imágenes MODIS poseen un valor específico de pixel (-28672). Para generar la máscara de cada una de las otras clases se siguió con una codificación semejante, utilizando los valores -28670 para nubes, -28671 para pixel con información dañada, -28673 sombra de nube y -28674 cuerpos de agua. La razón por la cuál se eligieron estos números es porque están totalmente fuera del rango posible de valores de reflectancia y son por lo tanto inconfundibles.

No se utilizaron los datos provenientes de la banda de cobertura de nubes provista por el servicio MODIS, porque esta tiene una resolución espacial de 1km que transformada a 250m produciría pixeles con falso dato de nube, sumado a que solamente informa de nubes, faltando información de sobras de nube y cuerpos de agua.

Tabla 3

Máscara	Banda	Media	Desvio Estándar
Nubes	1	0,4198	0,1587
	2	0,4893	0,1385
	6	0,4502	0,2077
Sombra de Nubes	1	0,0441	0,0158
	2	0,0820	0,0211
	6	0,1132	0,0310
Cuerpos de Agua	1	0,0177	0,0064
	2	0,0098	0,0161
	6	0,0241	0,0282

Valores de media y desvío estándar por banda MODIS para cada cobertura (ROI) a enmascarar. Para cada clase se tomaron un total de 150 pixeles de muestra.

Capítulo IV: Características Del Área de Estudio

IV.I Ambiente y Geografía física

La provincia de Córdoba cubre 165.321 km^2. Posee un clima templado cálido a sub-tropical, con una temperatura media anual desde 16ºC en el este-sureste a 19ºC en el noroeste, la precipitación media anual decrece en el mismo sentido desde más de 800 a 400 mm (Capitanelli 1979). En la zona norte y oeste de la provincia existe un gran déficit de agua, este tiene su pico durante los meses de final del invierno y principios de la primavera, época donde se producen la mayor cantidad de incendios forestales.

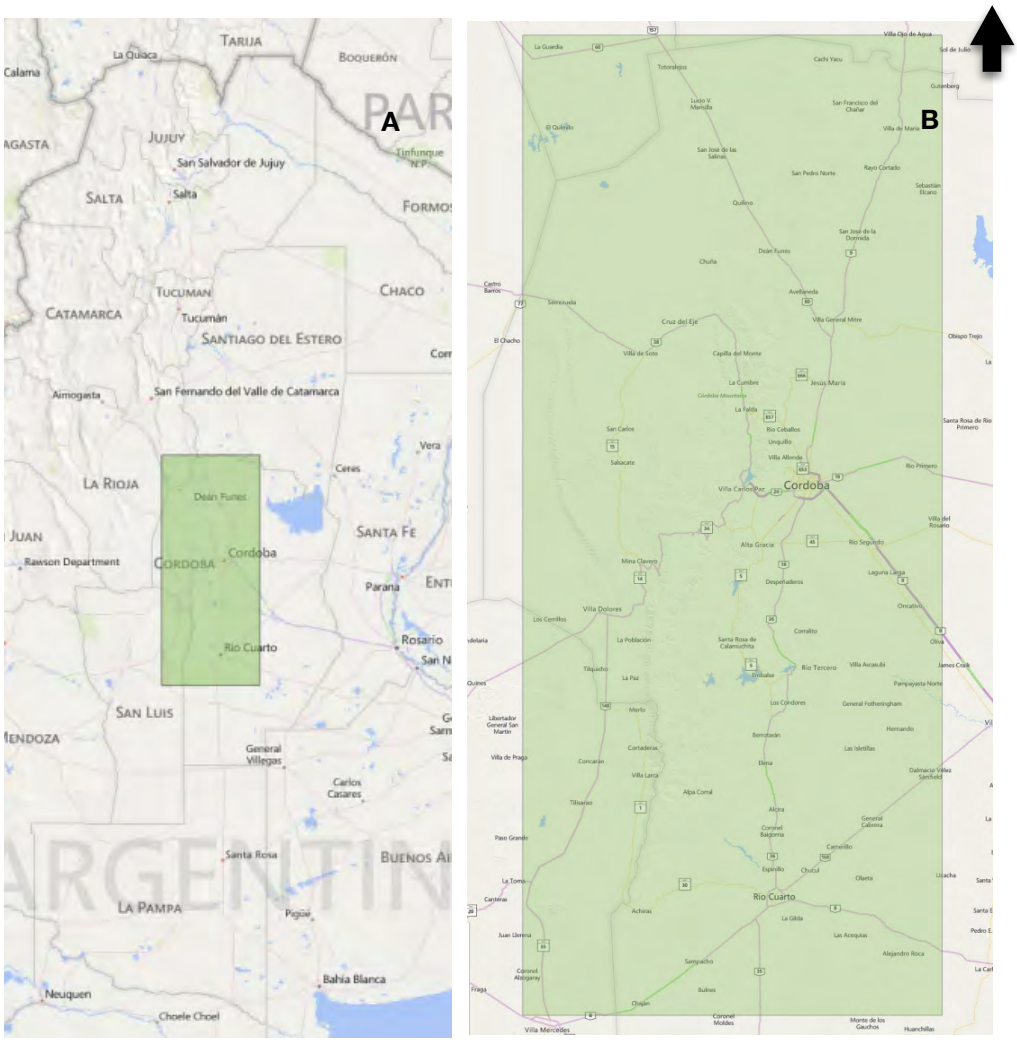
La provincia está recorrida en su parte centro-oeste por un cordón montañoso con un rango altitudinal desde los 600 hasta los 2884 msnm en su parte más alta, con grandes planicies hacia el este y oeste. La zona noroeste del territorio muestra una depresión salina (entre los 150 a 300 msnm) y también al extremo noreste donde existe una gran depresión salina con el lago salobre Mar Chiquita en su parte más baja. La zona de estudio comprende principalmente a la provincia fitogreográfica del Chaco (Luti *et al.* 1979), con las zonas montañosas perteneciendo al distrito de Chaco Serrano, mientras que las zonas bajas corresponden al distrito del Chaco Occidental. La vegetación presenta un patrón de bosques aislados, densos arbustales espinosos, pasturas semi-naturales y áreas cultivadas como consecuencia de la intervención humana que comenzó con quema de pasturas (actividad que perdura) por parte de las poblaciones nativas y continuó con la tala de bosques y el desarrollo de la agricultura por un periodo de más de 400 años, además se suma la deforestación producida por canteras y aumento de urbanización (Zak 2002, Gavier *et al.* 2006).

En las zonas de montaña la cobertura vegetal muestra marcados cambios a lo largo del gradiente altitudinal, por lo cual puede ser dividido en seis pisos altitudinales, llanura ubicada por debajo de los 500 msnm, bosque chaqueño entre los 500 y 800 msnm, bosque serrano entre los 800 y 1350 msnm, romerillal entre los 1350 y 1700 msnm, pastizal más bosque entre los 1700 y 2200 msnm y finalmente pastizal de altura ubicado por encima de los 2200 msnm (Capitanelli 1979).

El área de estudio elegida está delimitada con su esquina superior izquierda en 29º 30’ 4.43” latitud sur y 65º 34' 41.16” longitud oeste y su esquina inferior derecha en 33º 36' 26.93” latitud sur y 63º 31' 33.96” longitud oeste, abarcando una superficie de aproximadamente

88.622 Km^2 (Figura 2).

Figura 2



Polígono mostrando la ubicación del área de estudio. A Ubicación en el país. B Zoom.

IV.II Combustibles

Se considera combustible a toda cobertura vegetal presente en el pixel, englobando vegetación nativa, foránea, forestaciones y cultivos. El mapa de cobertura de combustibles fue obtenido a partir de combinar capas vectoriales de cobertura de áreas naturales (diferente tipos de bosque y pasturas), proveniente del trabajo de Zak *et al.* (2004), vectorizaciones de mapas disponibles sobre las distintas provincias fitogeográficas en la región (Luti *et al.*, 1979), mapas temáticos de pisos altitudinales de vegetación generados a partir de información bibliográfica sobre los distintos pisos de vegetación presentes en la

zona (Luti *et al.*, 1979), utilizando de base mapas de elevación digital (DEM) provistas por la misión Shuttle Radar Topography Mission (SRTM) (con una resolución aproximada de 90 m) (Nikolakopoulos *et al.* 2006*)* y capas vectoriales de uso de la tierra provistas por el Instituto Geográfico Nacional.

Utilizando el software ENVI, en un mosaico compuesto de 22 imágenes del sensor OLI (del satélite LandSat 8) con una resolución espacial aproximada de 30 m obtenidas entre los meses de mayo y junio de 2013, se marcaron regiones de interés (ROI) para clases vegetación (varias clases), suelo desnudo, roca y salinas, a partir de combinar la información disponible en las capas vectoriales y mapas temáticos.

Posteriormente utilizando las ROI seleccionadas y disponiendo en un archivo multi banda compuesto por, pisos de vegetación, mosaico LandSat (bandas 2 a 8) y DEM, todo a una resolución de 30 m, se realizó una clasificación supervisada mediante el algoritmo de Support Vector Machine (SVM) también utilizando ENVI, generando la capa final de cobertura de combustible, clasificada en un total de 12 clases (Figura 3) compuesta por, Clase 1 Pastizal de Altura, Clase 2 Romerillal, Clase 3 Vegetación Halófita, Clase 4 Bosques de Llanura, Clase 5 Salinas, Clase 6 Roca Desnuda, Clase 7 Pastizal de Llanura, Clase 8 Bosque Serrano, Clase 9 Pastizal más Bosque de Altura, Clase 10 Cuerpos de Agua, Clase 11 Cultivos, Clase 12 Zonas Urbanas. Cabe aclarar que en la clase Cultivos no se hace diferencia entre plantaciones de árboles y plantaciones de alimentos, ya que no se contaba con información al respecto.

IV.III Zonas de amortiguación

Las zonas de amortiguación son áreas de gradientes ubicados entre dos pixeles con características particulares a determinar por el investigador. No deben ser confundidas (aunque se pueden construir con el mismo método) con zonas de amortiguación como son aplicadas en los casos de protección de reservas naturales para mitigar los efectos de las zonas no protegidas.

Para generar la zonas de amortiguación se utilizaron capas vectoriales provistas por el IGN del proyecto Argentina SIG 250, realizado por el Instituto en el año 2000. Los datos se encuentran referenciados en coordenadas geográficas, utilizando el Sistema de Referencia WGS 84 y el Marco de Referencia POSGAR 07 (Código EPSG:4326).

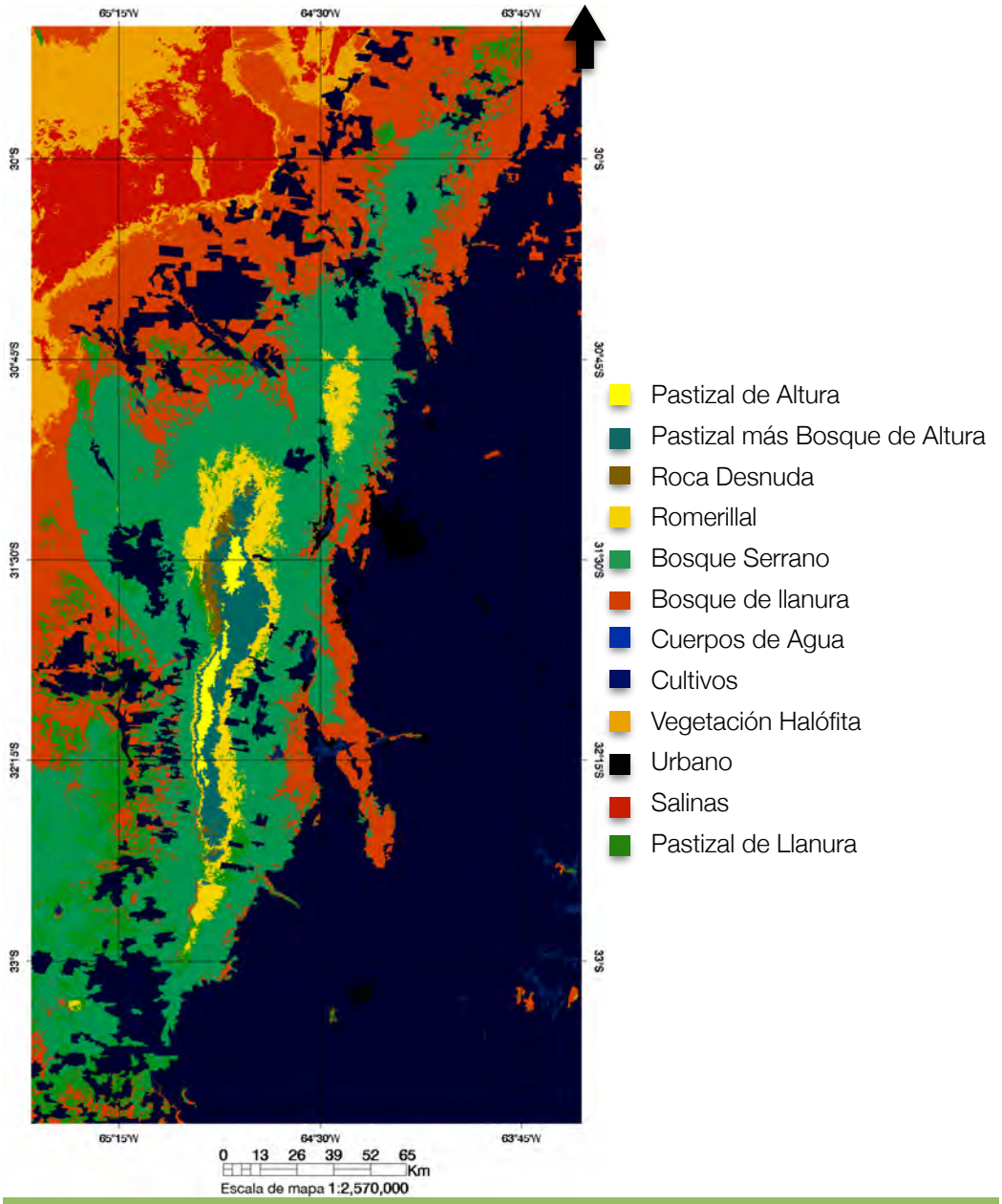
Las capas elegidas fueron de ríos y cuerpos de agua, centros urbanos y rutas. Utilizando el software QGIS se recortaron las capas vectoriales a la zona de estudio, ya que las mismas son capas de todo el territorio nacional. Mediante el software ENVI se crearon imágenes GeoTIFF de zonas de amortiguación a partir de las capas vectoriales.

Para generar los intervalos de las zonas se utiliza una escala discreta con un incremento de 1,30 puntos cada 250 m, los valores elegidos son una primera aproximación para cuantificar

los factores que incrementan el riesgo de incendio basados en lo planteado por Loboda *et al.* (2007)

Los mapas de zonas de amortiguación resultantes nos muestran valores de riesgo decrecientes desde el punto de origen para el caso de poblados y rutas, dado que estos serían puntos de origen para potenciales igniciones. Y valores crecientes para los ríos, ya que la humedad provista por los mismos genera zonas con baja probabilidad de ignición y además funcionarían como barreras corta fuego. Cabe destacar para el caso de rutas y caminos que si bien los mismos también actúan como barreras corta fuego, estos son en gran medida la fuente principal de orígenes de incendio desde sus márgenes, es por ello que la zona de amortiguación para este caso tiene su punto máximo en el origen del camino.

Figura 3



Mapa de cobertura de capas de combustibles.

Capítulo V: Correcciones topográficas de la evapotranspiración

V.I Información meteorológica

Las variables meteorológicas necesarias (velocidad de viento, temperatura y humedad), para realizar el MFPI fueron provistas por CONAE en formato de imagen digital GeoTIFF con georreferenciación en coordenadas geográficas. La resolución espacial de dichas imágenes fue de 0,045045 grados (10 Km aproximadamente).

Estas imágenes tienen un tiempo de actualización de 24 h y se obtiene temperatura máxima y mínima para el día, humedad relativa máxima y mínima y velocidad media del viento a 2 m del nivel de suelo. Durante los cálculos de evapotranspiración las mismas fueron redimensionadas a una resolución 250 m. Para realizar la redimiensión se utilizó el método de interpolación cubica *spline* como el que realiza ENVI pero el código fue obtenido del *Cookbook* de Scipy (http://wiki.scipy.org/Cookbook/Rebinning).

V.II Evapotranspiración

La evaporación es el proceso por el cual el agua líquida se convierte en vapor de agua (vaporización) y se retira de la superficie evaporante (remoción de vapor). El agua se evapora de una variedad de superficies, tales como lagos, ríos, caminos, suelos y la vegetación mojada (Allen *et al.* 2006).

Este proceso de evaporación depende de la cantidad de radiación solar directa, la humedad ambiente, la temperatura del aire y la velocidad del viento en superficie que desplaza las masas de aire saturado con vapor de agua permitiendo la continuidad del proceso (Allen *et al.* 2006).

La transpiración consiste en la vaporización del agua líquida contenida en los tejidos de la planta y su posterior remoción hacia la atmósfera. Los cultivos pierden agua predominantemente a través de los estomas. La transpiración, igual que la evaporación

directa, depende del aporte de energía, del gradiente de presión del vapor y de la velocidad del viento. Por lo tanto, la radiación, la temperatura del aire, la humedad atmosférica y el viento también deben ser considerados en su determinación (Allen *et al.* 2006).

Tanto evaporación y transpiración ocurren simultáneamente y no hay una manera sencilla de distinguir entre estos dos procesos, por lo cual la pérdida de agua que ocurre es denominada evapotranspiración (ET) (Allen *et al.* 2006).

La evapotranspiración se expresa normalmente en milímetros (mm) por unidad de tiempo. Esta unidad expresa la cantidad de agua perdida de una superficie vegetada en unidades de altura de agua. La unidad de tiempo puede ser una hora, día, 10 días, mes o incluso un completo período de cultivo o un año (Allen *et al.* 2006).

Existen diversas formas de medir la evapotranspiración y el método que se utilizó es la determinación de la evapotranspiración de referencia (ET_o). La misma es una tasa de evapotranspiración que ocurre en condiciones ideales y en un cultivo de referencia (pastura), la ventaja de utilizar dicha tasa es que se obtiene independientemente del tipo de suelo, formas de cultivo o manejo forestal, lo cual permite evaluar rápidamente y de una manera homogénea todo el territorio de estudio (Allen *et al.* 2006).

Los únicos factores que afectan ET_o son los parámetros climáticos y puede ser calculado a partir de datos meteorológicos. El método FAO Penman-Monteith se recomienda como el único método de determinación de ET_o con parámetros climáticos. Este método ha sido seleccionado debido a que se aproxima de una manera cercana la ET_o de cualquier localidad evaluada, tiene bases físicas sólidas e incorpora explícitamente parámetros fisiológicos y aerodinámicos (Allen *et al.* 2006). La fórmula para obtener ET_o es la siguiente:

$$ET_o = \frac{0{,}408\ \Delta (R_n - G) + \gamma \frac{900}{T + 273} u_2 (e_s - e_a)}{\Delta + \gamma (1 + 0{,}34\ u_2)}$$

Donde R_n es la radiación neta en la superficie del cultivo, G es el flujo del calor de suelo, T es la temperatura media del aire a 2 m de altura, u_2 la velocidad del viento a 2 m de altura, e_s la presión de vapor de saturación, e_a la presión real de vapor, Δ es la pendiente de la curva de presión de vapor y γ la constante psicrométrica.

El desarrollo completo de las fórmulas y los pasos a seguir durante cálculo de ET_o se presentan en el libro Evapotranspiración del cultivo de Allen *et al.* 2006, sin embargo se explica aquí la sección específica del cálculo de Radiación Extraterrestre ya que es el punto dónde en el presente trabajo se realizan modificaciones para obtener valores más cercanos a la realidad.

V.III Cálculos de la iluminación sobre planos inclinados.

Para los cálculos de la evapotranspiración es necesario saber cuánta energía solar llega a las diferentes zonas del terreno. El área de estudio, como ya se nombró con anterioridad, posee montañas. Éstas dependiendo de su orientación Norte-Sur y su inclinación, reciben diferente cantidad de energía para un mismo día.

Con el objetivo de obtener una medición lo más real posible, se obtuvieron a partir de un DEM del área y mediante el software ENVI, los valores de pendiente y orientación para cada pixel. Posteriormente con estos datos y siguiendo los cálculos propuestos por Iqbal (1983) y Kumar *et al.* (1997) se obtiene el ángulo de radiación solar al inicio y a la puesta del sol.

Una vez obtenidos los ángulos de inicio (ω_1) y puesta del sol (ω_2), se procede al cálculo corregido por topografía de la radiación extraterrestre (R_a), determinando así la cantidad total de horas de sol directo que recibe una ladera en particular y ajustando de la cantidad de energía solar que incide en cada pixel según su orientación y pendiente.

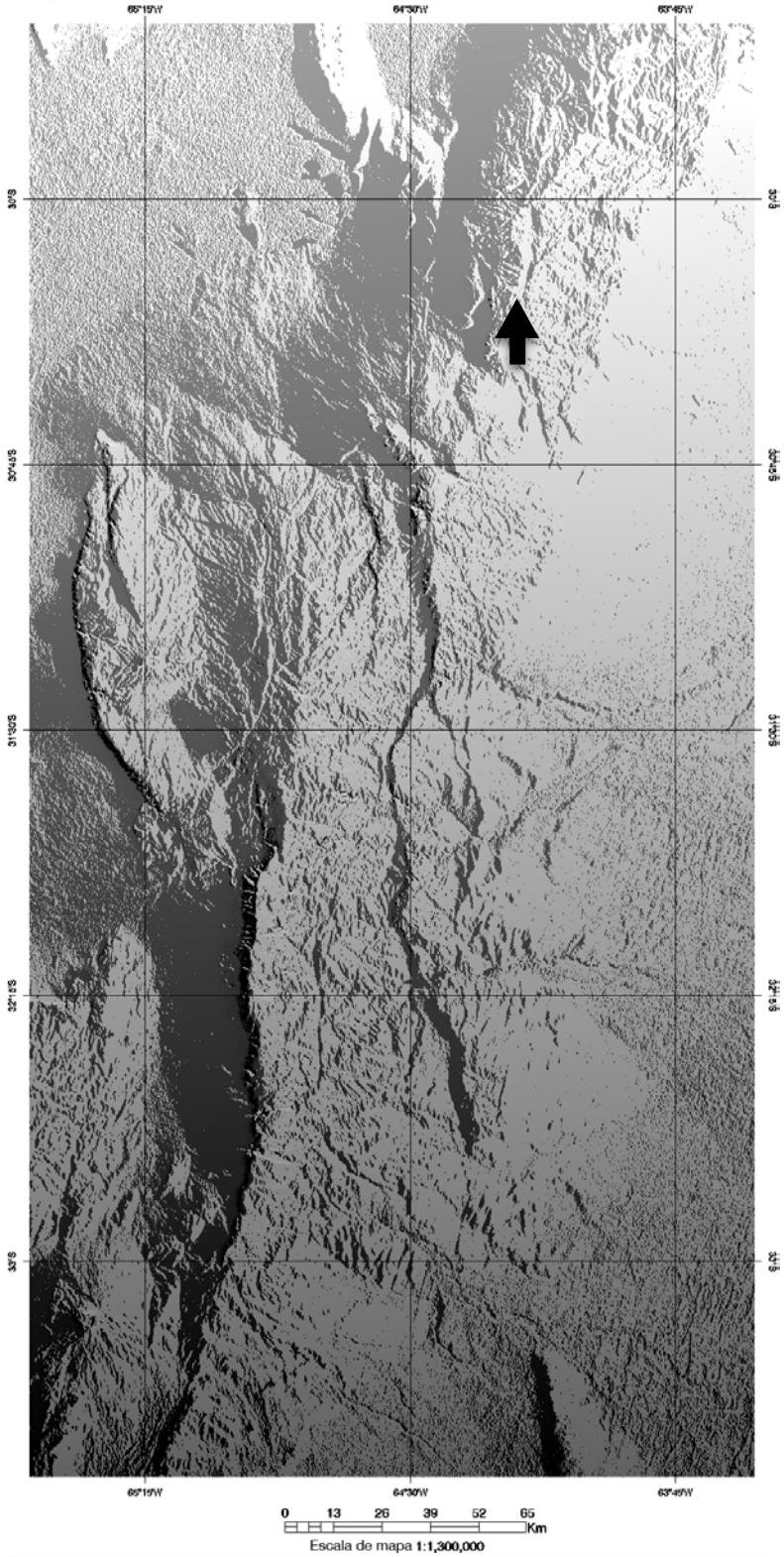
Para poder utilizar esta nueva información generada, se eligió la formula de R_a para periodos de tiempo en horas la cual permite introducir ambos ángulos al cálculo.

$$R_a = \frac{12*60}{\pi} \cdot G_{sc} \cdot d_r \cdot \left[(\omega_2 - \omega_1) \cdot sen(\varphi) \cdot sen(\delta) + cos(\varphi) \cdot cos(\delta) \cdot (sen(\omega_2) - sen(\omega_1))\right]$$

Donde G_{sc} es la Constante solar (0,082 MJ m^{-2} min^{-1}), d_r es la distancia relativa inversa Tierra-Sol, δ la declinación solar y φ la latitud en radianes. De esta forma se obtiene la totalidad de radiación extraterrestre para la cantidad de horas efectivas diarias (ω_2 - ω_1) que la ladera está expuesta (Figura 4).

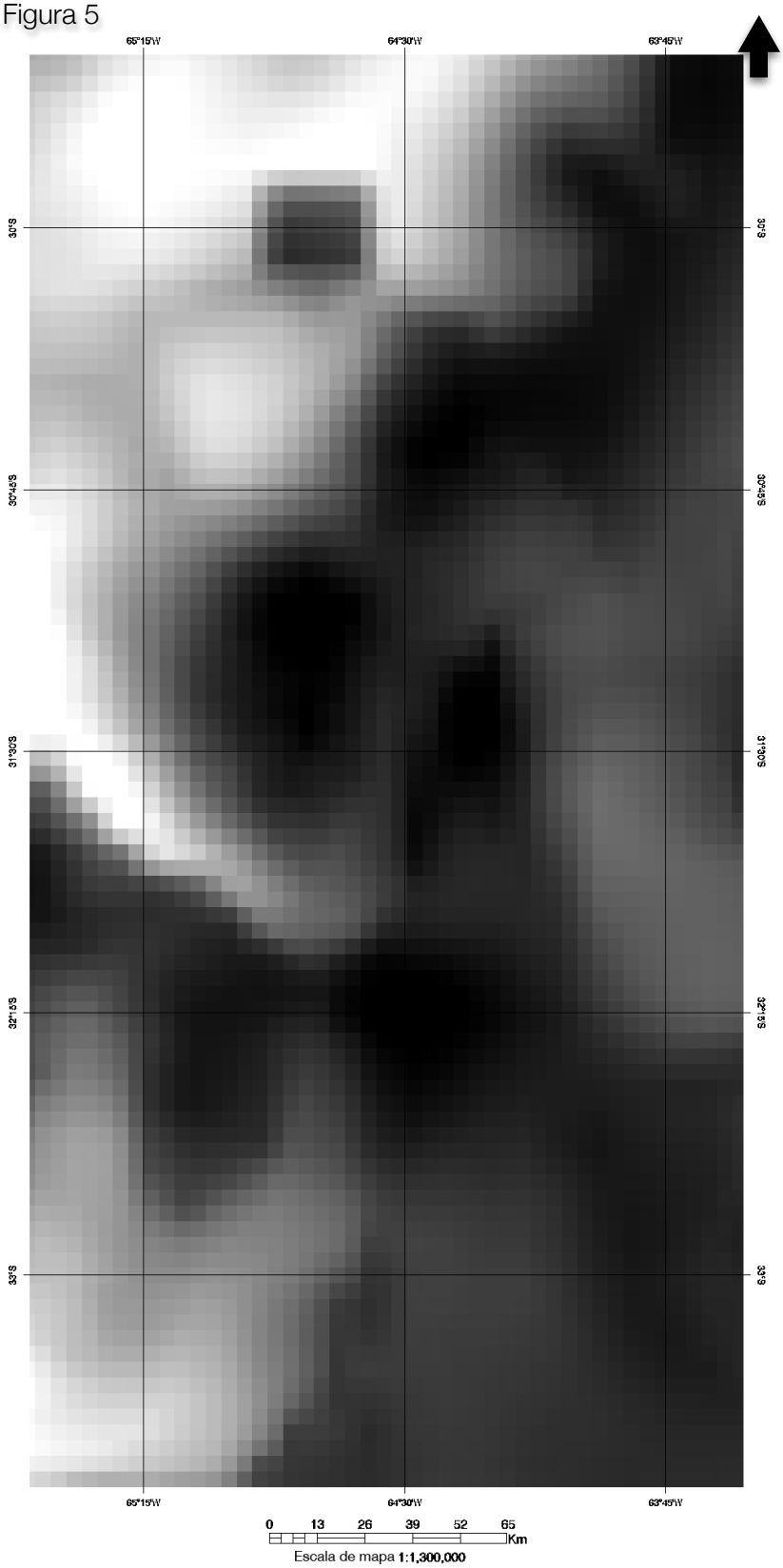
Si bien los valores calculados de ET_o están corregidos según la cantidad de energía incidente, cómo se puede ver en la Figura 5, debido a la baja resolución espacial disponible en los productos meteorológicos (de origen 10 km), se produce una homogeneización de los valores obtenidos, dando una apariencia “pixelada” a la imagen.

Figura 4



Radiación extraterrestre diaria, Zona Sierras de Córdoba. Colores más claros indican mayor radiación recibida. Imagen del día 5 - IX - 2013

Figura 5



Evapotranspiración del cultivo de referencia (ETo) para el area de Sierras de Córdoba. Colores más claros marcan mayor evapotranspiración. Imagen del día 5 - IX - 2013

Capítulo VI: Construcción del Software

VI.I Elección del lenguaje de programación

Al momento de elegir el lenguaje de programación que se utilizaría para implementar de manera automatizada la construcción del IRI, se tuvieron en cuenta varios candidatos, entre ellos IDL que es el lenguaje de programación de ENVI, MatLab, R y Python, siendo este último el elegido para el desarrollo del presente trabajo, por poseer las siguientes características.

Es gratuito y multiplataforma, es decir que los programas escritos en Python son independientes del Sistema Operativo (SO) donde fueron escritos.

Es un lenguaje robusto, sólido y potente. Python tiene relativamente pocas lineas de código, lo que genera que sea menos propenso a tener problemas, más fácil de depurar y de mantenimiento sencillo.

Es fácil de aprender y usar. Python tiene una sintaxis regular y simple. Las declaraciones terminan con fin de linea y la estructura de bloques se indica por indentación con espacios. Los programas Python poseen un aspecto de pseudo-código ejecutable. Esto elimina gran cantidad de dificultades que pueden surgir a programadores principiantes, especialmente en la ubicación de punto y coma, corchetes y paréntesis que son comunmente utilizados en otros lenguajes para determinar bloques.

Un ejemplo de la simplicidad de los programas Python es el clásico programa “Hola Mundo” que simplemente es:

```
print “hello World”
```

a diferencia de C++ donde es:

```
#include <iostream.h>
int main()
{
  cout << "Hello World!";
}
```

Sumada a esas características, existe también una amplia base de datos y comunidad de

soporte para procesamiento de datos científicos, entre ellos la librería Geospatial Data Abstraction Library (GDAL) que provee de gran cantidad de herramientas para el procesamiento y manipulación de imágenes satelitales.

VI.II Requerimientos del programa y ejecución.

La estructura del programa fue planteada a modo de múltiples sub-programas que son llamados por uno principal (Etapa 0) el cual se encarga de fijar los directorios de trabajo, obtener los valores de georeferencia para cuando son guardadas las imágenes GeoTIFF y ejecutar consecutivamente cada uno de los subprogramas.

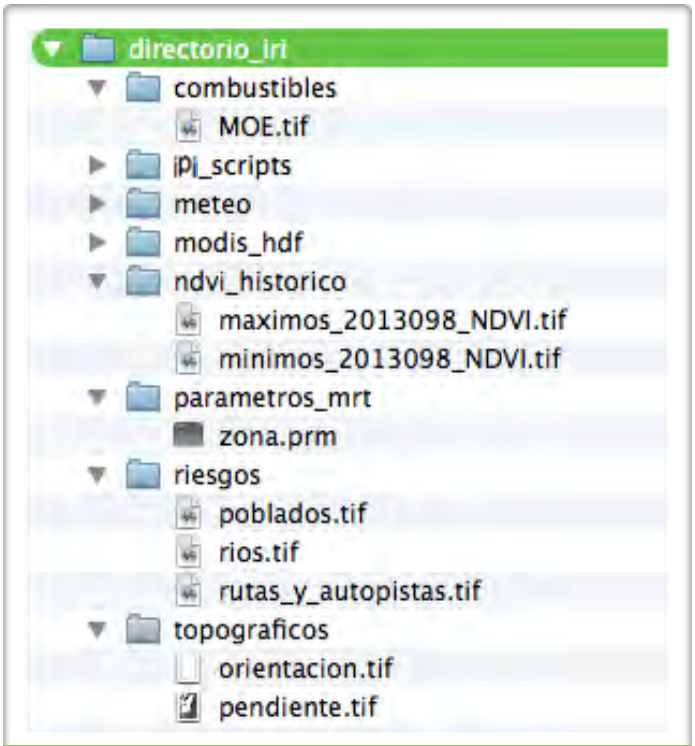
Los requisitos para el funcionamiento del programa son tener instalado Python 2.7.3, las librerías gráficas y de manipulación de imágenes de mapa de bits GDAL, libjpeg-libpng, matplotlib, las librerías científicas NumPy, scipy y el software MRTools.

El programa corre en la interfaz de linea de comandos del sistema operativo. Para poder comenzar la ejecución, además de las librerías mencionadas anteriormente, deben existir un directorio de trabajo que posea determinados sub-directorios. Se recomienda que el directorio principal de trabajo esté ubicado en la carpeta raíz del sistema operativo para facilitar el acceso por línea de comando y que la ruta a mismo no posea directorios con nombres con caracteres especiales (tildes, apóstrofes, etc), tampoco con nombres con espacios.

Dentro del directorio principal de trabajo deben existir una carpeta que contenga todos los *scripts* del programa, directorio con las imágenes originales MODIS en formato hdf, directorio con las imágenes de orientación y pendiente en formato GeoTIFF, directorio con la imagen en GeoTIFF de las capas de combustible, directorio con los archivos de parámetros para el funcionamiento de MRTools, directorio con las imágenes de máximo y mínimo históricos del NDVI en GeoTIFF, directorio con las imágenes de información meteorológica (temperatura máxima, temperatura mínima, humedad relativa máxima, humedad relativa mínima, velocidad del viento a 2 m) y finalmente una carpeta con las imágenes de zonas de amortiguación para los factores de riesgo.

Al comienzo de la ejecución se verifica la existencia de las carpetas necesarias para el funcionamiento, de faltar alguna, el programa se detiene inmediatamente informando el error y el nombre de carpeta faltante. Para evitar complejidad de operación para un usuario final se recomienda que tanto las carpetas básicas como los archivos que contienen estén nombrados correctamente. El árbol del directorio de trabajo debería queda como en la Figura 6. Se puede observar en la figura que ciertas carpetas no están desplegadas, en el caso de esos directorios específicos los nombres de los archivos que contienen son aquellos que forman los scripts del software (provistos en este trabajo) y las imágenes de mapa de bit provistas por el servicio MODIS o por CONAE (datos meteorológicos) por lo cual ya están asignados sus nombres previamente y no es necesario modificarlos.

Figura 6



Estructura del árbol de directorios, mostrando las carpetas iniciales necesarias para el funcionamiento del software.

Después de comprobar la existencia de los directorios básicos, el programa crea automáticamente las capetas de archivos de salida, estas son “archivados” donde se almacenan las imágenes hdf MODIS posterior a extraer las bandas necesarias, “remuestreos” carpeta destino de las bandas 1, 2, 6 y de calidad en formato GeoTIFF, “ndvi_dia” para los resultados en GeoTIFF del cálculo del NDVI para el día de obtención del IRI, “verdor_relativo” para almacenar los resultados de RG, “procesados” para archivar las imágenes GeoTIFF de las bandas después de que se calculó NDVI y RG y la carpeta “fpi” destino de la imagen en mapa de bit en formato GeoTIFF del Índice de Posibilidad de Incendio.

Como se nombró anteriormente el programa consta de 7 sub-programas, el primero de ellos (Etapa I, para MOD09GA y Etapa I, para MOD09GQ) es el encargado de realizar los mosaicos, recortarlos y cambiar la georeferencia a coordenadas geográficas de las imágenes MODIS en su formato HDF, para lograr eso se realiza un llamado del programa MRTools (python subprocess.call). Previo a esto se indexa la carpeta “modis_hdf” para generar el archivo de parámetros de mosaico (ver MODIS Reprojection Tool User’s Manual 2011) que contiene las rutas para los archivos que componen el mosaico.

La Etapa II es en la cual se produce el cálculo del NVDI para el día a procesar. Además del NDVI se marcan los pixeles que posean defectos (de acuerdo con la banda de calidad MODIS), pixeles con agua y aquellos con cobertura nubosa (nubes y sombra), pasando su valor al correspondiente según la codificación expuesta en el capítulo III. Una vez generado el NDVI y los pixeles enmascarados, la imagen es guardada en formato GeoTIFF.

En la Etapa III se realizan los cálculos primero del ET_o y posteriormente del fm10hs. Para calcular los valores del ET_o se procede de dos formas: utilizando los valores de temperatura, velocidad del viento y humedad en las dimensiones originales (con una resolución aproximada de 10 km) se calculan la presión media de vapor de saturación, la pendiente de curva de presión de saturación de vapor y la, presión real de vapor (e sub a). Por otro lado se producen los valores de radiación de extraterrestre utilizando datos de entrada a 250 m de resolución. Posteriormente al momento de calcular el valor de radiación solar, se redimensionan los datos de 10 km hasta 250 m, se finaliza el calculo del ET_o y se guarda la imagen en GeoTIFF. Después de obtener la evapotranspiración de referencia, se calcula el fm10hs usando los valores de humedad y temperatura (ya transformados a 250 m), que es también almacenada en formato GeoTIFF.

La Etapa IV calcula los valores de verdor relativo para toda el área, con este dato se pasa a la Etapa V donde se calcula el índice de posibilidad de incendio (IPI) con las correcciones topográficas utilizando la evapotranspiración. Una vez calculado el IPI se guarda también el resultado del mismo en una imagen GeoTIFF que puede ser posteriormente transformada en un mapa para ser distribuida a los servicios de alerta temprana.

En el Anexo I se dispone del código del software para poder apreciar de manera más detallada los algoritmos que se llevan acabo en cada una de las etapas.

Cabe destacar que cada una de las etapas procesa todos los archivos asignados y genera los archivos de salida para recién después comenzar la siguiente etapa, es decir si para la Etapa I existen 20 pares de archivos MOD09GA, se pasa a la Etapa II cuando todos los pares de imágenes hayan sido procesados. Este funcionamiento compartimentado permite frenar el proceso entre etapas y reanudarlo sin necesidad de comenzar desde la primera etapa, siempre que las variables que indican las rutas a los directorios de trabajo estén previamente fijadas.

Capítulo VII: Resultados y Conclusiones

VII.I Resultados en mapas

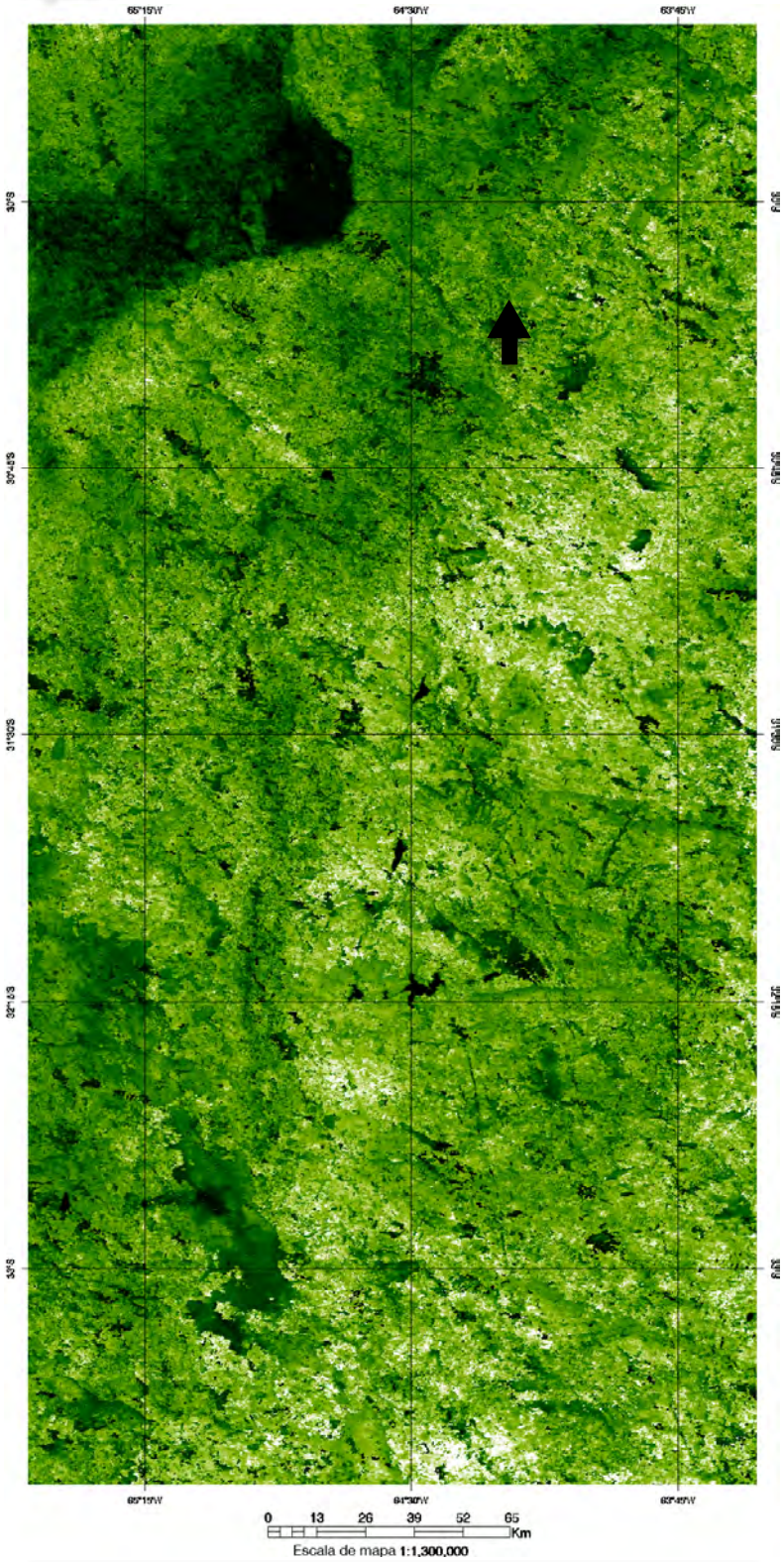
A lo largo del proceso de construcción y durante el funcionamiento del software para generar el mapa de posibilidad de incendio se crearon una serie de imágenes, algunas de ellas ya fueron presentadas en capítulos anteriores en estructura de mapas. A continuación se muestran los mapas restantes.

En primera instancia (Figuras 7 y 8) están las imágenes creadas a partir de los valores de NDVI obtenidos utilizado los archivos obtenidos de MODIS entre 2010 y 2011. Representados con una coloración artificial en tonos de verde, los valores de interés biológico del NDVI se encuentran en el rango de 0 a 1. En las figuras el color que es correspondiente a 0 es negro y aquel correspondiente a 1 es blanco, todas las variaciones de tonos de verde corresponden a los valores intermedios entre los extremos.

En la Figura 9 se pueden apreciar secciones de los mapas de zonas de amortiguación. Se presenta solamente una sección pequeña del area de estudio ya que se puede apreciar con mayor detalle el gradiente generado.

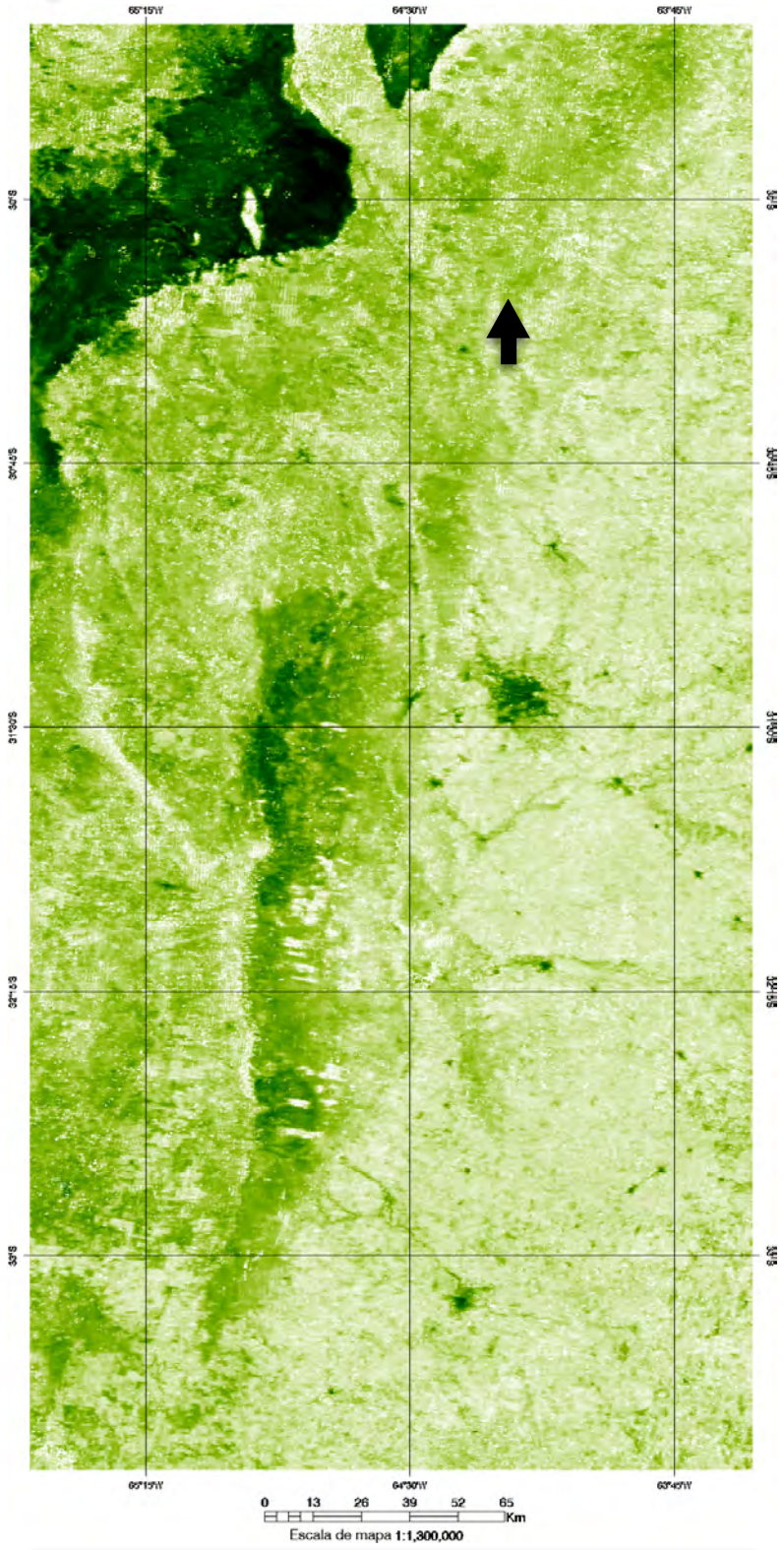
Finalmente la Figura 10 se presenta el mapa del índice de posibilidad de incendio. La imagen corresponde al 5 de septiembre de 2013, día anterior al comienzo de los incendios forestales en las Sierras de Córdoba que afectaron más de 40.000 hectáreas. El mapa del IPI tiene una escala que va del 1 a 6, siendo 1 el valor que indica que la posibilidad de incendio es nula o “muy baja” y 6 que la posibilidad de incendio es “muy alta”

Figura 7



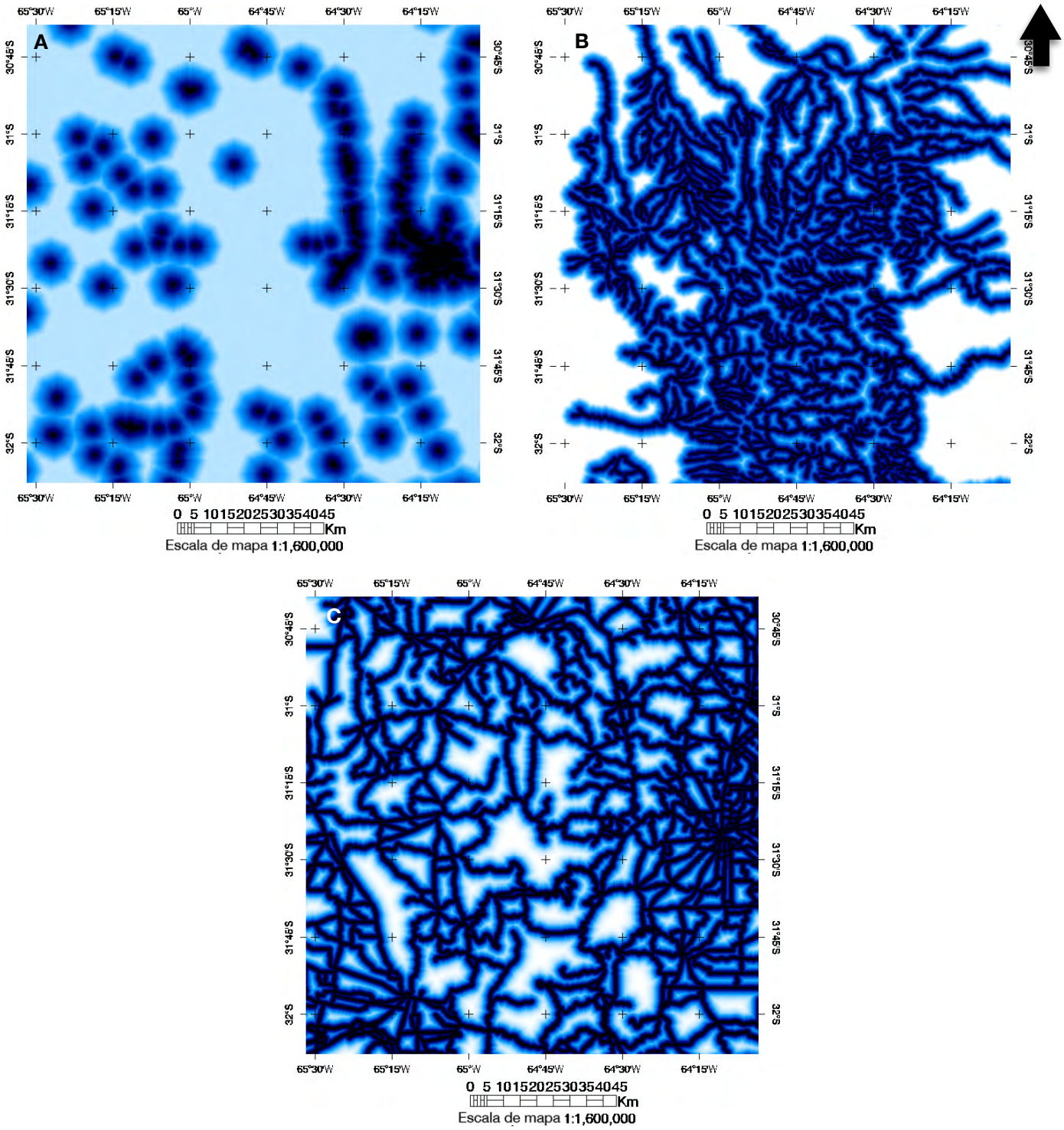
Mapa de NDVI histórico mínimo. Colores más claros indican mayor valor de NDVI.

Figura 8



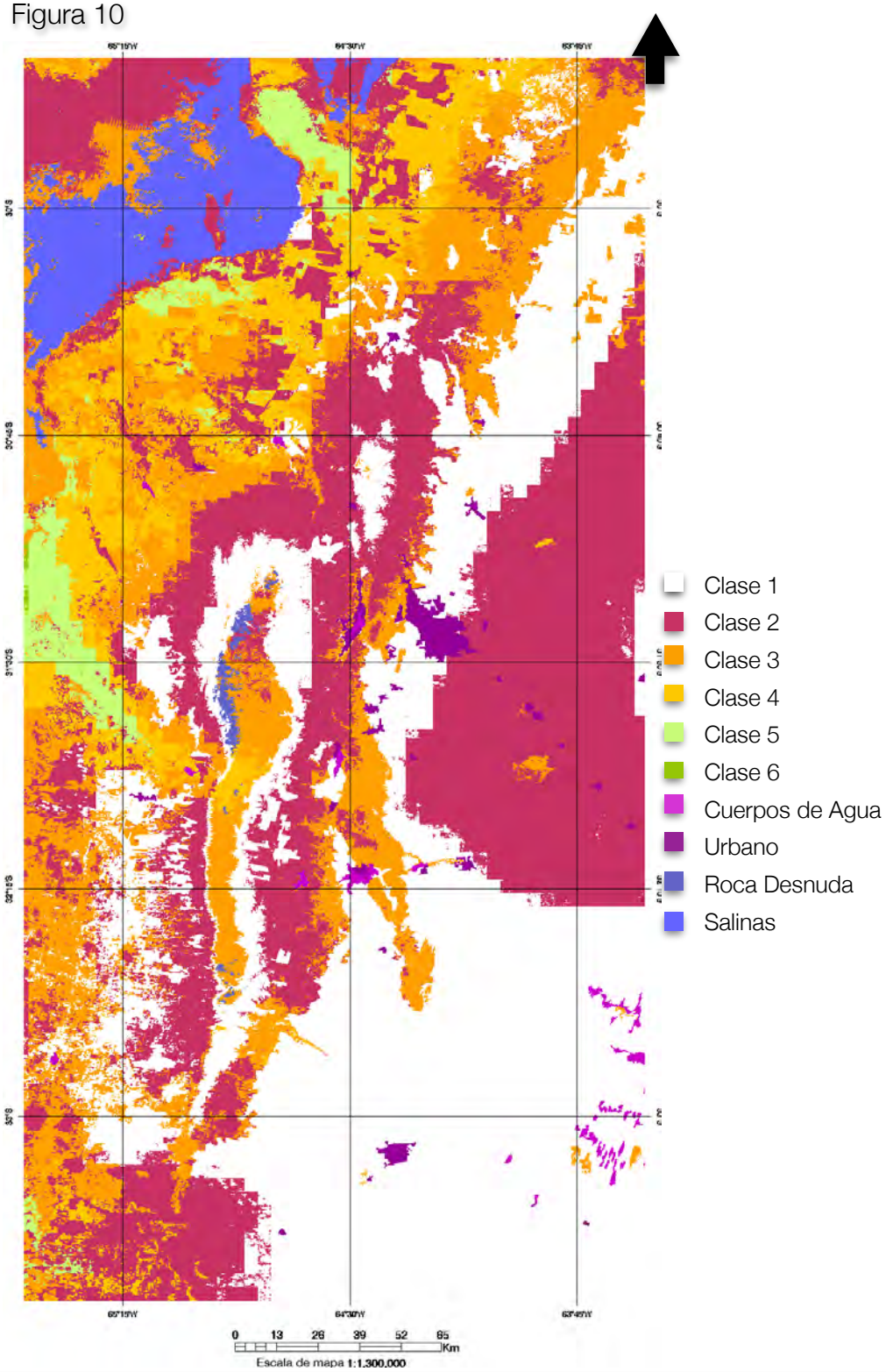
Mapa de NDVI histórico máximo. Colores más claros indican mayor valor de NDVI.

Figura 9



Secciones de mapas de zonas de amortiguación. **A** Distancias a poblados. **B** Distancias desde ríos. **C** Distancias a rutas. En todos los casos los colores más obscuros indican que el pixel se encuentra más cercano al punto de interés (río, poblado o ruta).

Figura 10



Mapa temático de clases de posibilidad de incendio para el día 5 - IX - 2013. El mapa está dividido en 6 clases, siendo la clase 1 la de menor y la clase 6 la de mayor posibilidad de incendio. Se muestran también 4 clases de zonas "no inflamables".

VII.II Conclusión

La implementación del MFPI fue lograda. Si bien no se realizo una verificación a campo, ya que excede los alcances de esta tesis, los valores mostrados en los mapas temáticos resultantes aparentarían coincidir con las condiciones de la vegetación y su propensión a desarrollar un incendio forestal, ya que se muestran con valores entres 3 y 4 zonas donde normalmente ocurren incendios. En el caso particular de la Figura 10 se puede observar que las zonas donde ocurrieron incendios en el día posterior a dicha imagen poseen valores entre 3 y 4, pero también se observa una gran superficie marcada como IPI clase 2 lo cual puede ser atribuido a que, como se menciona en el capítulo V, el hecho que las imágenes de entrada de datos meteorológicos tengan baja resolución, podría generar una homogeneización del resultado, produciendo sub o sobre estimaciones.

La inclusión de factores de riesgo, transformando el MFPI a un Índice de Riesgo de Incendio no produjo resultados que aportaran mejor información, es por ello que se decidió descartar dichos mapas. Para poder incluir este tipo de capas de información al modelo habría que, mediante trabajo de campo, cuantificar la tasa de ignición respecto a la distancia desde un camino o poblado. En el caso de los ríos, la cuantificación debería realizarse respecto si el efecto de la humedad provista por los mismos es suficiente como para aminorar el avance de un incendio o detenerlo por completo.

Esta primera implementación, mediante censado remoto, de un índice que permite estimar la posibilidad de ocurrencia de un incendio forestal en las Sierras de Córdoba de manera rápida y con bajo costo, es una herramienta que puede ser de suma utilidad a los servicios de prevención y respuesta a emergencias, ya sea tanto a nivel de cuerpos de bomberos, como de protección civil.

VII.III Consideraciones finales.

Es necesario profundizar en la generación y continuidad de un índice como el planteado en este trabajo, con el fin de lograr una herramienta de mayor precisión, que pueda servir como herramienta para el manejo y prevención de incendios para entes gubernamentales, así como también cumplir una función educativa para las poblaciones donde ocurren con mayor frecuencia este tipo de desastres. Para poder cumplir con ese objetivo deben ser revisados los siguientes puntos clave:

- Mejorar la resolución espacial de los datos meteorológicos utilizados,
- Evaluar la efectividad del índice revisando orígenes de incendios forestales en situaciones pasadas,
- Realizar una revisión histórica de los puntos de origen de ignición, respecto a asentamientos humanos y rutas para lograr una mayor exactitud en la aplicación de las zonas buffer.

Bibliografía

- Allen R., Pereira L., Raes D. y Smith M. Evapotranspiración del cultivo. Guías para la determinación de los requerimientos de agua de los cultivos. Roma: Organización de las Naciones Unidas para la Agricultura y la Alimentación - FAO, 2006. 298p
- Anderson. 1981. En: Estes J.R., Tyrl R. J. y Brunken J. N. (Eds.) Grasses and grasslands: sistematics and ecology: 297-308. University of Oklahoma Press, Norman, Oklahoma.
- Burgan R. E. y Hartford R. A.. 1993. Monitoring vegetation greenness with satellite data. General Technical Rep. INT-297, USDA Forest Service.
- Burgan R. E., Klaver R. W. y Klaver J. M. 1998. Fuel Models and Fire Potential from Satellite and Surface Observations. Int. J. Wildland Fire 8 (3): 159 —170.
- Capitanelli R.G. 1979. Clima. En Vázquez J., Miatello R. y Roque M. (eds.), Geografía Física de la provincia de Córdoba, pp. 297 – 368. Ed. Boldt, Buenos Aires.
- Ceccato, P., Flasse S. y Gregoire J. 2002a. Designing a spectral index to estimate vegetation water content from remote sensing data Part 2. Validation and applications. Remote Sensing of Environment, 82: 198 – 207.
- Ceccato P., Gobron M., Flasse S., Pinty B. y Tarantola S. 2002b. Designing a spectral index to estimate vegetation water content from remote sensing data: Part 1 Theoretical approach. Remote Sensing of Environment, 82: 188 – 197.
- Chuvieco E., Aguado I., Yebra M., Nieto H., Salas J., Martín M.P., Vilar L., Martínez F.J., Martín S., Ibarra P., De La Riva J., Baeza J., Rodríguez F., Molina J., Herrera M.A. y Zamora R. 2010. Development of a framework for fire risk assessment using remote sensing and geographic information system technologies. Ecological Modelling 221: 46 – 58.
- Díaz S., Acosta A. y Cabido M. 1994. Community structure in montane grasslands of central Argentina in relation to land use. Journal of Vegetation Science 5: 483 — 488.
- Kumar L., Skidmore A. K. y Knowles E. 1997. Modelling topographic variation in solar radiation in a GIS environment. Int. J. Geographical Innformation Science, 11 (5): 475 — 497
- Flannigana M. D., Stocksb B. J. y Wottonb B. M. 2000. Climate Change and Forest Fires. The Science of the Total Environment 262: 221 — 229.

- Fosberg M.A. y Deeming J.E. 1971. Derivation of the 1- and 10-hour time lag fuel moisture calculations of fire-danger. Fort Collins, CO: USDA, Forest Service, Rocky Mountain Forest and Range Experiment Station, Research Note RM-207.8p.
- FAO, 2006. Fire Management – Global Assessment 2006. FAO, Rome.
- Gavier G. I. y Bucher E. H. 2004. Deforestación de las Sierras Chicas de Córdoba (Argentina) en el periodo 1970 - 1997. Academia Nacional de Ciencias, Córdoba, Argentina.
- Gonçalves Z. J. y Lourenço L. 1990. Meteorological index of gorest fire risk in the portuguese mainland territory. en Proceedings of the OInternational Conference on Forest Fire Research, Coimbra, B.07 — 1/14.
- ICONA.1993. Manual de operaciones contra incendios forestales. Madrid, 5.1/65.
- Iqbal M.: An introduction to solar radiation, Academis Press Canada, Ontario, 1983.
- INMG. 1988. Nota explicativa sobre o Índice de Risco Meteotologico de Incendios Rurais. Instituto Nacional de Meteorologia e Geofisica.
- Laneve G., Jahjaha M., Ferruccib F., Hirnc B., Battazzad F. y de Bonis R. 2011. SIGRI: a national project for forest fire management. International Conference on Fire Behaviour and Risk Modelling. Alghreo, Italy.
- Luti R., Bertrán de Solís M. A., Galera M. F., Müller de Ferreira N., Berzal M., Nores M., Herrera M. A. y Barrera J. C. 1979. Vegetación. En J. Vázquez, R. Miatello y M. Roque (eds.), Geografía Física de la provincia de Córdoba, pp. 297 — 368. Ed. Boldt, Buenos Aires.
- Loboda T. V. y Csiszar A. 2007. Assessing The Risk Of Ignition In The Russian Far East Within A Modeling Framework Of Fire Threat . Ecological Applications, 17(3): 791 — 805.
- Martínez J., Vega-García C. y Chuvieco E. 2009. Human-Caused Wildfire Risk Rating for Prevention Planning in Spain. Journal of Environmental Management, 90: 1241 — 1252.
- Mallinis G., Mitsopoulos I. D., Dimitrakopoulos A. P., Gitas I. Z. y Karteris M. 2008. Local-Scale Fuel-Type Mapping and Fire Behavior Prediction by Employing High-Resolution Satellite Imagery. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 1 (4): 230 — 239.
- Miglietta S. 1994. Patrón de ocurrencia de fuegos y su efecto sobre la vegetación en el bosque Serrano de Córdoba. Tesis de Maestría, Facultad de Ciencias Exactas, Físicas y Naturales, Universidad Nacional de Córdoba, Córdoba, Argentina. 60 pp.
- MODIS Reprojection Tool User's Manual. 2011. Land Processes DAAC, USGS Earth Resources Observation and Science Center.

- Nikolakopoulos K. G., Kamaratakis E. K. y Chrysoulakis N. 2006. SRTM vs ASTER elevation products. Comparison for two regions in Crete, Greece. International Journal of Remote Sensing, 27 (21): 4819 — 4838.
- Pucheta E., Cabido M. y Días S. 1997. Modelos de estados y transiciones para los pastizales de altura de las Sierras de Córdoba, Argentina.. Ecotrópicos 10: 151 - 160.
- Reich P. B., Peterson D. W., Wedin D. A. y Wrage K. 2001. Fire and vegetation effects on productivity and nitrogen cycling across a forest-grassland continuum, Ecology, 82(6): 1703 — 1719.
- Renison D., Cingolani A. M. y Suarez R. 2002. Efectos del fuego sobre un bosquecillo de Polylepis australis (Rosaceae) en las montañas de Córdoba, Argentina.Rev. chil. hist. nat., 75 (4): 719 — 727.
- San-Miguel-Ayanz J., Barbosa P. M., Schmuck G. y Libertà G. 2003. “The European Forest Fire Information System (EFFIS),” Joint International Workshop of EARSeL SIG on Forest Fires and the GOFC/GOLD-Fire Program: Innovative Concepts and Methods in Fire Danger Esti- mation, Ghent.
- Scott J. H. y Burgan R. E. 2005. Standard fire behavior fuel models: a comprehensive set for use with Rothermel's surface fire spread model. General Technical Report RMRS-GTR-153, USDA Forest Service.
- Tucker C.J. (1979) 'Red and Photographic Infrared Linear Combinations for Monitoring Vegetation', Remote Sensing of Environment, 8(2): 127 — 150.
- Van Wagner C.E. y Pickett T.L. 1985. Equations and FORTRAN program for the Canadian Forest Fire Weather Index System. Canadian Forest Service, Ottawa, ON. Forestry Technical Report 33.
- Vermote E. F., Kotchenova S. Y. y Ray J. P.. 2011. MODIS Surface Reflectance User’s Guide. MODIS Land Surface Reflectance Science Computing Facility.
- Zak M. R. 2002. Spatial patterns of the Chaco vegetation of central Argentina: Integration of remote sensing and phytosociology. Applied Vegetation Science, 5: 213 — 226.
- Zak M., Cabido M. y Hodgson J. G. 2004. Do subtropical seasonal forests in the Gran Chaco, Argentina, have a future?. Biological Conservation, 120: 589 — 598.

Anexo I

En este anexo se transcriben los códigos del programa IRI. Todo los *scripts* están escritos respetando en lo posible las guías de programación en Python.

Etapa 0

Esta etapa verifica directorios y lanza las demás etapas.

```
#! /usr/bin/env python
"""
Created on Fri May 24 10:48:47 2013

@author: Gonzalo Quiroga
# ------------------------- Directorios de trabajo -----------------------------
'''
Solicita al usuario entrar la direccion del directorio de trabajo para crear
el string base para las direcciones de los directorios de trabajo. Cuando el
programa este funcionando hay que descomentarlos.
'''

'''
NOTA: Los modulos de GDAL para python, NumPy, marplotlib y SciPy deben estar
instalados para el funcionamiento del programa.
'''
# Librerias necesarias para realizar el mapa (no todas se usan en cada etapa)
import subprocess
import glob
from subprocess import os
import shutil
from osgeo import gdal
import numpy as np
from osgeo import osr
import datetime
import matplotlib.pyplot as plt
import scipy.interpolate
import scipy.ndimage

dirbase = os.getcwd()
print 'Este es el directorio de trabajo?: ',dirbase , '\n'
esdir=raw_input('y/n?: ')
if esdir == 'n':
   dirbase=raw_input('entre el directorio de trabajo sin apostrofes: ')

'''
NOTA: El directorio de trabajo debe contener las carpetas iniciales, con los
archivos de parametros para que funcione MRTools, con las imagenes modis a
utilizar, con la informacion meteorologica, con los mapas de riesgo, el mapa de
```

```
humedad de extincion y las imagenes de orientacion y pendiente
'''
## Directorio donde estan las imagenes hdf originales
#carpetai = 'modis_hdf'
#
## Drirectorio donde estan los parametros para MRTools
#parametros = 'parametros_mrt'

# Directorio donde estan almacenadas las imagenes
directorioi = os.path.join(dirbase,'modis_hdf')

# Directorio donde estan los archivos de parametros (.prm)
prms = os.path.join(dirbase, 'parametros_mrt')

# Directorio donde se almacenan las imagenes remuestreadas
remuesdir = os.path.join(dirbase, 'remuestreos')

# Directorio donde se almacenan las imagenes ya procesadas
ARCHIV = os.path.join(dirbase, 'archivados')

# Directorio donde se almacenan las imagenes NDVI
NDVI = os.path.join(dirbase, 'ndvi')

# Directorio donde se almacenan las imagenes recortadas procesadas
PROCESADOS = os.path.join(dirbase, 'procesados')

# Directorio donde se almacenan las imagenes remuestreadas
maxymin = os.path.join(dirbase, 'ndvi_historico')

# Directorio donde estan almacenadas las imagenes meteorologicas
dirclima = os.path.join(dirbase,'meteo')

# Directorio donde se van a almacenar las imagenes de evapotranspiracion.
direvapo = os.path.join(dirbase,'evapotranspiracion')

# Directorio donde estan las imagenes de pendiente y orientacion
dirdem = os.path.join(dirbase,'topograficos')

# Direcotio donde se guardan las imagenes de Verdor relativo
RGDIR = os.path.join(dirbase,'verdor_relativo')

VEGET = os.path.join(dirbase,'combustibles')

riskdir = os.path.join(dirbase,'riesgos')

SCRIPTDIR = os.path.join(dirbase,'iri_scripts')

FPIDIR = os.path.join(dirbase,'fpi')

# Verifica que existen todos los directorios descriptos arriba y si no existen
# los crea.
if not os.path.exists(os.path.join(dirbase, 'iri_scripts')):
   print 'Falta Carpeta con los archivos del programa'

if not os.path.exists(FPIDIR):
   os.makedirs(FPIDIR)

if not os.path.exists(direvapo):
   os.makedirs(direvapo)

if not os.path.exists(maxymin):
```

```
    os.makedirs(maxymin)

if not os.path.exists(NDVI):
    os.makedirs(NDVI)

if not os.path.exists(PROCESADOS):
    os.makedirs(PROCESADOS)

if not os.path.exists(remuesdir):
    os.makedirs(remuesdir)

if not os.path.exists(ARCHIV):
    os.makedirs(ARCHIV)

if not os.path.exists(RGDIR):
    os.makedirs(RGDIR)

# Verifica si existen archivo de parametros para el MRTools y hace que el u-
# suario elija uno de los archios prm
os.chdir(prms)
try:
    hayprm = sorted(glob.glob('*.prm'))
    hayprm[0]
except IndexError:
    print '************ No hay archivos de parametros ************\n'
    raw_input('Presione Enter para terminar')
    subprocess.sys.exit()

print '\n'
print 'Por favor ingrese el numero correspondiente al archivo de parametros \
que desea utilizar, sin comas.\n'
for i in xrange(0,len(hayprm),1):
    print i, '-', hayprm[i]

print '\n'
opcione = input('Opcion elegida: ')
prmrecorte = hayprm[opcione]
print prmrecorte


# Ejecucion de cada una da las partes para calcular el IPIM
ETAPAIGA = os.path.join(SCRIPTDIR,'Etapa_I_Mosaico_Remuestra_GA.py')
ETAPAIGQ = os.path.join(SCRIPTDIR,'Etapa_I_Mosaico_Remuestra_GQ.py')
ETAPAII = os.path.join(SCRIPTDIR,'Etapa_II_ndvi.py')
ETAPAIII = os.path.join(SCRIPTDIR,'Etapa_III_Evapotranspiracion.py')
ETAPAIV = os.path.join(SCRIPTDIR,'Etapa_IV_Verdor_Relativo.py')
ETAPAV = os.path.join(SCRIPTDIR,'Etapa_V_FPI.py')

print '\n','**** Procesando imagenes MOD09GA***'
execfile(ETAPAIGA)
print '\n', 'Terminado'
print '\n','**** Procesando imagenes MOD09GQ***'
execfile(ETAPAIGQ)
print '\n', 'Terminado'

# Obtiene los valores de proyeccion general para todas las imagenes a usar
lbase = glob.glob(os.path.join(remuesdir,'*MOD09*.tif'))
if len(lbase) < 1:
    lbase = glob.glob(os.path.join(PROCESADOS,'*MOD09*.tif'))

imabase = gdal.Open(lbase[0])
```

```
'''
Obtiene informacion (metadatos) sobre el sistema de proyeccion de la imagen.
Esto devuelve un texto (WKT).
'''
infoproyeccion = imabase.GetProjection()

'''
Obtiene datos geograficos para reproyectar la imagen (LON, delta LON,
rotation LON, LAT, rotation LAT, delta LAT)
'''
infogeografica = imabase.GetGeoTransform()

# Numero de columnas
columnas = imabase.RasterXSize

# Numero de filas
filas = imabase.RasterYSize

print '\n','**** Generando imagenes de NDVI y EWT ****'
execfile(ETAPAII)
print '\n', 'Terminado'
print '\n','**** Generando imagenes de Evapotranspiracion y Radiacion ****'
execfile(ETAPAIII)
print '\n', 'Terminado'
print '\n','**** Generando imagenes de Verdor Relativo y Porcentaje de \
Cobertura Vegetal ****'
execfile(ETAPAIV)
print '\n', 'Terminado'
print '\n','**** Generando Mapa de peligro de incendio ****'
execfile(ETAPAV)
print '\n', ‘Terminado, se obtuvo el IPI'
```

Etapa I

Esta etapa es igual para las imágenes MOD09GA como MOD09GQ, por ello aquí solo se muestra el código para porcesar uno de ambos grupos. Durante esta etapa se obtienen las bandas en formato GeoTIFF a partir de los archivos hdf originales.

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
"""
Created on Fri Feb 22 14:43:24 2013

@author: gonza
"""

#********************** Generar recortes MOD09GQ ***************************#

# Se obtienen las bandas 1, 2 y de calidad (QC)

# ---------- Comienzo del codigo para Mosaicos y Remuestreo ---------- #

# Obtiene el numero del primer tile de la lista esto permite armar los grupos
# para realizar los mosaicos
os.chdir(directorioi)
firsttile = sorted(glob.glob('*GQ*.hdf'))
```

```
os.chdir(dirbase)
haygq = True
try:
   alltiles= np.copy(firsttile)
   firsttile= alltiles[0][17:23]
except IndexError:
   print '***** No hay imagenes MOD09GQ En el directorio de trabajo *****\n'
   haygq = False

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# NOTA IMPORTANTE: Si el numero de tile encontrado justo corresponde a una    #
# imagen que esta solo una vez el programa no va a funcionar.                 #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
if haygq:
   # Genera la lista de imagenes que determina cada grupo para los mosaicos
   # basado en el numero de tile de la primera imagen de cada grupo.
   os.chdir(directorioi)
   imagenes = sorted(glob.glob('*GQ*' + firsttile + '*.hdf'))
   os.chdir(dirbase)

   # Comienzo de bloque de mosaicos
   for arch in xrange(0,len(imagenes),1):

      # Genera el archivo .prm que utiliza el Modis Tool mosaic para hacer
      #el mosaico

      imagen = imagenes[arch] # Nombre del grupo de images del mosaico
      fecha = imagen[9:16] # Fecha de las imagenes del mosaico
      # Archivos que componen el mosaico.
      archivos = sorted(glob.glob(os.path.join(directorioi,'*GQ.A' +
                  fecha + '.*.hdf')))
      # Verifica que haya por lo menos dos archivos (tile) por fecha
      if len(archivos) < 2:
         i = None
         for i in archivos:
            shutil.move(i,ARCHIV)

         continue

      # Verifica que las imagenes no hayan sido procesadas ya.
      EXISTE = glob.glob(os.path.join(remuesdir, '*GQ.A' + fecha + '.*.tif'))
      if len(EXISTE) > 0:
         i = None
         for i in archivos:
            shutil.move(i,ARCHIV)

         continue

      # Parametros para mrtmosaic
      mosaicoprm = os.path.join(prms,'TmpMosaic.prm')
      mosaicotemp = os.path.join(directorioi,'TmpMosaic.hdf')

      # Crea y escribe el archivo de parametros para mrtmosaic
      prm = open(mosaicoprm,'w+')
      for larch in range(len(archivos)):
         lista = archivos[larch]
         prm.write(lista + '\n')

      prm.close()

      # Genera el mosaico usando mrtmosaic del paquete MRTtools de MODIS
```

```
    comando = 'mrtmosaic -i ' + mosaicoprm + ' -s "0 1 1 1 0" ' \
    + ' -o ' + mosaicotemp

    '''
    NOTA: Este codigo al igual que comando2 es util para poder cambiar de
    directorios de entrada o salida facilmente # la opcion -s indica que
    bandas se van a procesar 0 no se procesa, 1 se procesa, es importante
    que la cantidad de ceros o unos sea igual a la cantidad de bandas que
    tiene la imagen, si no da error.
    '''

    # Llama a mrtmosaic como proceso del sistema
    subprocess.call(comando,shell=True)

    #nombre del archivo nuevo basado en la fecha y ubicacion del primer tile
    nuevonombre = imagen[0:23]

    # Nombre del archivo de parametros
    archprm = os.path.join(prms,prmrecorte)

    # Nombre del archivo de salida
    archout = os.path.join(remuesdir,nuevonombre)

    # Llama a resample como proceso del sistema Genera la imagen recortada
    # y remuestreada en una resolucion de 250m y coordenadas geograficas
    comando2 = 'resample -p ' + archprm + ' -i ' + mosaicotemp +  ' -o '\
            + archout + '.tif'

    subprocess.call(comando2, shell = True)

    # Limpia los archivos temporales y archiva imagenes procesadas.
    for i in archivos:
       shutil.move(i,ARCHIV)

    os.remove(os.path.join(prms, 'TmpMosaic.prm'))
    os.remove(os.path.join(dirbase,'resample.log'))
    os.remove(os.path.join(directorioi, 'TmpMosaic.hdf'))
    for todostmp2 in glob.glob(os.path.join(dirbase, 'tmp*')):
       os.remove(todostmp2)

    EXISTE = None
print '***** Todas las imagenes MOD09GQ ya fueron procesadas *****\n'
```

Etapa II

Durante esta etapa se calcula el NDVI

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
"""
Created on Thu Feb 21 15:31:00 2013

@author: Gonzalo Quiroga
"""

# Path a las imagenes GeoTiff
os.chdir(remuesdir)
```

```
firsttile = sorted(glob.glob('*.tif'))
haymod = True
try:
   firsttile = firsttile[0][-13:-4]
except IndexError:
   print '***** No hay imagenes MOD09GA En el directorio de trabajo *****\n'
   haymod = False
#   raw_input('Presione Enter para terminar')
#   subprocess.sys.exit()



if haymod:
   lista = sorted(glob.glob('*' + firsttile + '*.tif'))
   os.chdir(dirbase)
   # ------------------------- Carga de imagenes -------------------------
   for i in xrange(len(lista)):
      EXISTE = None
      fecha = lista[i][9:16] # Fecha de las imagenes del mosaico
      # Archivos que componen el mosaico
      archivos = sorted(glob.glob(os.path.join(remuesdir, '*'+ fecha +
                  '*.tif')))
      EXISTE = sorted(glob.glob(os.path.join(NDVI, '*' + fecha + '*.tif')))
      if (len(EXISTE) > 0):
         i = None
         for i in archivos:
            shutil.move(i, PROCESADOS)

         continue

      if len(EXISTE) == 0:
         print 'Comenzando bloque NDVI ' + archivos[0][-37:-21] + '\n'

         if len(archivos) == 5: #Cantidad de archivos que tiene que haber
                        # para este proceso particular.
            for banda in xrange(len(archivos)):
               if archivos[banda][-13:-11] == 'QC':
                  if archivos[banda][-32:-30] == 'GQ':
                     GQQC = banda
                  else:
                     GAQC = banda
               else:
                  if archivos[banda][-9:-6] == 'b01':
                     GQb1 = banda
                  elif archivos[banda][-9:-6] == 'b02':
                     GQb2 = banda
                  else:
                     GAb6 = banda


            # Lee la imagen GeoTiff con GDAL
            banda1 = gdal.Open(archivos[GQb1]) # Banda 1 MOD09GQ (Rojo)
            banda2 = gdal.Open(archivos[GQb2]) # Banda 2 MOD09GQ (NIR)
            banda6 = gdal.Open(archivos[GAb6]) # Banda 6 MOD09GA (SWIR)

            # Transforma el objeto GDAL en un array de numpy
            arreglo1 = banda1.GetRasterBand(1).ReadAsArray().astype(
                  'float32')
            arreglo2 = banda2.GetRasterBand(1).ReadAsArray().astype(
                  'float32')
            arreglo6 = banda6.GetRasterBand(1).ReadAsArray().astype(
```

```
        'float32')

# ------------------ Calculo de NDVI ------------------------
ndvi = ((arreglo2 - arreglo1)/(arreglo2 + arreglo1))

# --- Mascara de nubes, cuerpos de agua, sombra y datos con
# error ---

# Carga de la banda de Calidad
bandaQC = gdal.Open(archivos[GQQC]) # Banda QC MOD09GQ
bandaQC2 = gdal.Open(archivos[GAQC]) # Banda QC MOD09GA
arregloQC = bandaQC.GetRasterBand(1).ReadAsArray().astype(
        'int32')
arregloQC2 = bandaQC2.GetRasterBand(1).ReadAsArray().astype(
         'int32')

# Limpia variables que no se usan mas
bandaQC = None
bandaQC2 = None
banda1 = None
banda2 = None

ndvic = np.copy(ndvi)
for j in xrange(filas):
   for k in xrange(columnas):
      # Utilizando la banda 6 y QC de MOD09GA

      valorok = '01000000' # Parte del valor correcto de
      # bada QC para b6. El valor binario de cada pixel QC
      # MOD09GA tiene 32 posiciones de las cuales se
      # necesita conocer (de derecha a izquierda) las 2
      # primeras, la 21 a 26 y las 2 ultimas (31 y 32)

      largobinario = len(bin(arregloQC2[j, k]))

      if largobinario == 33 or largobinario == 34:
         pixbin = bin(arregloQC2[j, k])
         if len(pixbin) == 34:
            valorobs = pixbin[2:4]+pixbin[8:12]+pixbin[
                  32:34]
         elif len(pixbin) == 33:
            valorobs ='0' + pixbin[2] + pixbin[7:11] + \
            pixbin[31:33]
            largobinario = None
      else :
         valorobs = '0'
         largobinario = None

      if any([arregloQC[j, k] == 4096, arregloQC[j, k] ==
          4097]):

         # Pixeles faltantes
         if any([arreglo1[j, k] == -28672, arreglo2[j, k]
             == -28672]):
            ndvic[j, k] = -28672

         # Cuerpos de agua
         elif all([arreglo1[j, k] <= 250, arreglo2[j, k]
               <= 100, ndvi[j, k] < 0]):
            ndvic[j, k] = -28674
```

```
            # Nubes
            elif all([arreglo1[j, k] > 2500, arreglo2[j, k]
                  > 2500]):
               ndvic[j, k] = -28670

            # Correccion de sombra de nubes
            elif all([valorobs == valorok, (448-147) <
                  arreglo1[j, k] < (448+147), (832-196) <
                  arreglo2[j, k] < (832+196), (1119-287) <
                  arreglo6[j, k] < (1119+287)]):
               ndvic[j, k] = -28673
               valorobs = None

            elif all([(448-147) <= arreglo1[j, k] <= (
                  448+147), (832-196) <= arreglo2[j, k] <=
                  (832+196), (0.315-0.112) <= ndvi[j, k] <=
                  (0.315+0.112)]):
               ndvic[j, k] = -28673
               valorobs = None

            # Pixel 0K
            else :
               ndvic[j, k] = ndvic[j, k]

         # Pixel defectuoso
         else :
            ndvic[j, k] = -28671


   j = None
   k = None

   for j in xrange(1, filas-1):
      for k in xrange(1, columnas-1):

         # Corrección de valores negativos sin clasificar
         if all([ndvic[j, k] < 0, ndvic[j, k] > -28600]):
            ndvic[j, k] = -28675

         # Homogeneizado de zonas clasificadas
         if (ndvic[j-1, k-1] == ndvic[j-1, k] == ndvic[j-1,
            k+1] == ndvic[j, k+1] == ndvic[j+1, k+1] ==
            ndvic[j+1, k] == ndvic[j+1, k-1] == ndvic[j,
            k-1]):
             ndvic[j, k] = ndvic[j-1, k-1]



   # Graba la imagen de NDVI corregida en geotiff
   nombre = ('MOD09_' + fecha + '_NDVI.tif')
   driver = gdal.GetDriverByName("GTiff")
   ndviras = driver.Create(os.path.join(NDVI, nombre), columnas,
                     filas, 1, gdal.GDT_Float64)
   ndviras.SetGeoTransform(infogeografica)
   ndviras.SetProjection(infoproyeccion)
   ndviras.GetRasterBand(1).WriteArray(ndvic)
   ndviras = None

else:
   print 'Faltan imagenes en bloque ', archivos[0][-37:-21],'\n'
   j = None
```

```
    # Limpia variables antes de recomenzar

    ndvi = None
    ndvic = None
    ndvicor = None
    ndvicorar = None
    arreglo1 = None
    arreglo2 = None
    arregloQC = None
    arreglo6 = None
    j = None
    for j in archivos:
        shutil.move(j, PROCESADOS)

    print 'Terminado bloque ' + archivos[0][-37:-21] + '\n'
```

Etapa III

Cálculo de la ET_o y de la humedad de combustible muerto de 10 horas de retraso de ignición (fm10hs)

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 22 11:55:25 2013

@author: gonza
"""

os.chdir(dirbase)
# -------------------- Carga las imagenes para calcular ----------------------

# Lista las imagenes de datos topograficos.

nombrepo = prmrecorte[:-4]  # la variable prmrecorte viene de Etapa_I
imadem = sorted(glob.glob(os.path.join(dirdem, '*' + nombrepo + '*.tif')))

# Carga las imagenes de pendiente y orientacion.
for IMAGEN in xrange(len(imadem)):
    if 'orientacion' in imadem[IMAGEN]:
        orientacion = gdal.Open(imadem[IMAGEN])
    elif 'pendiente'in (imadem[IMAGEN]):
        pendiente = gdal.Open(imadem[IMAGEN])
    elif 'dem' in (imadem[IMAGEN]):
        DEM = gdal.Open(imadem[IMAGEN])


orientacion = orientacion.GetRasterBand(1).ReadAsArray().astype('float32')
pendiente = pendiente.GetRasterBand(1).ReadAsArray().astype('float32')
DEM = DEM.GetRasterBand(1).ReadAsArray().astype('float32')

# Nuevas matrices vacías de variables a calcular
RSUBA = np.empty(np.shape(orientacion))
horassol = np.empty(np.shape(orientacion))

# Constantes
```

```
pi = np.arccos(-1)
Gsc = 0.082 # Constante solar [0,0820 MJ m-2 min-1]
Krs = 0.16 # Coeficiente de ajuste para la formula de radiación de Hargreaves
asubs = 0.25    # Fracción de la radiación extraterrestre que llega a la
            # tierra en un día nublado
bsubs = 0.50
diadelanio = int(fecha[-3:]) #numero de dia del año de la imagen que se procesa

# Distancia relativa inversa tierra-sol
dr = 1 + 0.033 * np.cos(2*pi/365 * diadelanio)

# Declinacion solar
delta = 0.409 * np.sin(2*pi/365 * diadelanio - 1.39)

# Angulos utiles en radianes
noventa = 90*pi/180
quince = 15*pi/180

# Carga imagenes de datos climaticos

# Lista las imagenes de datos de temperatura, humedad relativa, velocidad del
# viento
TEMPERATURAS = sorted(glob.glob(os.path.join(dirclima,'*Tm*.tif')))
HUMRELS = glob.glob(os.path.join(dirclima,'*RH*.tif'))
VIENTOS = glob.glob(os.path.join(dirclima,'*wind*.tif'))

# ************ Datos generales de las imagenes climaticas *************
# Obtiene datos base desde las imagenes de temperatura.
IBASE = gdal.Open(TEMPERATURAS[0])

BASEARR = IBASE.GetRasterBand(1).ReadAsArray().astype('float32')

# Obtiene informacion (metadatos) sobre el sistema de proyeccion de la imagen.
# esto devuelve un texto (WKT).
CLIMPROYCC = IBASE.GetProjection()

# Obtiene datos geograficos para reproyectar la imagen
CLIMGEOTRANSFORM = IBASE.GetGeoTransform()

# Crea un objeto de referencias espaciales
CLIMREFESPACIAL = osr.SpatialReference()

# Numero de columnas
CCOLUMNAS = IBASE.RasterXSize

# Numero de filas
CFILAS = IBASE.RasterYSize

# Cantidad de imagenes
CDIM = len(TEMPERATURAS)

BASEARR = None

IBASE = None

# Crea arreglos de Latitud y longitud para los arreglos
LATC = np.empty(np.shape(orientacion))
LONC = np.empty(np.shape(orientacion))
CLIMALATC = np.empty([CFILAS, CCOLUMNAS], dtype = 'float32')*0
CLIMALONC = np.empty([CFILAS, CCOLUMNAS], dtype = 'float32')*0
```

```
# ---------------------------- Recorte y Remuestreo ----------------------------
# Como el arreglo de temperaturas tiene un tamano menor que el raster final hay
# que hubicar el pixel de radiacion dentro del de temperatura.
for fila in xrange(filas):
   for columna in xrange(columnas):
#       Obtiene los valores de latitud y longitud de cada pixel
       LONC[fila,columna] = infogeografica[0] + infogeografica[1] * columna +\
       infogeografica[2] * fila
       LATC[fila,columna] = infogeografica[3] + infogeografica[4] * columna +\
       infogeografica[5] * fila

#       Obtiene el valor de latitud y longitud dentro de la matriz de temp.
for i in xrange(CFILAS):
   for j in xrange(CCOLUMNAS):
       CLIMALONC[i,j] = CLIMGEOTRANSFORM[0] + CLIMGEOTRANSFORM[1] * j + \
       CLIMGEOTRANSFORM[2] * i
       CLIMALATC[i,j] = CLIMGEOTRANSFORM[3] + CLIMGEOTRANSFORM[4] * j + \
       CLIMGEOTRANSFORM[5] * i


# Se obtiene la posicion de los corners de la imagen de radiacion dentro de la
# de clima
ULAT = np.where(CLIMALATC[:,0]>LATC[0,0])[0][-1]
LLAT = np.where(CLIMALATC[:,0]>LATC[-1,-1])[0][-1]
LLON = np.where(CLIMALONC[0,:]<LONC[0,0])[0][-1]
RLON = np.where(CLIMALONC[0,:]<LONC[-1,-1])[0][-1]

# Funcion para hacer un cambio de tamaño en el arreglo sin interpolacion obten_
# nida de http://www.scipy.org/Cookbook/Rebinning
def rebin(a, newdims, centre=False, minusone=False):
   if not a.dtype in [np.float64, np.float32]:
       a = np.cast[float](a)

   m1 = np.cast[int](minusone)
   ofs = np.cast[int](centre) * 0.5
   old = np.array( a.shape )
   ndims = len( a.shape )
   if len( newdims ) != ndims:
       print "[congrid] dimensions error. " \
       "This routine currently only support " \
       "rebinning to the same number of dimensions."
       return None

   newdims = np.asarray( newdims, dtype='float32' )
   dimlist = []

   #     if method in ['spline']:
   oslices = [ slice(0,j) for j in old ]
   oldcoords = np.ogrid[oslices]
   nslices = [ slice(0,j) for j in list(newdims) ]
   newcoords = np.mgrid[nslices]

   newcoords_dims = range(np.rank(newcoords))
   #make first index last
   newcoords_dims.append(newcoords_dims.pop(0))
   newcoords_tr = newcoords.transpose(newcoords_dims)
   # makes a view that affects newcoords

   newcoords_tr += ofs

   deltas = (np.asarray(old) - m1) / (newdims - m1)
```

```
    newcoords_tr *= deltas

    newcoords_tr -= ofs

    newa = scipy.ndimage.map_coordinates(a, newcoords)

    return newa



# ---------------------------------- Humedad ----------------------------------
# Se cargan las imagenes de Humedad relativa
HUMA = gdal.Open(HUMRELS[0])
HUMA = HUMA.GetRasterBand(1).ReadAsArray().astype('float32')
HUMB = gdal.Open(HUMRELS[1])
HUMB = HUMB.GetRasterBand(1).ReadAsArray().astype('float32')
if any([HUMA[0,0] > HUMB[0,0], HUMA[4,4] > HUMB[4,4]]):
    HMAX = np.copy(HUMA)
    HMIN = np.copy(HUMB)
else:
    HMAX = np.copy(HUMB)
    HMIN = np.copy(HUMA)

# Lee los archivos de Temperaturas
TMAXI= gdal.Open(TEMPERATURAS[0])
TMAX = TMAXI.GetRasterBand(1).ReadAsArray().astype('float32')
TMINI = gdal.Open(TEMPERATURAS[1])
TMIN = TMINI.GetRasterBand(1).ReadAsArray().astype('float32')
#if TEMA[0,0] > TEMB[0,0]:
#    TMAX = np.copy(TEMA)
#    TMIN = np.copy(TEMB)
#else:
#    TMAX = np.copy(TEMB)
#    TMIN = np.copy(TEMA)

# Se eliminan variables que no se usan mas
HUMA = None
HUMB = None
TEMA = None
TEMB = None

# Se recorta las imagenes de Humedad relativa y Temperatura respecto a los
# corners
HMAXR = np.copy(HMAX[ULAT:LLAT+1,LLON:RLON+1])
HMINR = np.copy(HMIN[ULAT:LLAT+1,LLON:RLON+1])

TMAXR = np.copy(TMAX[ULAT:LLAT+1,LLON:RLON+1])
TMINR = np.copy(TMIN[ULAT:LLAT+1,LLON:RLON+1])

FILASR = np.shape(HMAXR)[0]
COLUMNASR = np.shape(HMAXR)[1]

# *************** Presion Media de vapor de saturacion (e sub s) **************
ES0TMAX = 0.6108 * np.exp((17.27 * (TMAXR - 273.16)) / ((TMAXR - 273.16) \
+ 237.3))
ES0TMIN = 0.6108 * np.exp((17.27 * (TMINR - 273.16)) / ((TMINR - 273.16) \
+ 237.3))
ESUBS = (ES0TMAX + ES0TMIN) / 2

# ************ Pendiente de curva de presion de Sat de Vap. (DELTA) ***********
```

```
DELTA = np.empty([FILASR,COLUMNASR])
TMEDR = (TMAXR + TMINR)/2
DELTA = (4098 * (0.6108 * np.exp((17.27 * (TMEDR - 273.16))\
      / ((TMEDR - 273.16) + 237.3))))/((TMEDR + 237.3)**2)

# ********************** Presion Real de vapor (e sub a) *********************
ESUBA = ((ES0TMIN * (HMAXR/100)) + (ES0TMAX * (HMINR/100))) / 2

# -------------------------------- Radiacion ---------------------------------
# ****************** Radiacion Extraterrestre (R sub a) **********************
for fila in xrange(filas):
   for columna in xrange(columnas):
#      Valor de Pendiente (beta) y Orientacion (Azs) para el pixel
      beta = (pi/180) * pendiente[fila, columna]
      Azs = (pi/180) * orientacion[fila, columna]
#      Obtiene los valores de latitud y longitud de cada pixel
      LON = infogeografica[0] + infogeografica[1] * columna + \
      infogeografica[2] * fila
      LAT = infogeografica[3] + infogeografica[4] * columna + \
      infogeografica[5] * fila

#      Obtiene el valor central de latitud y longitud del pixel.
      LONC = LON + (infogeografica[1] / 2.0)
      LATC = LAT + (infogeografica[5] / 2.0)

#      Se transforma la latitud a radianes Phi
      phi = (pi/180)*LATC

#      Operaciones para ahorrar tiempo
      sindelta = np.sin(delta)
      sinphi = np.sin(phi)
      sinbeta = np.sin(beta)
      sinAzs = np.sin(Azs)
      cosdelta = np.cos(delta)
      cosphi = np.cos(phi)
      cosbeta = np.cos(beta)
      cosAzs = np.cos(Azs)
      tanbeta = np.tan(beta)
      tanAzs = np.tan(Azs)
      tandelta = np.tan(delta)
      tanphi = np.tan(phi)

#      Angulo solar a la puesta del sol Omega sub s
      omegasr = np.arccos(-1*tanphi*tandelta)
      omegass = -1*omegasr

#      Correccion topografica de la radiacion segun Muhammad Iqbal (1983)

#      Angulo solar al amanecer del sol Omega sub sr prima(OSRprima) y al
#      atardecer (OSSprima)

#      Para pendientes orientadas al Ecuador (hacia el N en hemisferio sur)
      AzsG = Azs*pi/180
      if any([AzsG <= 5, AzsG >= 355 ]):
         OSRprima = np.arccos(-1 * tandelta * np.tan(phi-beta))
         OSSprima = -1 * OSRprima
#      Para pendientes orientadas al Este, Oeste o Sur
      else:
         aomegasr = (cosphi / sinAzs * tanbeta) + (sinphi / tanAzs)
         bomegasr = tandelta * ((sinphi / sinAzs * tanbeta) - \
         (cosphi / tanAzs))
```

```
#         Os = Omega Sub s
          OsPrima1 = (aomegasr ** 2) + 1
          OsPrima2 = (-1 * aomegasr * bomegasr)
          if ((aomegasr ** 2) - (bomegasr ** 2) + 1) < 0:
             OsPrima3 = np.sqrt(0.001)
          else:
             OsPrima3 = np.sqrt((aomegasr ** 2) - (bomegasr ** 2) + 1)

          OsPrima4a = np.arccos((OsPrima2 + OsPrima3)/OsPrima1)
          OsPrima4b = np.arccos((OsPrima2 - OsPrima3)/OsPrima1)

#         Eleccion de angulo horario solar puesta del sol y amanecer segun
#         si la pendiente esta orientada al este o al oeste
          if (Azs * 180 / pi) < 180: # Ladera al este
             OSRprima = OsPrima4b
             OSSprima = -1 * OsPrima4a
          else:
             OSRprima = OsPrima4a
             OSSprima = -1 * OsPrima4b

#       Cotrol numerico por errores en la formula debido a delta
#       Segun la orientacion el agunlo de salida o de puesta de la
#       superficie inclinada nunca puede ser mayor que el del plano
        OSRprimaC = min(omegasr, OSRprima)
        OSSprimaC = max(omegass, OSSprima)

#       Calculo de la Radiacion neta extraterrestre (R sub a)
#       segun las formulas de la FAO OmegaSR tiene que ser negativo y
#       OmegaSS positivo por lo cual en este punto se invierten los signos
        OSRprimaF = -1 * OSRprimaC
        OSSprimaF = -1 * OSSprimaC

#       Calcula la sumatoria de radiacion en periodos de 30' durante todas
#       las horas de sol. Quince grados cada 30 minutos.
        OmegaInicio = OSRprimaF
        OmegaFinal = OSRprimaF + quince
#       Se calcula la inclinacion para el primer perdiodo del dia.
        thetasubi = np.arccos(sindelta * sinphi * cosbeta + \
        sindelta * cosphi * sinbeta * cosAzs + \
        cosdelta * cosphi * cosbeta * np.cos(OmegaInicio) - \
        cosdelta * sinphi * sinbeta * cosAzs * np.cos(OmegaInicio) - \
        cosdelta * sinbeta * sinAzs * np.sin(OmegaInicio))
#       Si la inlcinacion solar es menor a 90º se calcula la radiacion
#       incidente (Rsuba) para la primer media hora del dia
        if (thetasubi*180/pi) < 90:
           Rsuba = (12 * 60 / pi) * Gsc * dr * (\
           ((OmegaFinal - OmegaInicio) * np.sin(phi) * np.sin(delta)) + \
           (np.cos(phi) * np.cos(delta) * \
           (np.sin(OmegaFinal) - np.sin(OmegaInicio))))
        else:
           Rsuba = 0.00

        while OmegaFinal < OSSprimaF:
#          Calcula la radiacion incidente durante el periodo.
           Rsuba0 = (12 * 60 / pi) * Gsc * dr * (\
           ((OmegaFinal - OmegaInicio) * np.sin(phi) * np.sin(delta)) + \
           (np.cos(phi) * np.cos(delta) * \
           (np.sin(OmegaFinal) - np.sin(OmegaInicio))))
#          Calcula la inclinacion solar para el periodo
           thetasubi = np.arccos(sindelta * sinphi * cosbeta + \
           sindelta * cosphi * sinbeta * cosAzs + \
```

```
        cosdelta * cosphi * cosbeta * np.cos(OmegaInicio) - \
        cosdelta * sinphi * sinbeta * cosAzs * np.cos(OmegaInicio) - \
        cosdelta * sinbeta * sinAzs * np.sin(OmegaInicio))
#        Si la inclinacion solar es menor a 90º se suma la radiacion a la
#        radiacion del periodo anterior
        if (thetasubi*180/pi) < 90:
           Rsuba = Rsuba + Rsuba0
        else:
           Rsuba = Rsuba

#        Se recalculan las variables para el proximo periodo.
        OmegaInicio = OmegaFinal
        OmegaFinal = OmegaFinal + quince

#      Termina la condicion "while" y se realiza el calculo de radiacion
#      incidente durante el ultimo periodo. Este periodo no es necesariamente
#      de 30' minutos.
#      Calcula la inclinacion solar del ultimo periodo
      thetasubi = np.arccos(sindelta * sinphi * cosbeta + \
      sindelta * cosphi * sinbeta * cosAzs + \
      cosdelta * cosphi * cosbeta * np.cos(OmegaInicio) - \
      cosdelta * sinphi * sinbeta * cosAzs * np.cos(OmegaInicio) - \
      cosdelta * sinbeta * sinAzs * np.sin(OmegaInicio))

#      Si la inclinacion es menor a 90º se procede al calculo de la
#      radiacion incidente durante el ultimo periodo.
      if (thetasubi*180/pi) <= 90:
         OmegaFinal = OSSprimaF
         Rsuba = Rsuba + (12 * 60 / pi) * Gsc * dr * (\
         ((OmegaFinal - OmegaInicio) * np.sin(phi) * np.sin(delta)) + \
         (np.cos(phi) * np.cos(delta) * \
         (np.sin(OmegaFinal) - np.sin(OmegaInicio))))
      else:
         Rsuba = Rsuba

      RSUBA[fila, columna] = Rsuba
      horasol = (OSRprimaC * 180 / pi - OSSprimaC * 180 / pi  )/15
      horassol[fila, columna] = horasol

#      Se borran los resultados de memoria.
      Rsuba = None
      horasol = None

# ********************** Radiacion Solar (R sub s) **************************

# Se cambia el tamaño de los arreglos de 10km a 250m.
TMAX250 = rebin(TMAXR,np.shape(RSUBA))
TMIN250 = rebin(TMINR,np.shape(RSUBA))

RSUBS = Krs * np.sqrt((TMAX250 - 273.16) - (TMIN250 - 273.16)) * RSUBA

# *************** Radiacion Solar en día despejado (R sub so) *****************

# Crea el arreglo RSUBSO
RSUBSO = np.empty([filas,columnas])

RSUBSO = (0.75 + (2E-5) * DEM) * RSUBA #DEM = Altitud

# ************ Radiación neta solar o de onda corta  (R sub ns) ***************

RSUBNS = np.copy(RSUBS)
```

```
RSUBNS = RSUBNS * 0.77

# ********** Radiación neta de onda larga  (R sub nl) **********
SIGMA = 4.903E-9 # Constante de Stefan-Boltzmann MJ*K^-4*m^-2*dia^-1
RSUBNLA = ((TMAX250 ** 4) + (TMIN250 ** 4)) / 2
ESUBA250 = rebin(ESUBA,np.shape(RSUBA))
RSUBNLB = 0.34 - (0.14 * np.sqrt(ESUBA250))
RSUBNLC = 1.35 * (RSUBS/RSUBSO) - 0.35
RSUBNL = SIGMA * RSUBNLA * RSUBNLB * RSUBNLC

# ********** Radiación neta  (R sub n) **********
RSUBN = RSUBNS - RSUBNL

# ---------- Viento ----------
VIENTO10M = gdal.Open(VIENTOS[0])
VIENTO10M = VIENTO10M.GetRasterBand(1).ReadAsArray().astype('float32')
ZETA = 10 #Altura de medicion de la variable
VIENTO2M = VIENTO10M * (4.87 / (np.log(67.8 * ZETA - 5.42)))
VIENTO2MR = np.copy(VIENTO2M[ULAT:LLAT+1,LLON:RLON+1])
VIENTO2M250 = rebin(VIENTO2MR,np.shape(RSUBA))

#---------- Evapotranspiracion Estandar (ET sub 0) ----------
# Constante psicometrica (obtenida de http://www.fao.org/docrep/X0490E\
# /x0490e0j.htm#annex%202.%20meteorological%20tables)
GAMMA = np.array([[0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1000,\
 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2000, 2100, 2200,\
 2300, 2400, 2500, 2600, 2700, 2800, 2900, 3000, 3000, 3100, 3200, 3300, 3400,\
 3500, 3600, 3700, 3800, 3900, 4000], [67, 67, 66, 65, 64, 64, 63, 62, 61, 61,\
 60, 60, 59, 58, 58, 57, 56, 56, 55, 54, 54, 53, 53, 52, 52, 51, 51, 50, 49,\
 49, 48, 47, 47, 47, 46, 46, 45, 45, 44, 43, 43, 42, 42, 41]])
GF = np.shape(GAMMA)[0]
GC = np.shape(GAMMA)[1]

DELTA250 = rebin(DELTA,np.shape(RSUBA))
TMED250 = rebin(TMEDR,np.shape(RSUBA))
ETSUB0A =  0.408 * DELTA250 * RSUBN

I = None
J = None
ETSUB0B = np.empty(np.shape(RSUBA))
ETSUB0D = np.empty(np.shape(RSUBA))

for I in xrange(filas):
   for J in xrange(columnas):
      ALTITUD = DEM[I,J]
      INDICES = np.where(GAMMA[0, :] <= ALTITUD)[0]
      INDICE = INDICES[-1]
      GAMMALOC = GAMMA[ 1, INDICE]
      ETSUB0B[I,J] = GAMMALOC * 900 / TMED250[I,J] * VIENTO2M250[I,J]
      ETSUB0D[I,J] = DELTA250[I,J] + GAMMALOC * (1 + 0.34 * VIENTO2M250[I,J])



ETSUB0C = rebin((ESUBS-ESUBA),np.shape(RSUBA))

ETSUB0 = (ETSUB0A + ETSUB0B * ETSUB0C)/ETSUB0D



# ---------- Graba las imagenes generadas - ----------
nombre = ('ETSUB0_' + fecha + '.tif')
```

```
driver = gdal.GetDriverByName("GTiff")
evaporas = driver.Create(os.path.join(direvapo,nombre), columnas, \
filas, 1, gdal.GDT_Float64)
evaporas.SetGeoTransform(infogeografica)
evaporas.SetProjection(infoproyeccion)
evaporas.GetRasterBand(1).WriteArray(ETSUB0)
evaporas = None

nombre2 = ('RSUBA_' + fecha + '.tif')
driver = gdal.GetDriverByName("GTiff")
rsubaras = driver.Create(os.path.join(direvapo,nombre2), columnas, \
filas, 1, gdal.GDT_Float64)
rsubaras.SetGeoTransform(infogeografica)
rsubaras.SetProjection(infoproyeccion)
rsubaras.GetRasterBand(1).WriteArray(RSUBA)
rsubaras = None

# ------------------------ Calculo del EMC y fm10hs --------------------------#
HMINR250 = rebin(HMINR,np.shape(RSUBA))

i = None
j = None
EMC = np.empty(np.shape(RSUBA))
for i in xrange(filas):
   for j in xrange(columnas):
      if HMINR250[i, j] > 50:
         EMC[i, j] = (21.0606 + 0.005565*(HMINR250[i, j]**2) -
                  0.00035*HMINR250[i, j]*TMAX250[i, j] -
                  0.483199*HMINR250[i, j])
      elif HMINR250[i, j] < 10:
         EMC[i, j] = (0.03229 + 0.281073*HMINR250[i, j] -
                  0.000578*HMINR250[i, j]*TMAX250[i, j])
      else:
         EMC[i, j] = (2.22749 + 0.160107*HMINR250[i, j] -
                  0.014784*TMAX250[i, j])



fm10hs = 1.28 * EMC

#fm10hsfrac = gdal.Open(glob.glob(os.path.join(VEGET,'fm10hsfrac.tif'))[0])
#fm10hsfrac = fm10hsfrac.GetRasterBand(1).ReadAsArray().astype('float32')

# ---------------------- Graba las imagenes generadas - ----------------------
nombre2 = ('fm10hs_' + fecha + '.tif')
driver = gdal.GetDriverByName("GTiff")
fm10hsfras = driver.Create(os.path.join(VEGET,nombre2), columnas, \
filas, 1, gdal.GDT_Float64)
fm10hsfras.SetGeoTransform(infogeografica)
fm10hsfras.SetProjection(infoproyeccion)
fm10hsfras.GetRasterBand(1).WriteArray(fm10hs)
fm10hsfras = None
```

Etapa IV

Cálculo del Verdor Relativo

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
"""
Created on Thu May 23 15:17:26 2013

@author: gonza
"""
# Carga imagen del NDVI del dia.
ndvidia = gdal.Open((glob.glob(os.path.join(NDVI, '*' + fecha + '*.tif')))[0])
ndvihist = sorted(glob.glob(os.path.join(maxymin, '*.tif')))
ndvimax = gdal.Open(ndvihist[0])
ndvimin = gdal.Open(ndvihist[1])
ndvimaxarr = ndvimax.GetRasterBand(1).ReadAsArray().astype('float32')
ndviminarr = ndvimin.GetRasterBand(1).ReadAsArray().astype('float32')
ndvidiaarr = ndvidia.GetRasterBand(1).ReadAsArray().astype('float32')
modelos = glob.glob(os.path.join(VEGET,'Clases_SVM.tif'))
modelosI = gdal.Open(modelos[0])
arraymo = modelosI.GetRasterBand(1).ReadAsArray().astype('int')
fecha = None
fecha = sorted(glob.glob(os.path.join(NDVI, '*.tif')))[-1][-16:-9]
# Calcula el verdor relativo
verrelSC = ((ndvidiaarr - ndviminarr)/(ndvimaxarr - ndviminarr))*100
RESTA = ndvimaxarr - ndviminarr
verrel = np.copy(verrelSC)
verrel[RESTA == 0] = 0
verrel[verrelSC < 0] = 0
verrel[verrelSC > 100] = 100
verrel[ndvidiaarr < -27999] = float('nan')
verrelSC = None


# Obtiene el valor esperado de verdor relativo para los pixeles faltantes/nubes

# Calcula el verdor relativo promedio por clase y genera un arreglo lineal
# con una entrada para cada promedio por clase
vrm = []

vr1 = verrel[np.where(arraymo == 1)]
vr1m = np.nanmean(vr1)
vrm.append(vr1m)

vr2 = verrel[np.where(arraymo == 2)]
vr2m = np.nanmean(vr2)
vrm.append(vr2m)

vr3 = verrel[np.where(arraymo == 3)]
vr3m = np.nanmean(vr3)
vrm.append(vr3m)

vr4 = verrel[np.where(arraymo == 4)]
vr4m = np.nanmean(vr4)
vrm.append(vr4m)

vr5 = verrel[np.where(arraymo == 5)]
vr5m = np.nanmean(vr5)
vrm.append(vr5m)

vrm.append(float('NaN'))

vrm.append(float('NaN'))
```

```
vr8 = verrel[np.where(arraymo == 8)]
vr8m = np.nanmean(vr8)
vrm.append(vr8m)

vr9 = verrel[np.where(arraymo == 9)]
vr9m = np.nanmean(vr9)
vrm.append(vr9m)

vr10 = verrel[np.where(arraymo == 10)]
vr10m = np.nanmean(vr10)
vrm.append(vr10m)

vrm.append(float('NaN'))

vr12 = verrel[np.where(arraymo == 12)]
vr12m = np.nanmean(vr12)
vrm.append(vr12m)

# Estima el valor de verdor relativo para el pixel con nube o defectuoso
for i in xrange(filas):
   for j in xrange(columnas):
      if all([ndvidiaarr[i,j] < -27999, arraymo[i,j] < 13]):
         ndvidiae = (vrm[arraymo[i,j] - 1]/100)*ndvimaxarr[i,j]
         verrel[i,j] = ((ndvidiae - ndviminarr[i,j])/(ndvimaxarr[i,j]
                  - ndviminarr[i,j]))*100


verrel[RESTA == 0] = 0 # Ultima corrección de los valores que resultan de di-
                  # vidir por cero.

nombre2 = ('RG_' + fecha + '.tif')
driver = gdal.GetDriverByName("GTiff")
RGras = driver.Create(os.path.join(RGDIR,nombre2), columnas, \
filas, 1, gdal.GDT_Float64)
RGras.SetGeoTransform(infogeografica)
RGras.SetProjection(infoproyeccion)
RGras.GetRasterBand(1).WriteArray(verrel)
RGras = None
verrel = None
```

Etapa V

Cálculo del MFPI

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 12 12:40:27 2013

@author: gonza
"""
 # ------------- Calculo del FPI segun WFAS 2013 y Laneve 2011 --------------
# http://goo.gl/FwJLeI

ndvidia = gdal.Open((glob.glob(os.path.join(NDVI, '*' + fecha + '*.tif')))[0])
ndvidiaarr = ndvidia.GetRasterBand(1).ReadAsArray().astype('float32')

fm10hsl = gdal.Open(os.path.join(VEGET,'fm10hs_'+ fecha +'.tif'))
```

```
fm10hsarr = fm10hsl.GetRasterBand(1).ReadAsArray().astype('float32')

moel = glob.glob(os.path.join(VEGET,'MOE.tif'))
moe = gdal.Open(moel[0])
moearr = moe.GetRasterBand(1).ReadAsArray().astype('int')

ndvihist = sorted(glob.glob(os.path.join(maxymin, 'maximos_*NDVI.tif')))
ndvimaxl = gdal.Open(ndvihist[0])
ndvimaxarr = ndvimaxl.GetRasterBand(1).ReadAsArray().astype('float32')

verrell = gdal.Open(sorted(glob.glob(os.path.join(RGDIR, 'RG_'+ fecha +
        '.tif')))[-1])
verrelarr = verrell.GetRasterBand(1).ReadAsArray().astype('float32')

fm10hsfrac = np.array(np.shape(moearr), dtype='float32')
NDmx = np.array(np.shape(moearr), dtype='float32')
LRmx = np.array(np.shape(moearr), dtype='float32')
verrelfrac = np.array(np.shape(moearr), dtype='float32')
Lf = np.array(np.shape(moearr), dtype='float32')

fm10hsfrac = (fm10hsarr - 2)/(moearr - 2)
NDmx = (ndvimaxarr*100) + 100
LRmx = 35 + 40 * (NDmx  - 100)/ 80
verrelfrac = verrelarr/100
Lf = verrelfrac*LRmx/100

FPI = np.array(np.shape(moearr), dtype='float32')

FPI = (1 - Lf)*(1 - fm10hsfrac)*100
FPI[FPI < 0] = 0
FPI[FPI > 100] = 100
# ----------------------------- Escala de riesgo -------------------------------
FPIescala = np.zeros(np.shape(FPI), dtype='float32') # Arreglo vacio

# Calsificador
clases_de_riesgo = [ 1, 2, 3, 4, 5]
intervalos_fpi = [ -1, 21, 41, 56, 71, 100]
clases_no_incendiables = [ 101, 102, 103, 104]
clases_no_risk = [ 8, 9, 10, 11]

# Clases de riesgo
for i in range(len(clases_de_riesgo)-1):
   FPIescala = FPIescala + clases_de_riesgo[i] * np.logical_and(FPI >=
                  intervalos_fpi[i], FPI < intervalos_fpi[i + 1] )

FPIescala = FPIescala + clases_de_riesgo[i + 1] * np.logical_and(FPI >=
        intervalos_fpi[i + 1], FPI <= intervalos_fpi[i + 2] )

# Clases no incendiables
for i in range(len(clases_no_incendiables)):
   FPIescala[moearr == clases_no_incendiables[i]] = clases_no_risk[i]


# ------------------- Reevalua cada pixel segun Laneve 2011 ------------------
ETSUB0 = gdal.Open(sorted(glob.glob(os.path.join(direvapo, 'ETSUB0*.tif')))
             [-1])
evapot = ETSUB0.GetRasterBand(1).ReadAsArray().astype('float32')
fpi5 = []
fpi4 = []

# Valores de evapotranspiracion para de las clases 4 y 5
```

```
fpi5.append( evapot[ FPIescala == 5])
fpi4.append( evapot[ FPIescala == 4])

# Media y Desvio Estandar de evapotranspiracion de las clases 4 y 5
fpi5M = np.mean(np.ma.masked_array(fpi5,np.isnan(fpi5)))
fpi5SD = np.std(np.ma.masked_array(fpi5,np.isnan(fpi5)))
fpi4M = np.mean(np.ma.masked_array(fpi4,np.isnan(fpi4)))
fpi4SD = np.std(np.ma.masked_array(fpi4,np.isnan(fpi4)))

# Correccion de la escala de riesgo teniendo en cuenta la evapotranspiracion
FPIevapcor = np.copy(FPIescala)

FPIevapcor[np.logical_and( FPIescala == 5, evapot > ( fpi5M + fpi5SD))] = 6

FPIevapcor[np.logical_and( FPIescala == 5, evapot < ( fpi5M - fpi5SD))] = 4

FPIevapcor[np.logical_and( FPIescala == 4, evapot > ( fpi4M + fpi4SD))] = 5

FPIevapcor[np.logical_and( FPIescala == 4, evapot < ( fpi4M - fpi4SD))] = 3

# Clases no incendiables
for i in range(len(clases_no_incendiables)):
   FPIevapcor[moearr == clases_no_incendiables[i]] = clases_no_risk[i]

## ---------------------- Graba las imagenes generadas - ----------------------

nombre2 = ('fpi_' + fecha +  '.tif')
driver = gdal.GetDriverByName("GTiff")
FPIecras = driver.Create(os.path.join(FPIDIR,nombre2), columnas, \
filas, 1, gdal.GDT_Float64)
FPIecras.SetGeoTransform(infogeografica)
FPIecras.SetProjection(infoproyeccion)
FPIecras.GetRasterBand(1).WriteArray(FPIevapcor)
FPIecras = None
```