**Fondo Sectorial para la Investigación, el Desarrollo y la Innovación Tecnológica Forestal CONAFOR-CONACYT**

**PROYECTO "DESARROLLO DE UN SISTEMA DE PREDICCIÓN DE PELIGRO DE INCENDIOS FORESTALES PARA MEXICO"**

**Clave de registro**: 252620


**Etapa: 3 de 3.**

**ANEXO 3- MANUAL DE MANTENIMIENTO DEL SISTEMA DE PREDICCIÓN DE PELIGRO DE INCENDIOS FORESTALES PARA MEXICO.**

**Versión 1.0 Septiembre 2018.**

Autores: Jaime Briseño, Daniel J. Vega, Norma Monjarás-Vega, Carlos Briones, Eric Calleros, Maria Guadalupe Nava Miranda, Pablito Marcelo López Serrano, Javier Corral.

# CONTENIDOS.

## 1. REQUERIMIENTOS TÉCNICOS DEL SISTEMA.

El **Sistema de predicción de peligro de incendios forestales para México** ha sido desarrollado en una plataforma para su funcionamiento a través de internet, por lo cual es necesario que el equipo en el que se hospeda el sistema (Servidor) cuente con los siguientes requisitos para su correcto funcionamiento:

### Requisitos de Hardware

Equipo con arquitectura de servidor con las siguientes características:

| | Requerimiento mínimo | Recomendado |
|---|---|---|
| Procesador | Intel Xeon de 2.6GHz | Intel Xeon de 2.6GHz |
| Memoria RAM | 32 GB | 64GB |
| Espacio disponible en disco | 500GB | 2 TB |
| Conexión a internet | 20 MB de uso exclusivo para el sistema | 40 MB de uso exclusivo para el sistema |

### Requisitos de Software

| **Software** | **Funcionalidad** |
|---|---|
| Microsoft Windows 2008 Server R2 o superior | Sistema Operativo usado de Plataforma para el uso de las aplicaciones. |
| Servidor WEB Apache v 2.X | Servidor WEB |
| PHP 7 | Lenguaje de programación del lado del servidor |
| MariaDB 10 | Servidor de base de datos |
| Lenguaje Python 2.7 | Lenguaje de programación del lado del servidor. |
| PostgreSQL 9.5 o superior | Servidor de base de datos geográfica |
| Geoserver 2.8.X | Servidor de mapas |
| ArcGis v. 10.0 con licencia de Spatial analyst | Sistema de información geográfica. Utilizado para el geoprocesamiento. |
| Sistema Operativo Linux CenOS 7 | Sistema operativo |
| Compilador gcc y g++ | Lenguaje de programación para Linux |
| Bibliotecas de gdal | Operaciones de geoprocesamiento |
| Interpretes csh, bash, ksh, python | Auxiliares para programación a nivel sistema operativo |

## 2. INSTALACIÓN DEL SISTEMA Y ESTRUCTURA DE ARCHIVOS

### 2.1. INSTALACIÓN.

**Entrega de información**

La información se ha puesto disponible en un servidor FTP con las siguientes carpetas:

| Nombre de la carpeta | Contenido |
|---|---|
| *proceso_diario* | Toda la Información necesaria para la actualización diaria del sistema |
| *incendios_predichos* | Información del módulo de número de incendios predichos por estado |
| *incendios_activos* | Información sobre la capa de incendios activos que provee CONAFOR diariamente. |
| *pagina_web* | Información de la página Web |

Se recomienda que las carpetas "*proceso_diario*", "*incendios_predichos*" e "*incendios_activos*" se encuentre en la misma carpeta, por ejemplo:

- "D:\proyectos\incendios\proceso_diario"
- "D:\proyectos \incendios\incendios_predichos"
- "D:\proyectos\incendios\activos"

La carpeta "*pagina_web*" contiene la carpeta "*incendios*" la cual se recomienda sea instalada de manera separada dentro de la carpeta pública del servidor web, por ejemplo "~*htdocs**incendios*".

**Proceso diario**

La carpeta "*proceso_diario*" contiene la carpeta "*py*" dentro de la cual se encuentra el archivo "*incendios_proceso_diario.py*", las líneas 21 a la 23 contiene lo siguiente:

```
rutaBase = "D:/proyectos/incendios/proceso_diario"
rutaBasePaginaWEB = "C:/xampp/htdocs/incendios"
rutaFTP = "D:/ftp/conabio_incendios/data"
```

| **Variable** | **Contenido** |
|---|---|
| rutaBase | Es la ruta donde se copiará el contenido de la carpeta "*proceso_diario*" |
| rutaBasePaginaWEB | Ruta completa pública del servidor web donde se encuentra la carpeta "incendios". |
| rutaFTP | Ruta del servidor FTP donde la CONABIO deposita diariamente en forma automática información. |

La línea 95 contiene la definición de la variable "cmd_gdal", en la cual debe especificarse la ruta donde se ha instalado el ejecutable "*gdal_translate.exe*" del software "*OSGeo4W64*", ejemplo: "*C:/OSGeo4W64/bin/gdal_translate.exe*".

**Incendios predichos por estado**

La carpeta "incendios*_predichos*" contiene contiene la carpeta "*py*" dentro de la cual se encuentra el archivo "*proceso_diario_NINC_PRED.py*", las líneas 24 y la 25 contiene lo siguiente:

rutaBase = "D:/proyectos/incendios/proceso_diario/incendios_predichos"
rutaBasePaginaWEB = "C:/xampp/htdocs/incendios"

| Variable | Contenido |
|---|---|
| rutaBase | Carpeta del servidor donde se encuentra la carpeta "*incendios_predichos*". |
| rutaBasePaginaWEB | Ruta completa pública del servidor web donde se encuentra la carpeta "incendios". |

**Incendios activos**

La carpeta "incendios*_activos*" contiene contiene la carpeta "*py*" dentro de la cual se encuentra el archivo "*incendios_activos.py*", las líneas 20 a la 24 contiene lo siguiente:

rutaInFTP = "D:/ftp/conafor_incendios"
rutaInAlmacenKMLs = "D:/proyectos/incendios/incendios_activos/in/KML"
rutaInAlmacenShapes = "D:/proyectos/incendios/incendios_activos/in/shapes"
rutaOutWEB = "C:/xampp/htdocs/incendios/cartografia/capas/conafor/shape"
rutaTMP = "D:/proyectos/incendios/incendios_activos/tmp"

| Variable | Contenido |
|---|---|
| rutaInFTP | Carpeta del servidor FTP donde la CONAFOR deposita diariamente el archivo KML con los incendios diarios. |
| rutaInAlmacenKMLs | Carpeta en la carpeta de datos del proyecto donde se genera un respaldo del archivo KML depositado por CONAFOR. |
| rutaInAlmacenShapes | Carpeta donde se almacenan los archivos SHAPE generados a partir del archivo KML. |
| rutaOutWEB | Ruta donde se encuentran los archivos shape publicados en la página WEB. |
| rutaTMP | Carpeta temporal para el proceso de conversión del archivo KML a Shape. El contenido de esta carpeta se limpia y actualiza constantemente. |

**Página Web**

La carpeta "*pagina_web*" contiene la carpeta "incendios", la cual debe ser colocada en la carpeta pública del servidor web. Ejemplo: "~htdocs/incendios".

Base de datos MySQL (MariaDB)

Dentro de la carpeta principal del sistema se encuentra la carpeta "*bdatos*" que contiene el archivo "*incendios.sql*" y que deberá ejecutarse en el servidor de base de datos. Este script generará la base de datos y creará las tablas necesarias para el funcionamiento del sistema.

En las líneas 10 y 11 del archivo "~clases/pdo_mysql.class.php" se deberá especificar el usuario y la clave de la base de datos.

La carpeta "~include/constantes.php" contiene rutas de archivos necesarios para el funcionamiento del sistema, como lo son las líneas de la 62 a la 68, donde se debe especificar la ruta donde se encuentra el ejecutable de Python.

En las líneas 8 y 9 del archivo "~include/ conexion_geoserver.php" deberá especificarse el usuario y la contraseña de la base de datos del servidor de geoserver.

En las líneas 10 y 11 del archivo "~include/ conexion_postgres_conectar.php" se deberá indicar el usuario y la clave de la conexión al servidor de postgres.

## Consulta de información en línea

Para la descarga y consulta en línea de la información publicada en el sitio Web de incendios, existe un acceso disponible a través de un servicio FTP (File transfer service).

Para acceder a la información disponible en un servidor FTP es necesario hacerlo a través de un software "cliente" que permite la consulta y descarga de archivos.

El software del cliente FTP más comúnmente utilizado por ser gratuito, así como por su facilidad de uso es "FileZilla", el cual puede descargarse desde el enlace:

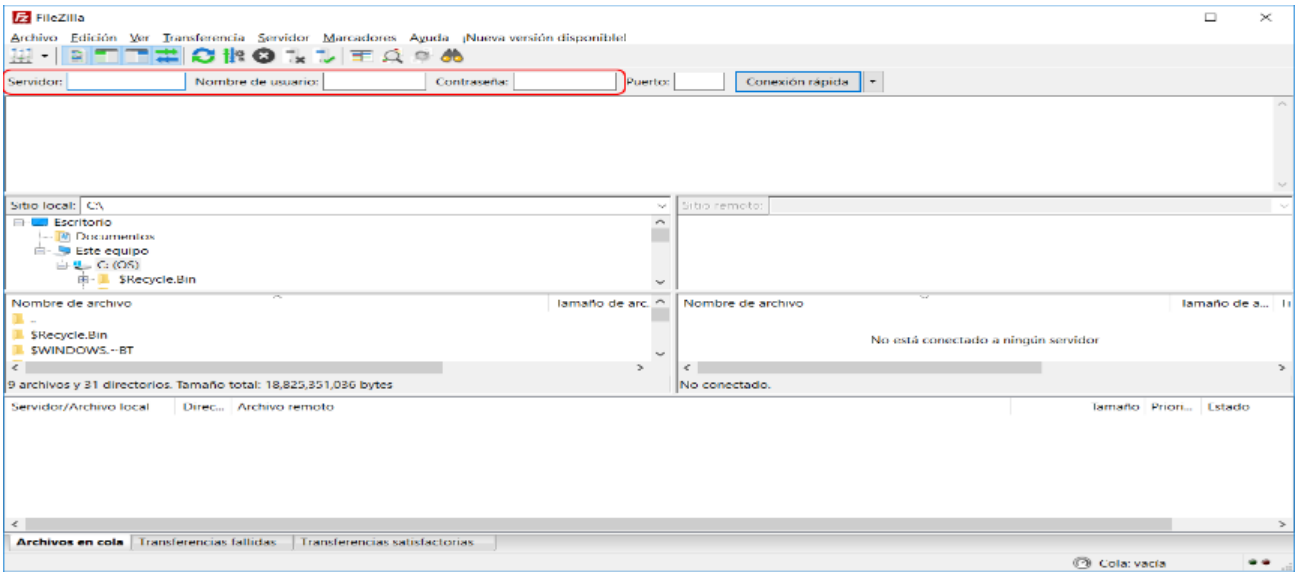
https://filezilla-project.org/download.php?type=client

Después de instalar el software será necesario proporcionar los siguientes datos para acceder al sitio donde se encuentra disponible la información del sitio web:

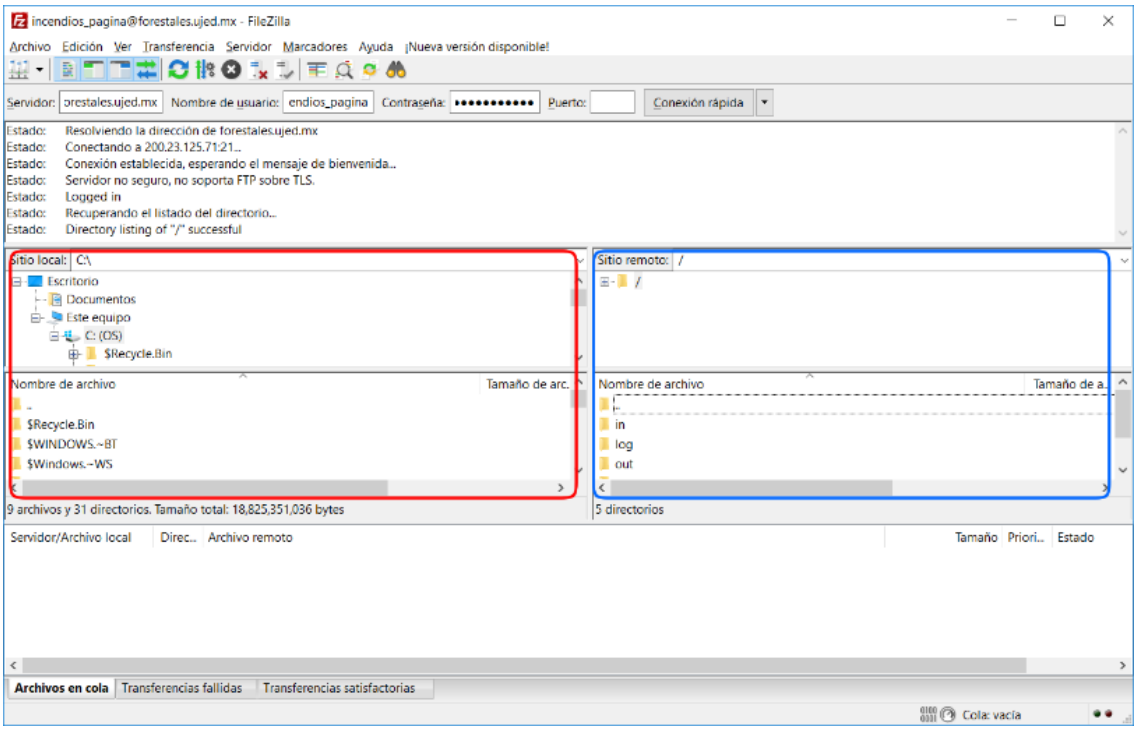
Servidor: forestales.ujed.mx

Nombre del usuario: incendios_pagina

Contraseña: Incendios.Diario




Después de ingresar al sitio, se mostrarán dos secciones, en el lado izquierdo se mostrará el contenido de las carpetas del equipo local (marco rojo), del lado derecho se mostrarán los archivos y carpetas del sitio remoto.



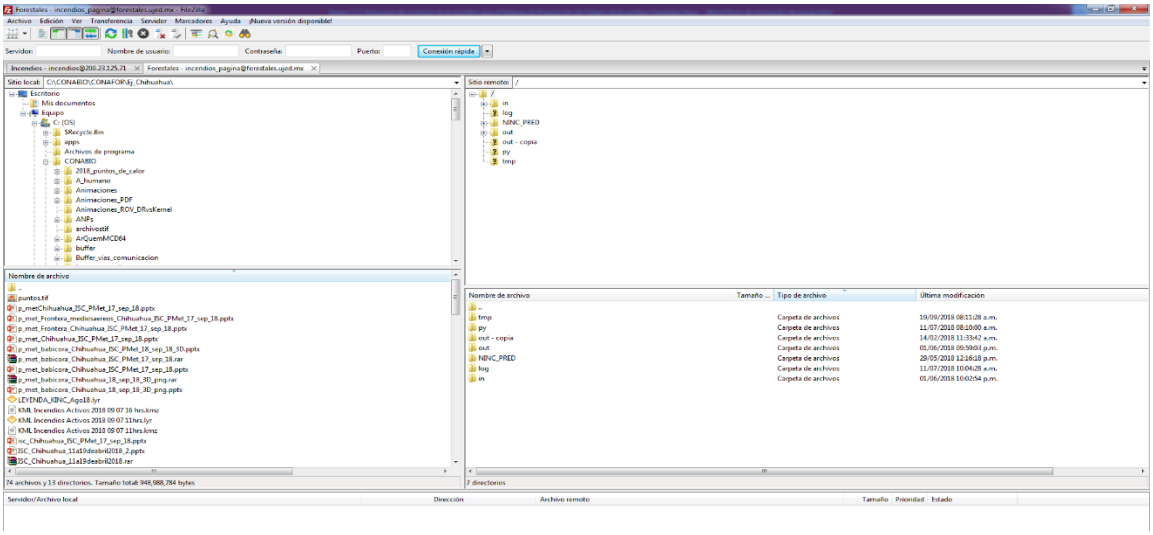
2.2. **ESTRUCTURA DE ARCHIVOS EN EL SITIO FTP.**

**IN :** Archivos de entrada; **OUT**: archivos de salida

**NINC**: cálculo del número de incendios por estado.

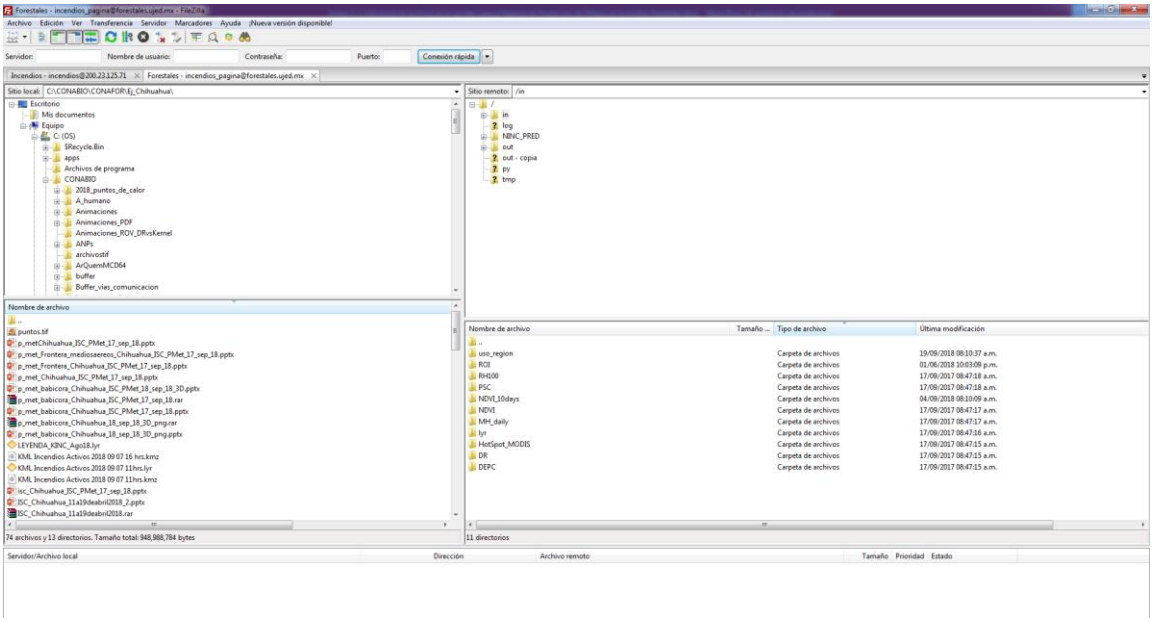
**PY:** archivos *phyton* con las operaciones del Sistema.

**TMP:** archivos temporales del Sistema.



**Ruta base**: "D:/proyectos/incendios/proceso_diario"

## IN (ENTRADAS):



**Descripción de la estructura de archivos en el Sistema de Peligro:**

**IN:**

**Capas de entrada diarias:**

1. **Hostspot_MODIS** (Puntoscalor.shp)
2. **MH_daily (**LM100Hañodiajuliano.bin y .hdr)
3. **NDVI_10days (**compositeNDVIañomessemana_merge.tiff)

**Capas de entrada fijas:**

4. **Uso_region** (uso_region.tiff)
5. **NDVI** (NDVImin.tiff, NDVIdiff.tiff)
6. **RH100** (H100_HMIN.tiff, H100_HMAX.tiff, H100_mask_NDVI_1.tiff)
7. **DR** (LR_max.tiff)
8. **PSC** (PSCCoeficientes.csv)
9. **DEPC** (FDIac_30ecuaciones.csv, tabla_c.csv)
10. **ROI** (ROH100.tif, mask_cult.tif)

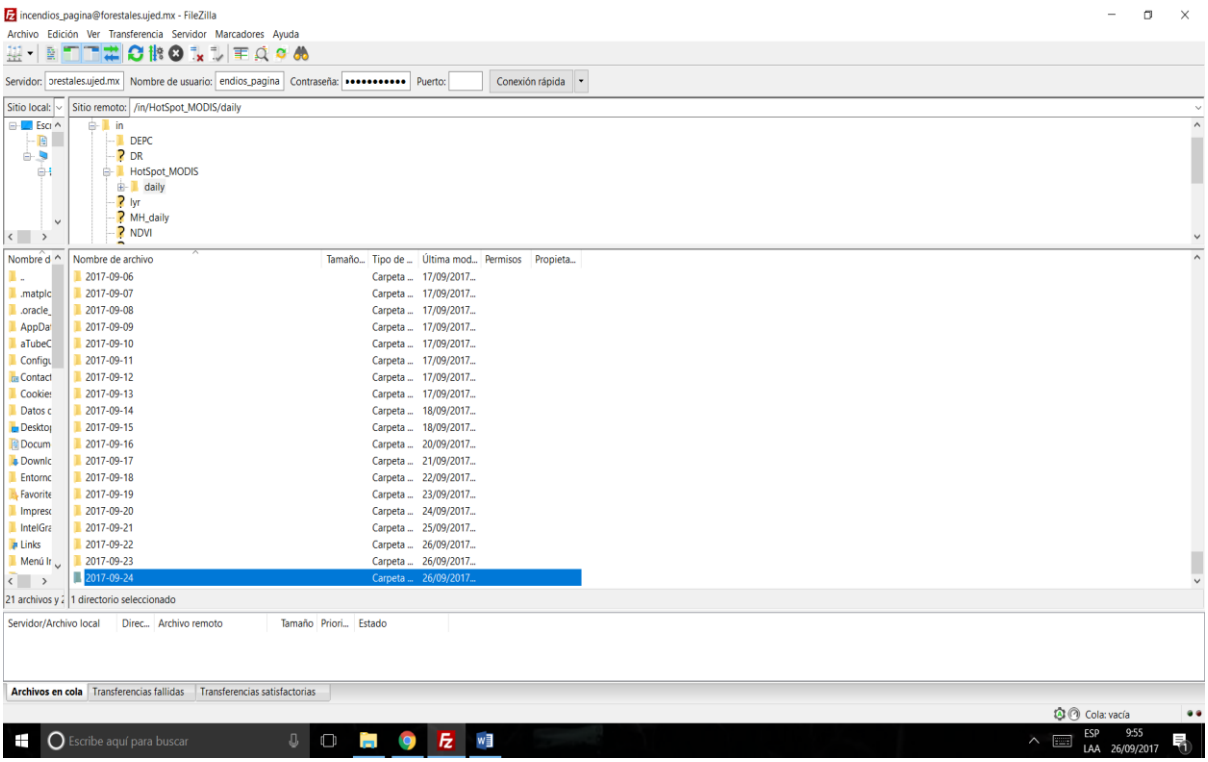
**Capas de entrada diarias:**

1. **Hostspot_MODIS** (Puntoscalor.shp)

**Ruta:** "PuntosMODIS": rutaBase + "/in/HotSpot_MODIS/daily"

**Contenido:** Contiene el shapefile de los puntos de calor MODIS observados en CONABIO.

Cada archivo de puntos de calor diario se encuentra en una carpeta con la fecha diaria.



2. **MH_daily (**LM100Hañodiajuliano.bin y .hdr)

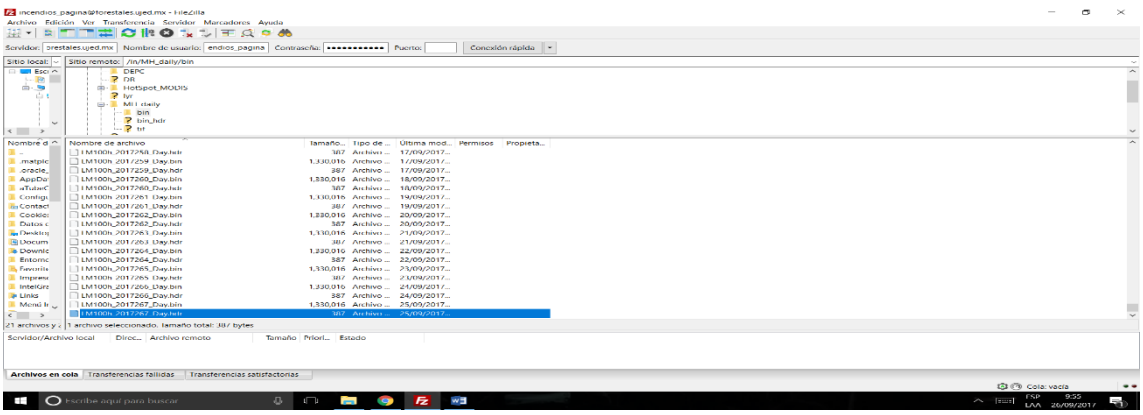
**Ruta:** "LM100bin": rutaBase + "/in/MH_daily/bin",

"LM100binHDR": rutaBase + "/in/MH_daily/bin_hdr/LM100h_Day.bin.hdr",

**Contenido:** Imagen diaria de Humedad del combustible de 100h de CONABIO.

Se recibe diariamente un archivo en formato .bin y un .hdr de CONABIO.

Con nombre **LM100H*añodiajuliano*.bin y .hdr**

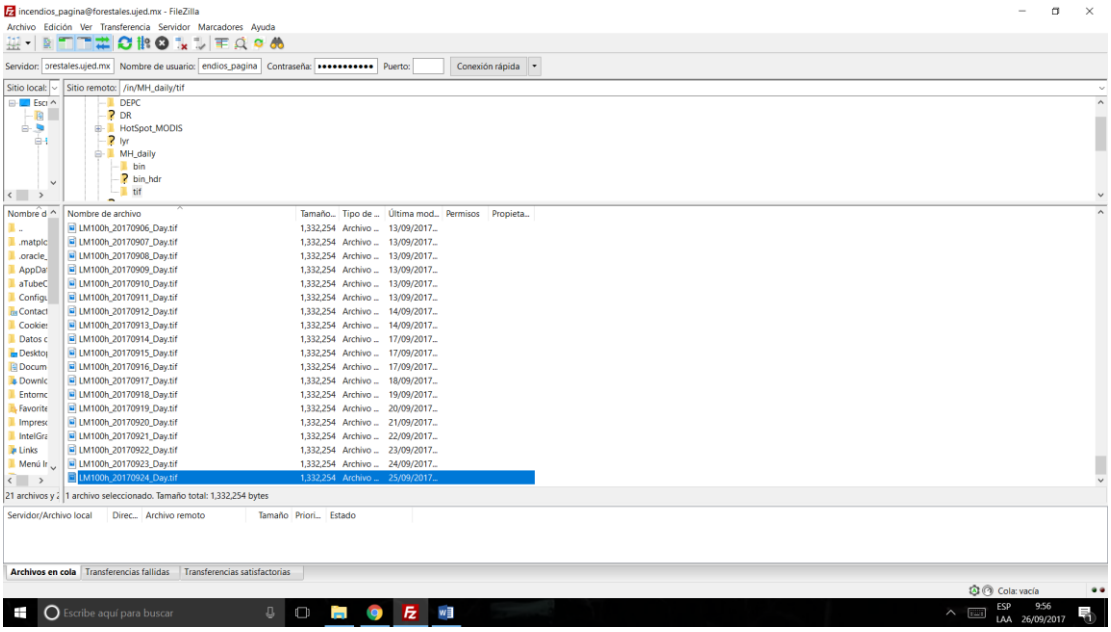


Este archivo LM100h se convierte a .tiff:

**Ruta:** "LM100tif": rutaBase + "/in/MH_daily/tif",

**Contenido:** Imagen diaria de Humedad del combustible de 100h de CONABIO en formato .tiff.

Con nombre **LM100H*añodiajuliano*.tiff**

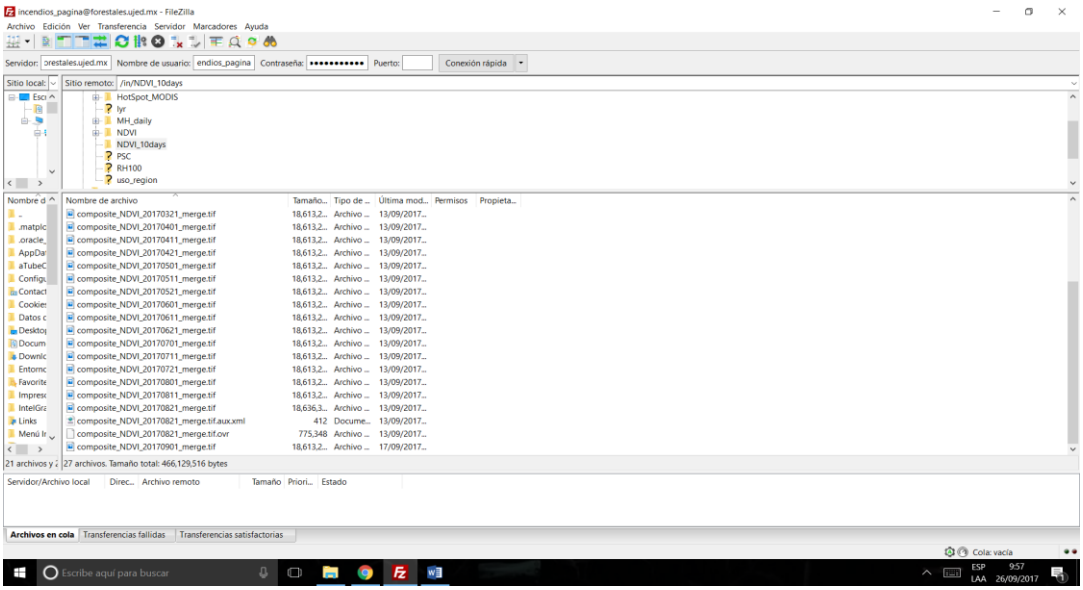


3. **<u>NDVI_10days:</u>**

**Ruta:** "NDVI_10days": rutaBase + "/in/NDVI_10days",

**Contenido:** Contiene el archivo .tiff del compuesto de índice de verdor de vegetación NDVI de diez días proporcionado por CONABIO.

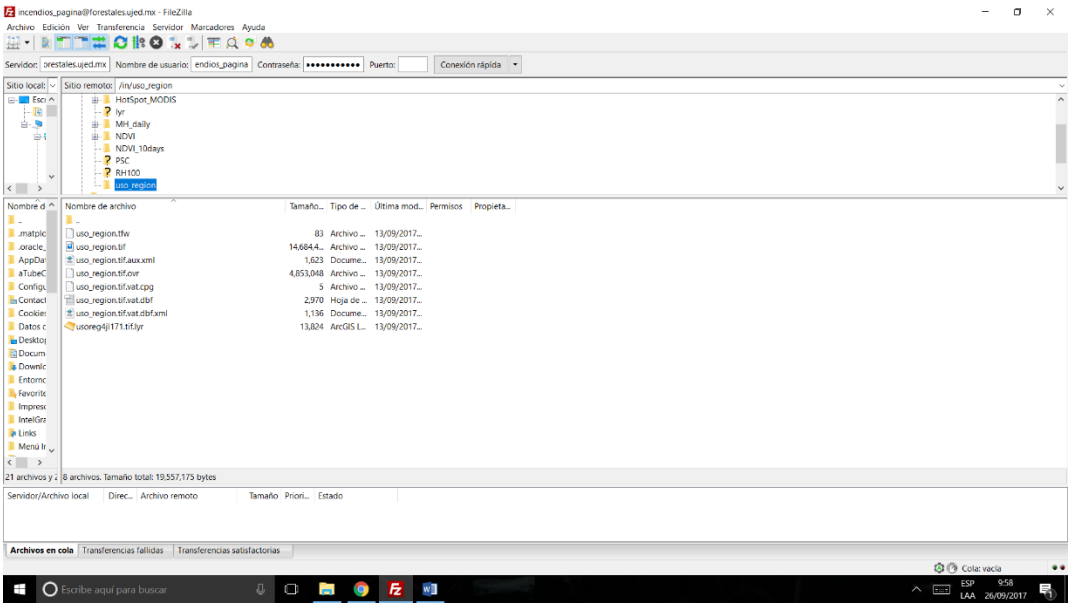
Con nombre **compositeNDVI*añomessemana*_merge.tiff**



**<u>Capas de entrada fijas:</u>**

4. **<u>Uso_region:</u>**

**Ruta:** "TIFFusoRegion": rutaBase + "/in/uso_region/uso_region.tiff"
**Contenido:** Archivo .tiff con la capa combinada de uso y región del país.



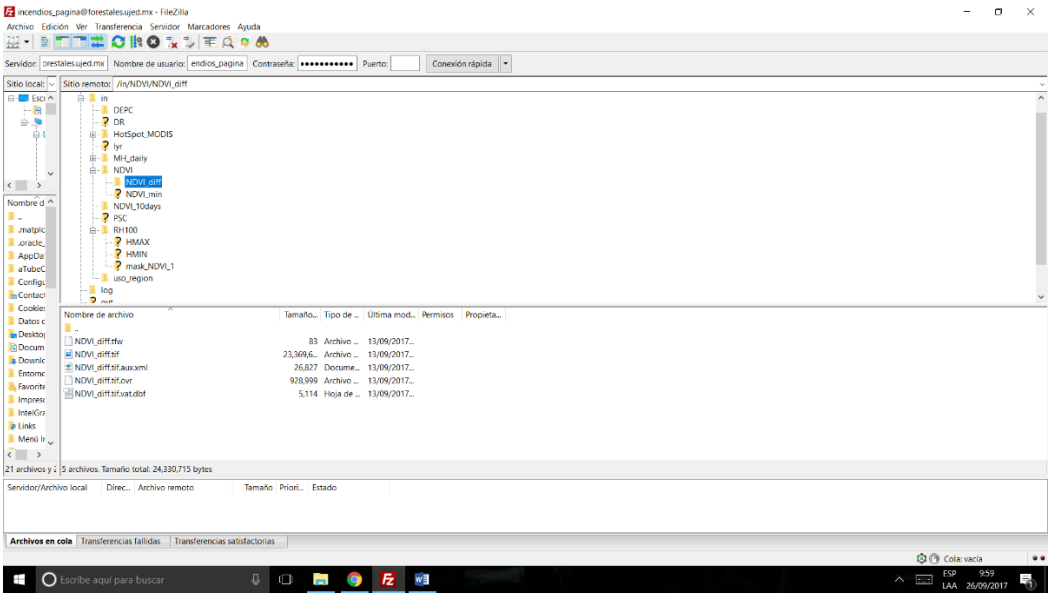
5. **NDVI:**

**Ruta:**

- "NDVImin": rutaBase + "/in/NDVI/NDVI_min/NDVI_min.tif",
- "NDVIdiff": rutaBase + "/in/NDVI/NDVI_diff/NDVI_diff.tif",

**Contenido:**

- "NDVImin": Archivo .tiff del mínimo valor de NDVI para cada pixel de 1 km calculado a partir de los históricos de NDVI de CONABIO.
- "NDVIdiff': Archivo .tiff de la diferencia entre el valor máximo y mínimo de NDVI para cada pixel de 1 km calculado a partir de los históricos de NDVI de CONABIO.



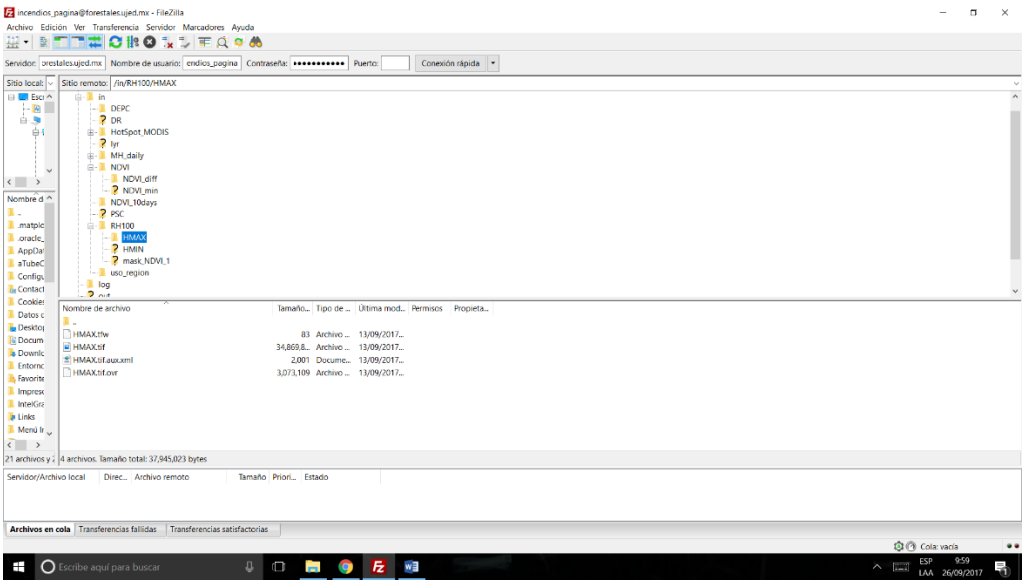

6. **RH100**

**Ruta:**

- "H100_HMIN": rutaBase + "/in/RH100/HMIN/HMIN.tif",
- "H100_HMAX": rutaBase + "/in/RH100/HMAX/HMAX.tif",
- "H100_mask_NDVI_1": rutaBase + "/in/RH100/mask_NDVI_1/mask_NDVI_1.tif",

**Contenido:**

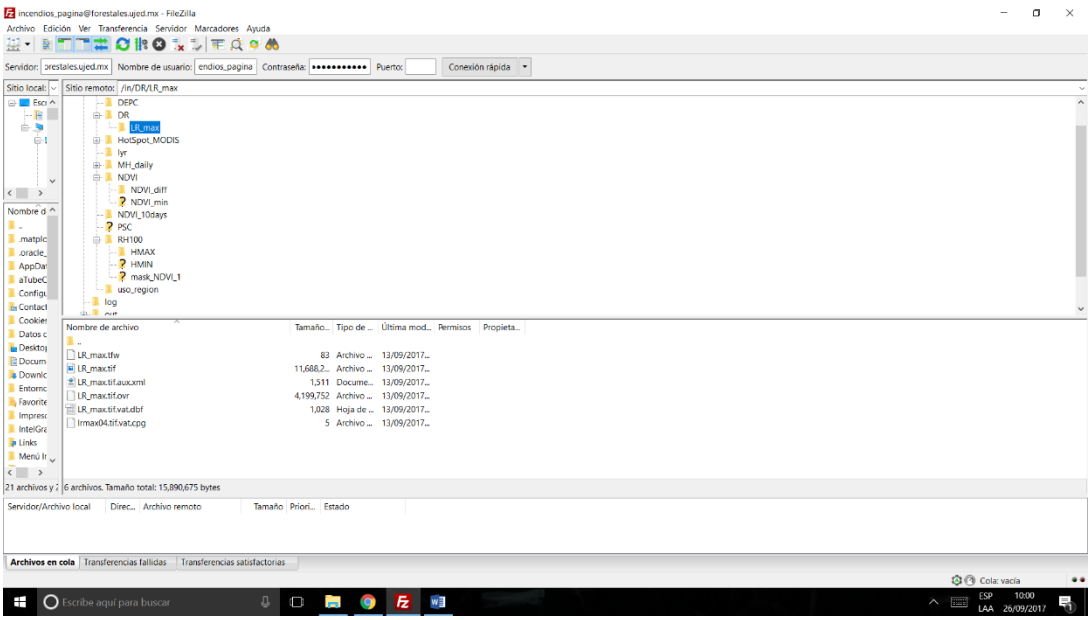
- H100_HMIN: Archivo .tiff con el valor mínimo de H100, calculado a partir de los históricos de CONABIO.
- H100_HMAX: Archivo .tiff con el valor máximode H100, calculado a partir de los históricos de CONABIO.
- H100_mask_NDVI_1: Archivo .tiff con máscara de 1 km del extent de las imágenes de NDVI para los archivos de H100, usado para calcular intersección de los extent de ambos archivos.



7. **DR:**

**Ruta:** "DR_LR_max": rutaBase + "/in/DR/LR_max/LR_max.tif",

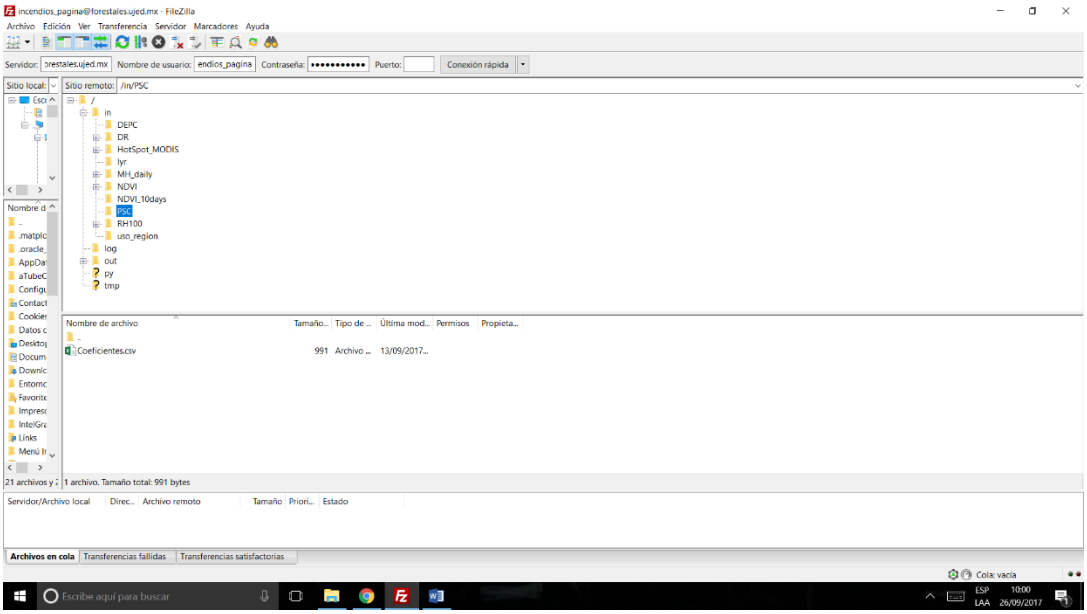
**Contenido:** LRMAX: Archivo en formato .tiff del valor máximo de Live Ratio (Burgan, 1998, calibrado para México por Vega et al., 2017), calculado a partir de los históricos de NDVI de CONABIO.



8. **PSC:**

**Ruta:** "archivoPSCcoeficientes": rutaBase + "/in/PSC/Coeficientes.csv"

**Contenido:** Archivo .csv con los coeficientes para cada uso_region para el cáculo de PSC.



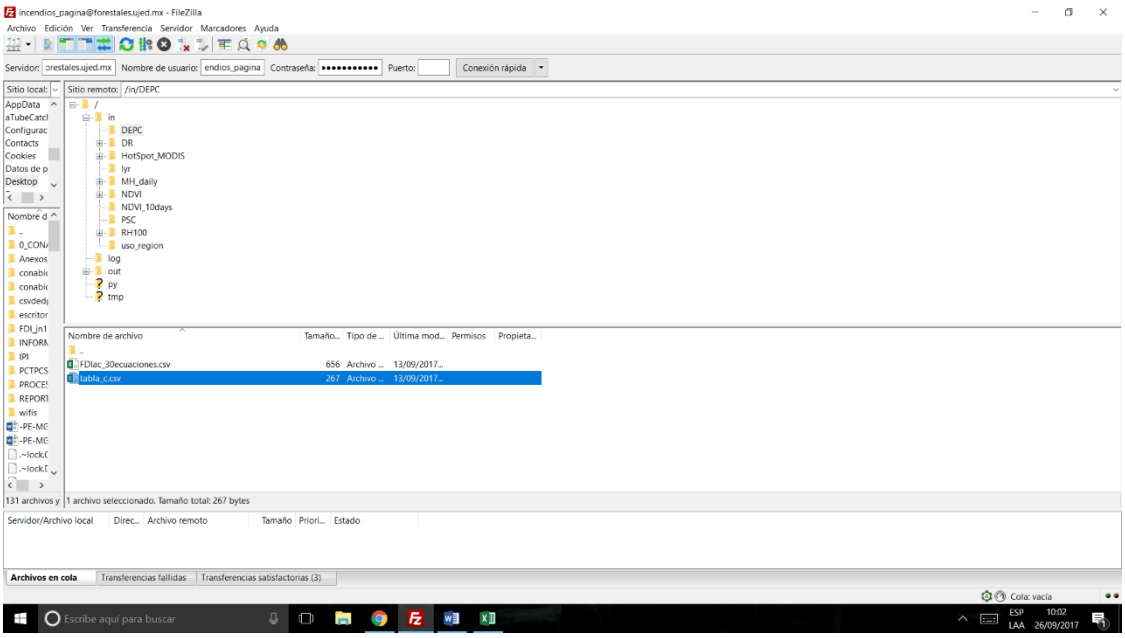

9. **DEPC:**

**Ruta:**

- "DEPC_archivo_parametros": rutaBase + "/in/DEPC/FDIac_30ecuaciones.csv",
- "DEPC_contante_c": rutaBase + "/in/DEPC/tabla_c.csv",

**Contenido:**

- **“**FDIac_30ecuaciones”: tabla .csv con coeficientes a y b para ecuaciones de cálculo de DEPC_FDI
- “tabla_c”: tabla .csv con coeficientes c para ecuaciones de cálculo de DEPC.



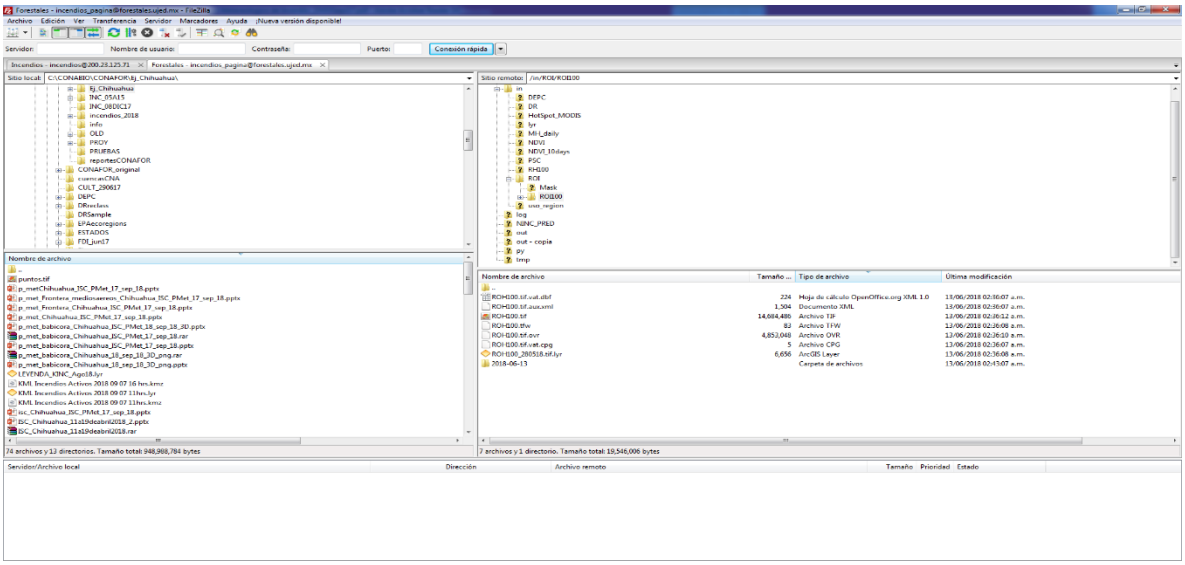
## 10. ROI

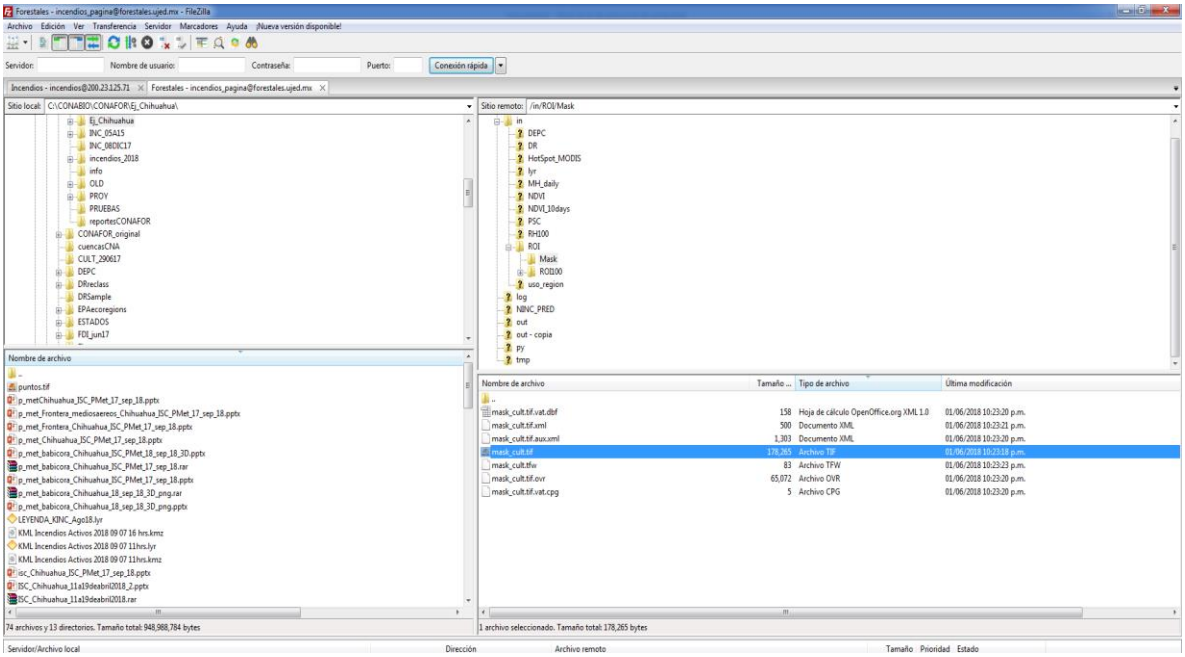
**Ruta:**

- rutaBase + "/in/ROI/ROI100/ROH100.tif
- rutaBase + "/in/ROI/ROI100/mask_cult.tif

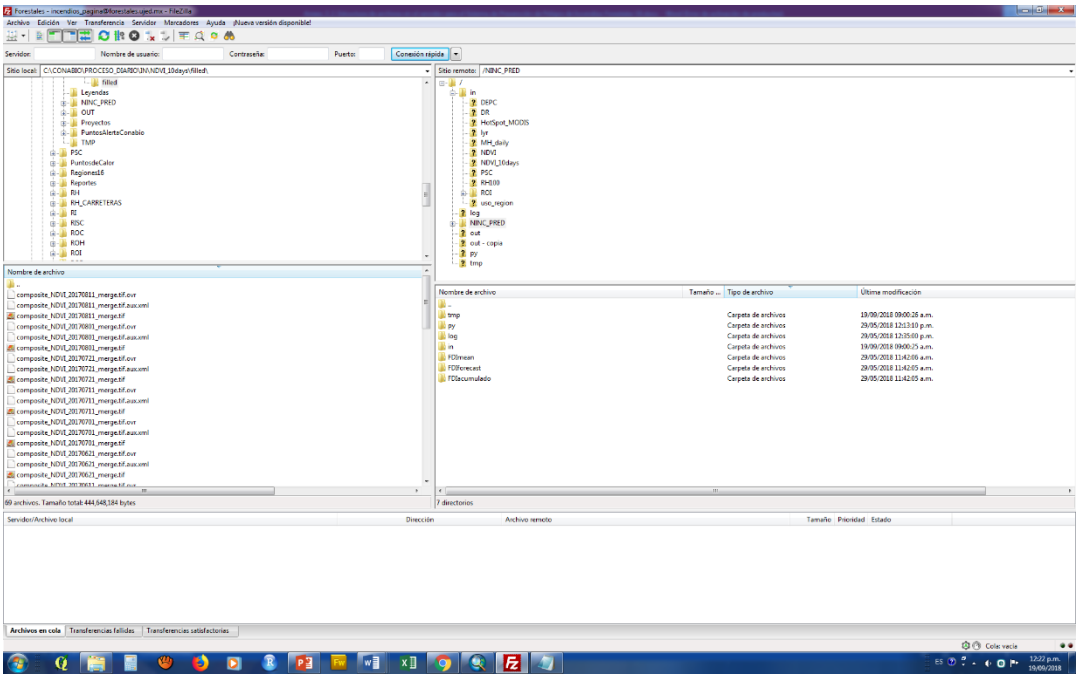
**Contenido:**

- ROH100.tif: Archivo .tif del Mapa de riesgo de Incendio por factores humanos.
- mask_cult.tif: Archivo .tif de máscara de cultivos.





## NINC_PRED



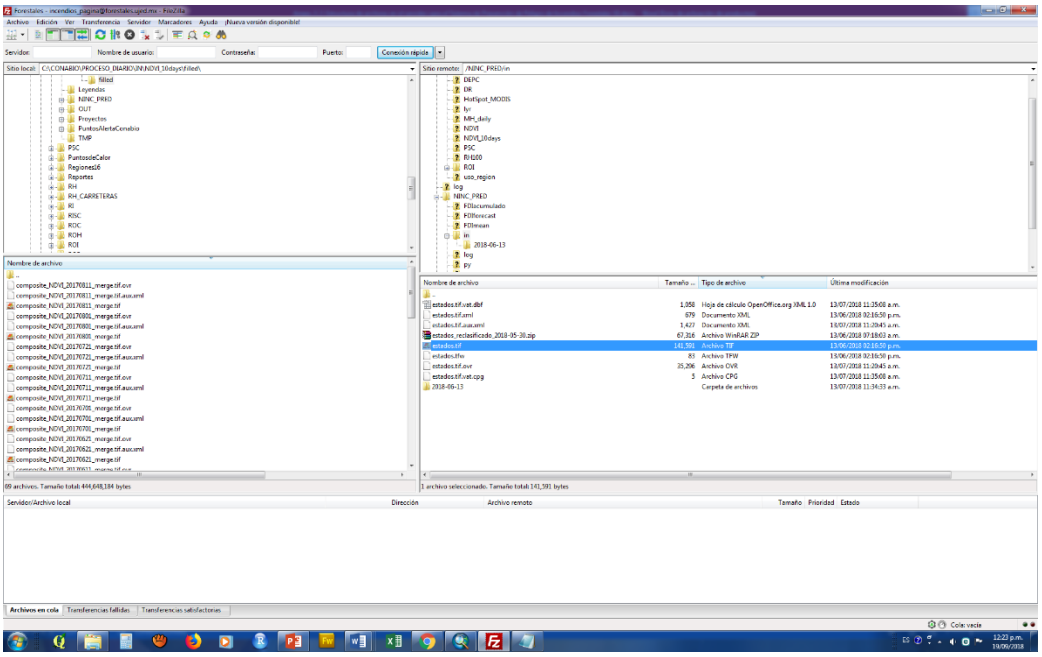
**IN :** Archivos de entrada;

**FDImean, FDIforecast, FDIacumulado**: salidas del sistema;

**PY:** archivos *phyton* con las operaciones del Sistema.

**TMP:** archivos temporales del Sistema.

## IN (ENTRADAS):



**Ruta:**

rutaBase + "/NINC_PRED/in/ estados.tif

**Contenido:**

estados.tif: archivo .tif de los estados de la republica mexicana.

**FDIforecast:**

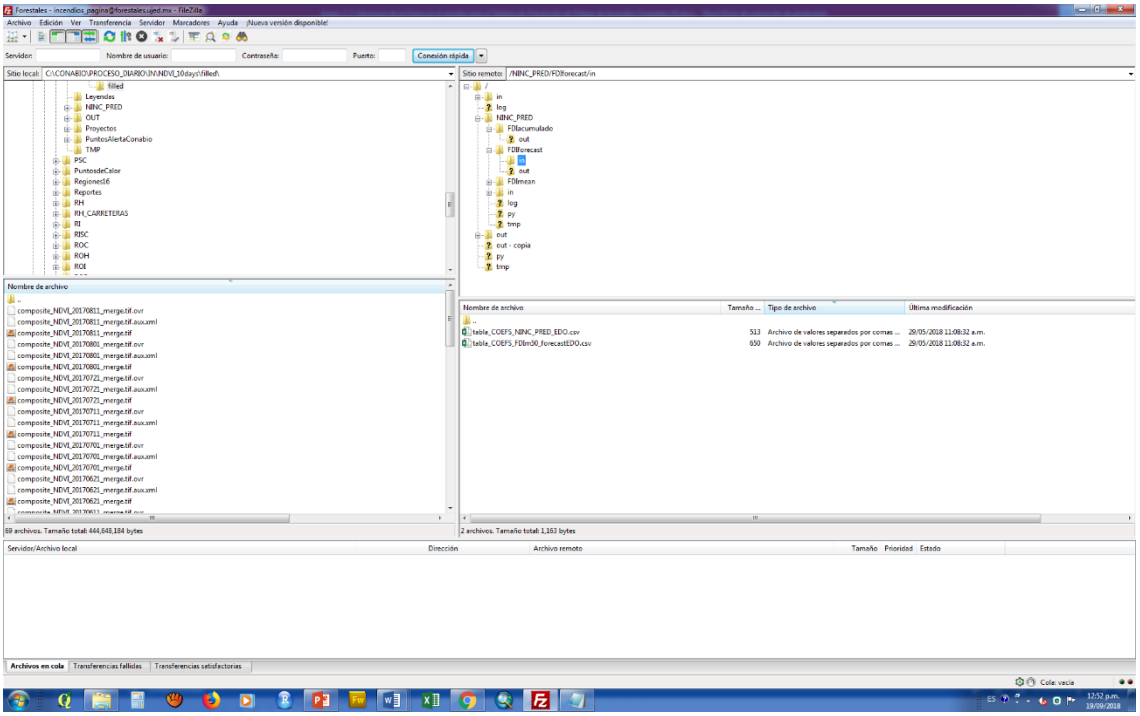
**Ruta:**

- rutaBase + "/NINC_PRED/FDIforecast/tabla_COEFS_NINC_PRED_EDO.csv
- rutaBase + "/NINC_PRED/FDIforecast/ tabla_COEFS_FDIm50_forecastEDO.csv

**Contenido:**

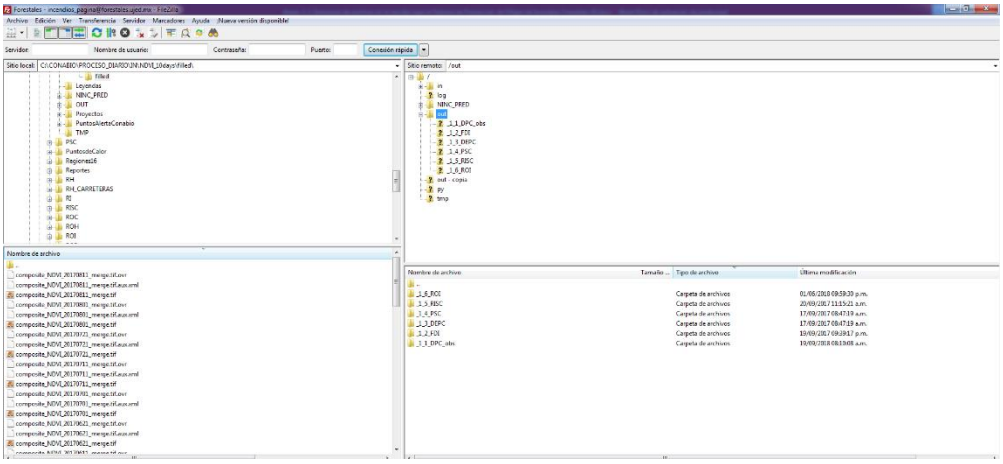
COEFS_NINC_PRED_EDO.csv: archivo .csv con los coeficientes para el cálculo de NINC por estado.

tabla_COEFS_FDIm50_forecastEDO.csv: archivo .csv con los coeficientes para el cálculo de FDIforecast



## SALIDAS (OUT):

- **1_1_DPC_obs** (DPCobs*AñoMesDia*.csv)
- **1_2_FDI**
- **1_3_DEPC**
- **1_4_PSC**
- **1_5_RISC**
- **1_6_ROI**
- **NINC_PRED**

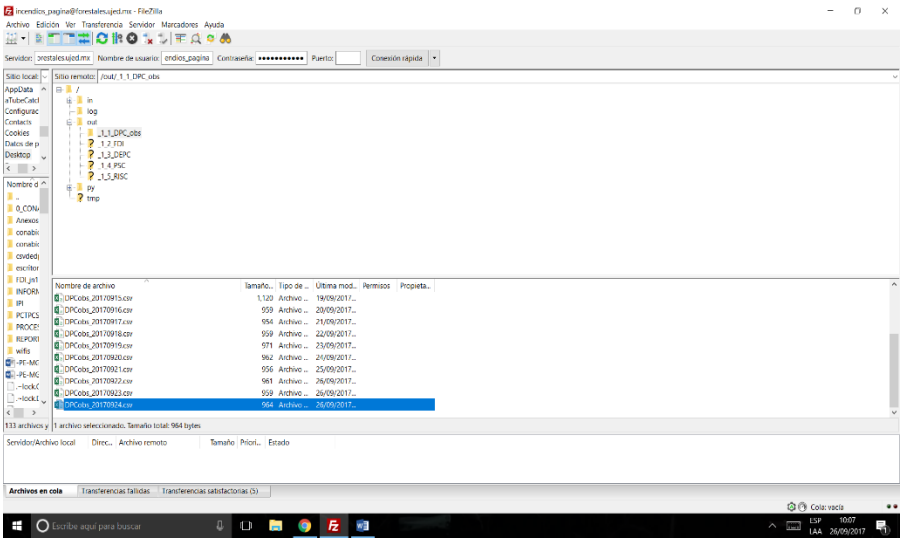


## 1_1_DPC_obs

**Ruta:** "rutaSalidaDPC": rutaBase + "/out/_1_1_DPC_obs",

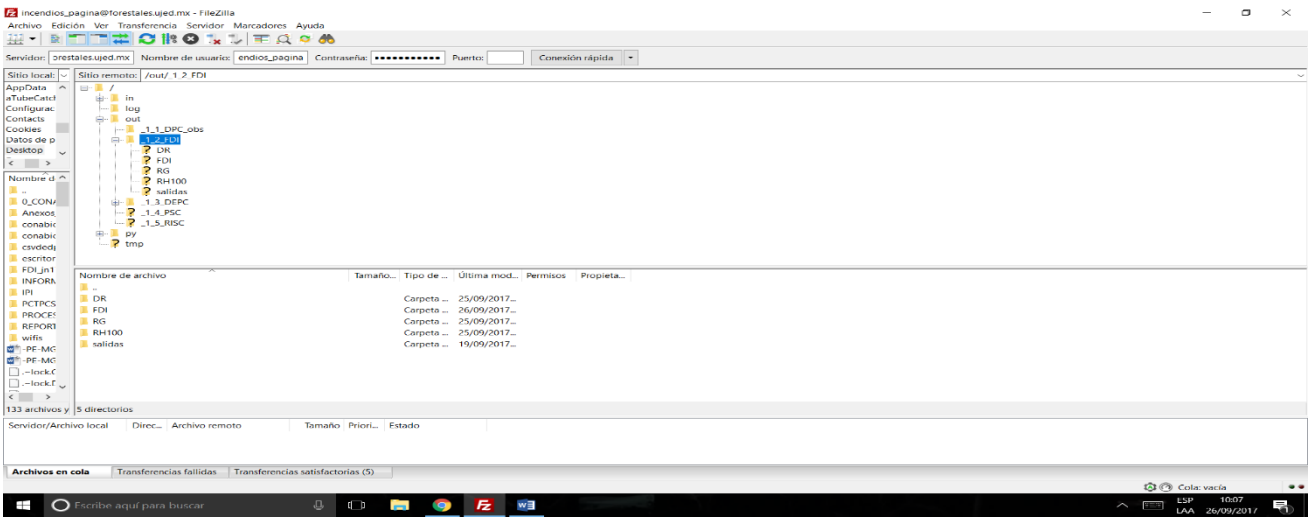
**Contenido:** archivo .csv de Densidad de Puntos de Calor Observada (DPCobs)

Se calcula un archivo diario con nombre **DPCobs*AñoMesDia*.csv**



## 1_2_FDI:

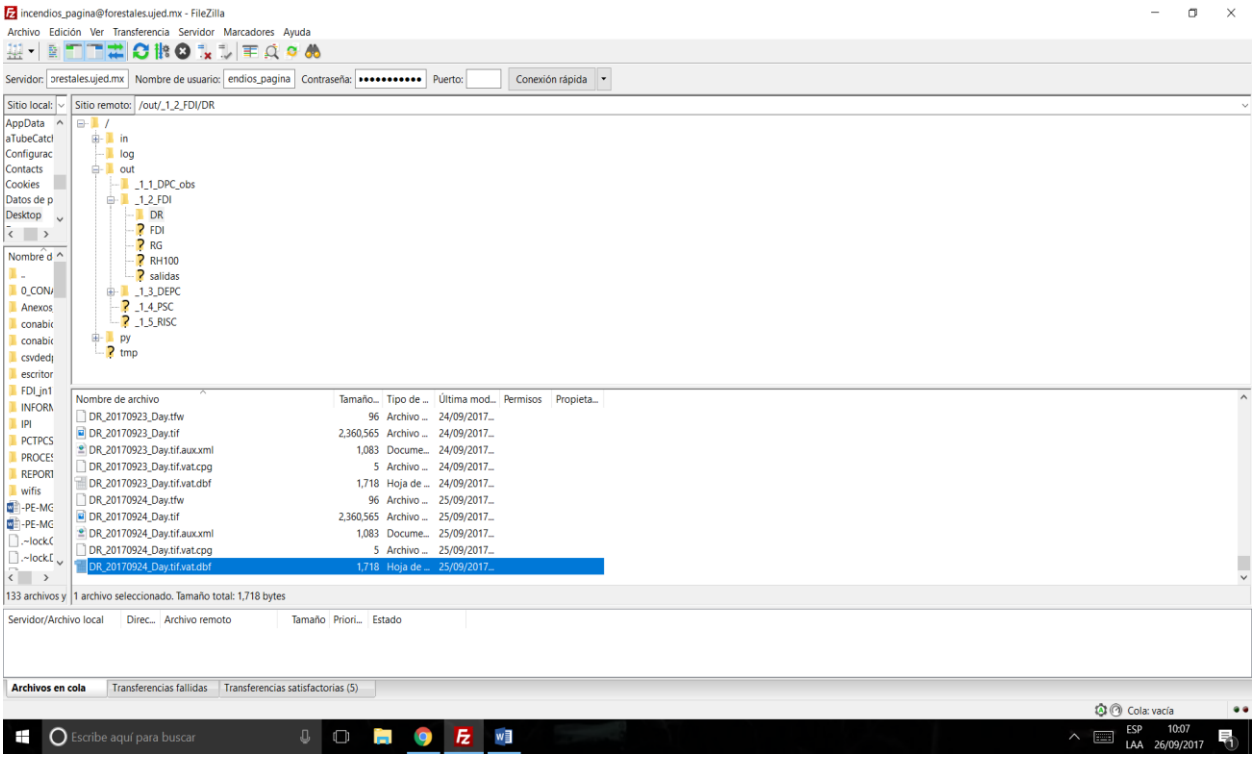
- **DR**
- **RG**
- **RH100**
- **FDI**




- **DR:**

**Ruta:** "rutaFDI_DR_DR": rutaBase + "/out/_1_2_FDI/DR",

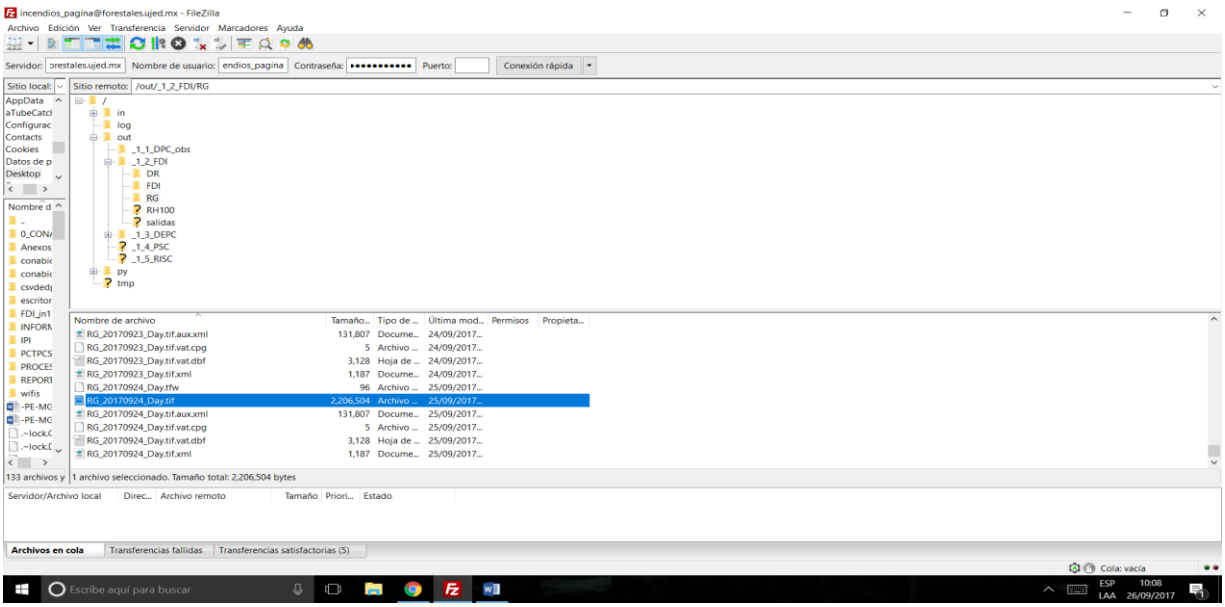
**Contenido:** Archivo .tiff de Dry Ratio (DR) con nombre **DR_*AñoMesDia*.tiff**



- **RG:**

**Ruta:** "rutaFDI_DR_RG": rutaBase + "/out/_1_2_FDI/RG",

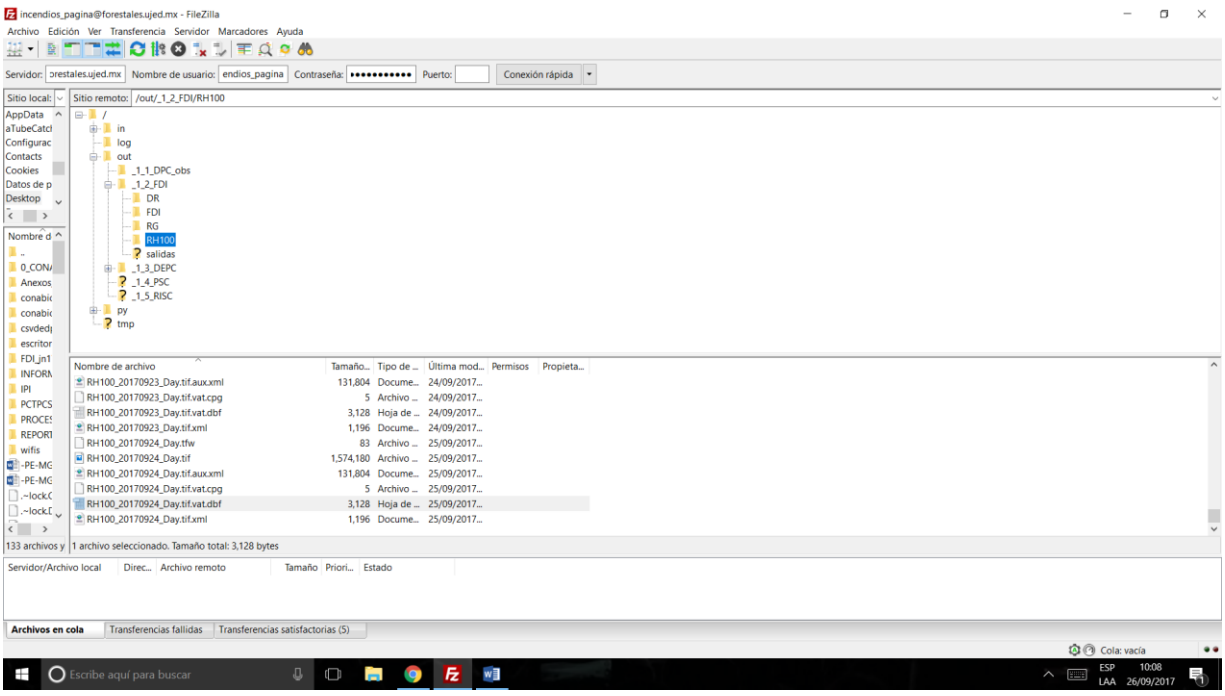
**Contenido:** Archivo .tiff de *Relative Greeness* con nombre **RG_*AñoMesDia*.tiff**




- **RH100:**

**Ruta:** "rutaFDI_RH100": rutaBase + "/out/_1_2_FDI/RH100",

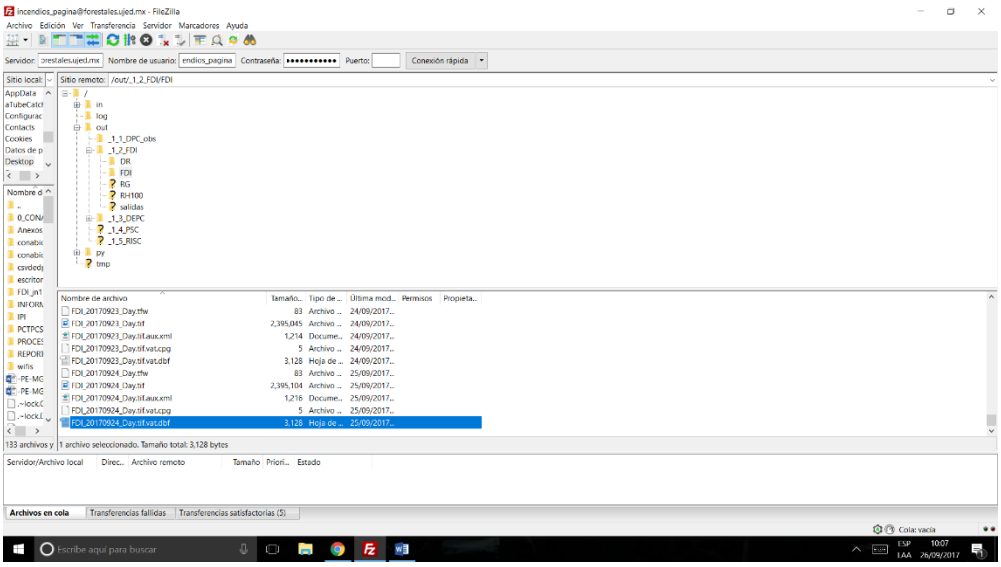
**Contenido:** Archivo .tiff de RH100 *(H100 ratio)* con nombre **RH100_*AñoMesDia*.tiff**



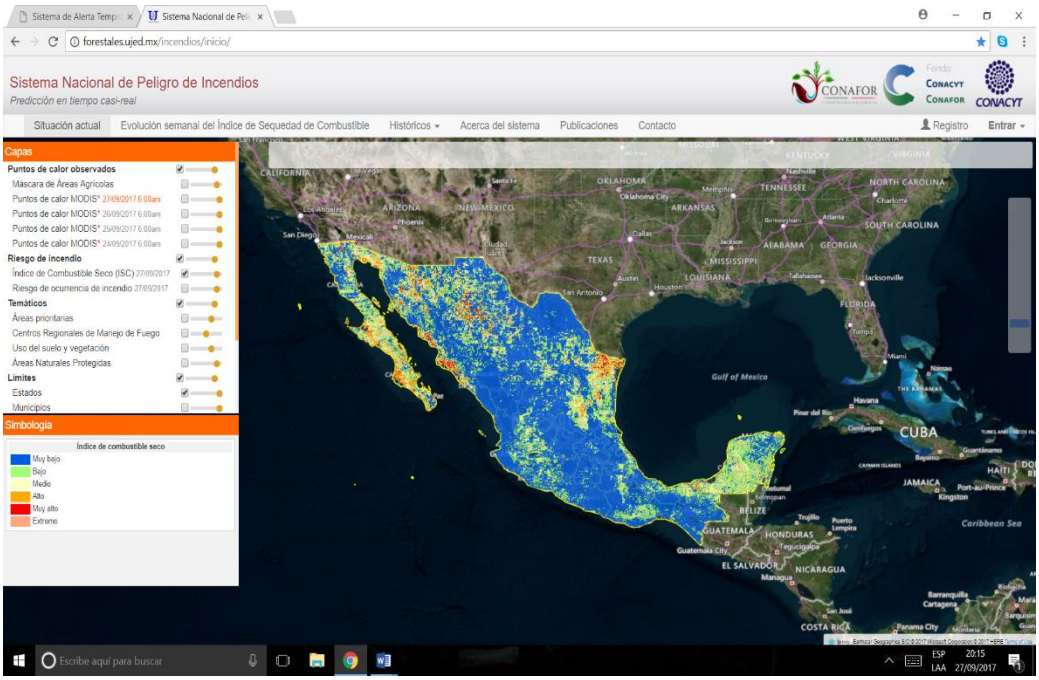
- **FDI:**

**Ruta:** "rutaFDI_FDI": rutaBase + "/out/_1_2_FDI/FDI",

**Contenido:** Archivo .tiff de FDI *(Fuel Dryness Index)* –Índice de Sequedad del Combustible (ISC)-con nombre **FDI_*AñoMesDia*.tiff**



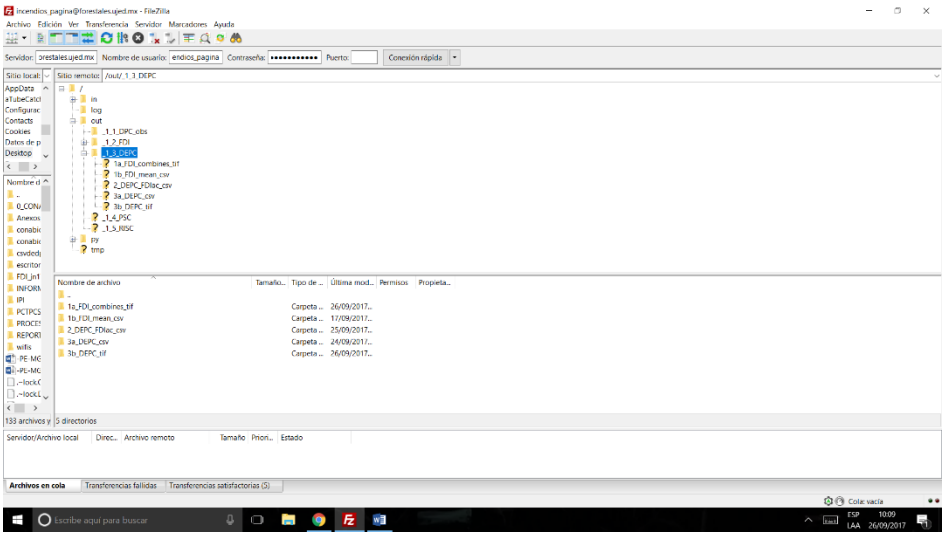
El archivo F **FDI_*AñoMesDia*.tiff** diario se sube diariamente al visor de la plataforma en la sección  Índice de Sequedad del Combustible (ISC) (dia/mes/año)




**Captura de la imagen de Índice de Sequedad del Combustible (ISC) de situación actual en el Sistema de Peligro de Incendios.**

## 1_3_DEPC:

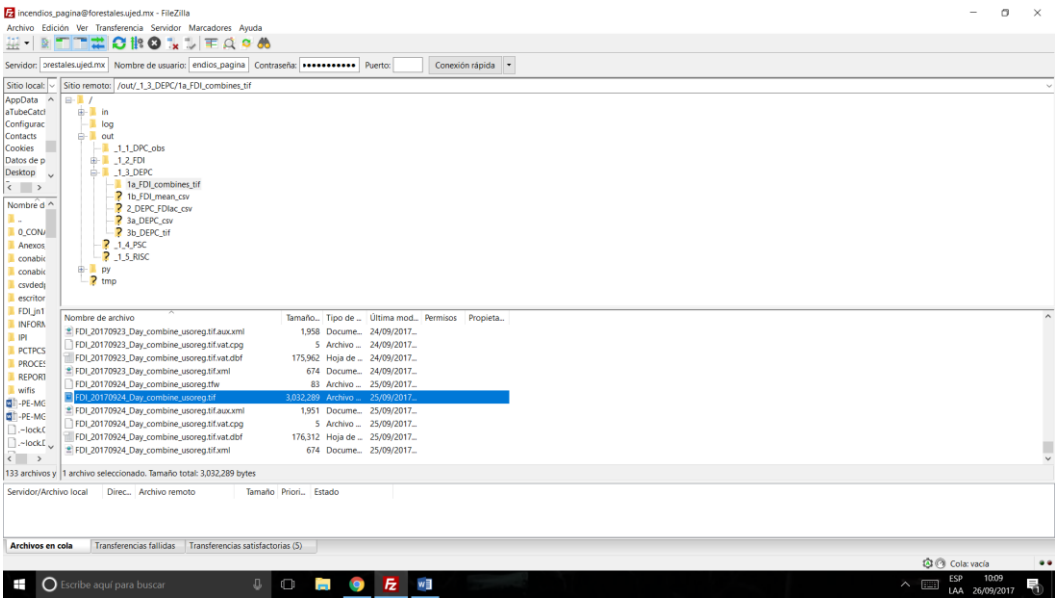
- 1a_FDI_combine
- 1b_FDImean
- 2_DEPC_FDIac
- 3a_DEPC_pred
- 3b_DEPC_tiff



- **1a_FDI_combine:**

**Ruta:** "DEPC_FDI_combines_tif": rutaBase + "/out/_1_3_DEPC/1a_FDI_combines_tif",

**Contenido:** Archivo .tiff de combine de FDI con uso región con nombre **FDI_*AñoMesDia*_Day.tiff**

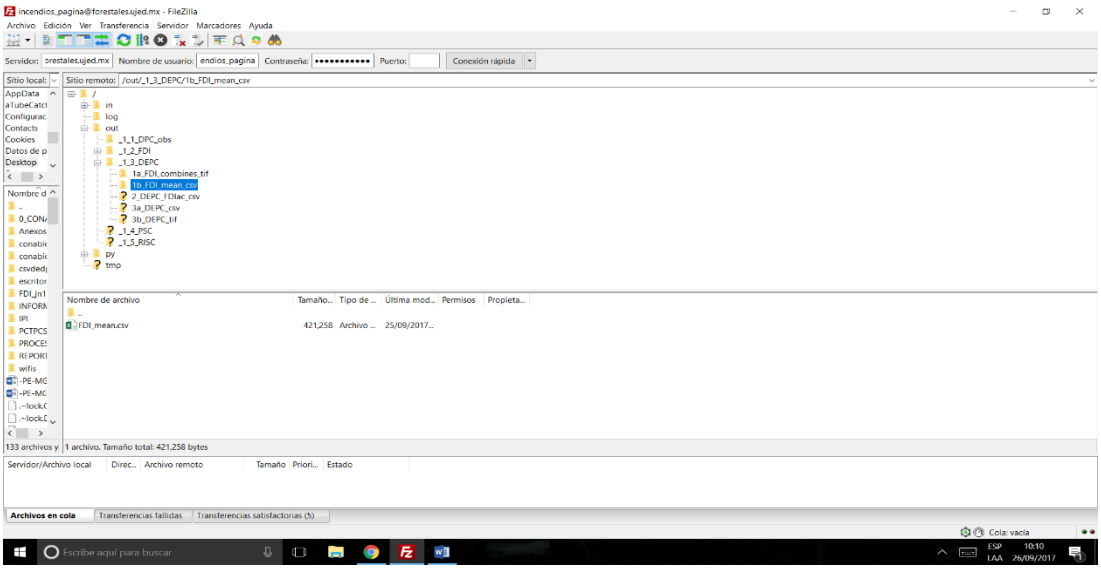


- **1b_FDImean:**

**Ruta:** "archivoPSCEntradaFDImean":

rutaBase +/out/_1_3_DEPC/1b_FDI_mean_csv/FDI_mean.csv",

**Contenido:** Archivo .csv con los promedios diarios por uso región de FDI (FDImean).

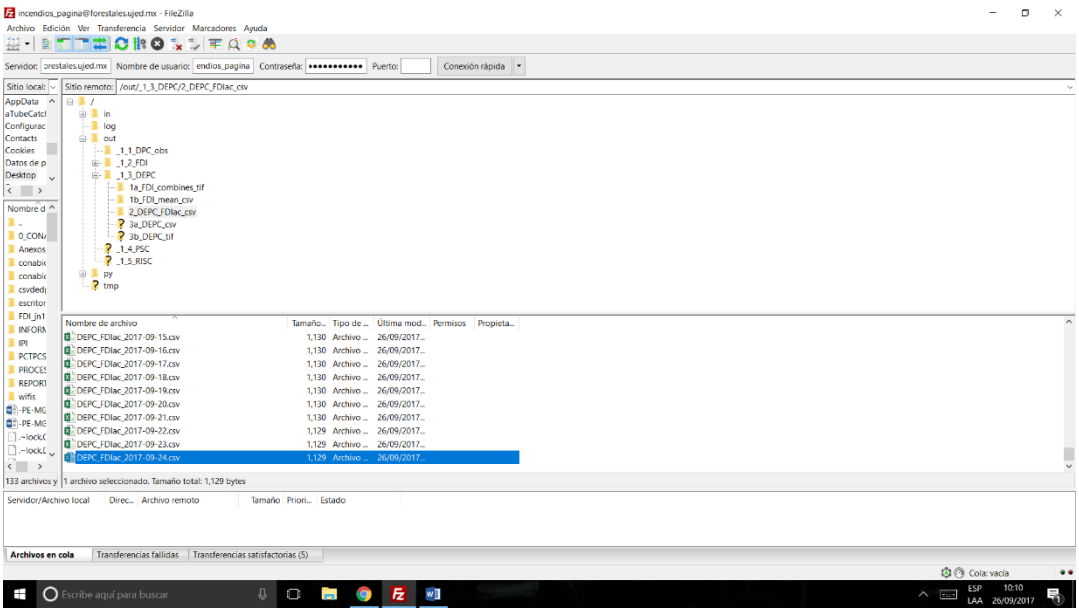


- **2_DEPC_FDIac:**

**Ruta:** "DEPC_FDIac_csv":

rutaBase + "/out/_1_3_DEPC/2_DEPC_FDIac_csv/DEPC_FDIac_Fecha.csv",

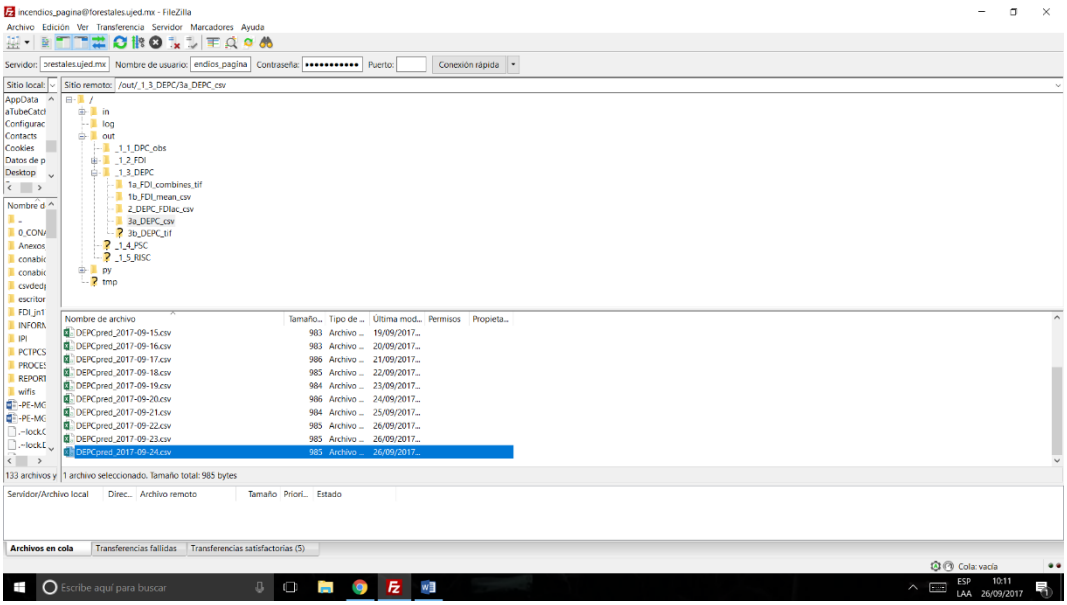
**Contenido:** Archivo .csv con el valor de FDI acumulado (FDIac) diario para cada uso región con nombre **DEPC_FDIac_*Año-Mes-Dia*.csv**



- **3a_DEPC_csv:**

**Ruta:** "rutaDEPCsalida_csv": rutaBase + "/out/_1_3_DEPC/3a_DEPC_csv",

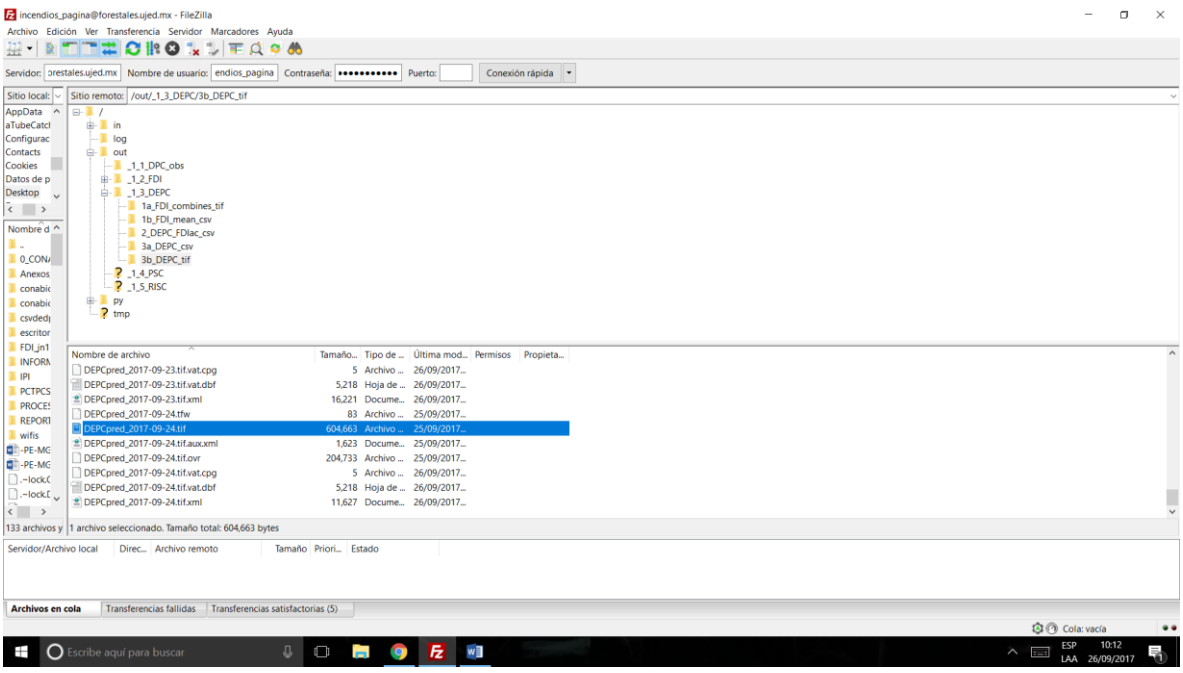
**Contenido:** Archivo .csv diario de Densidad Esperada de Puntos de Calor (DEPC), con nombre **DEPCpred_*Año-Mes-Dia*.csv**



- **3b_DEPC_tiff**

**Ruta:** "rutaDEPCsalida_tif": rutaBase + "/out/_1_3_DEPC/3b_DEPC_tif",

**Contenido:** Archivo .tiff diario de Densidad Esperada de Puntos de Calor (DEPC), con nombre **DEPCpred_*Año-Mes-Dia*.tiff**



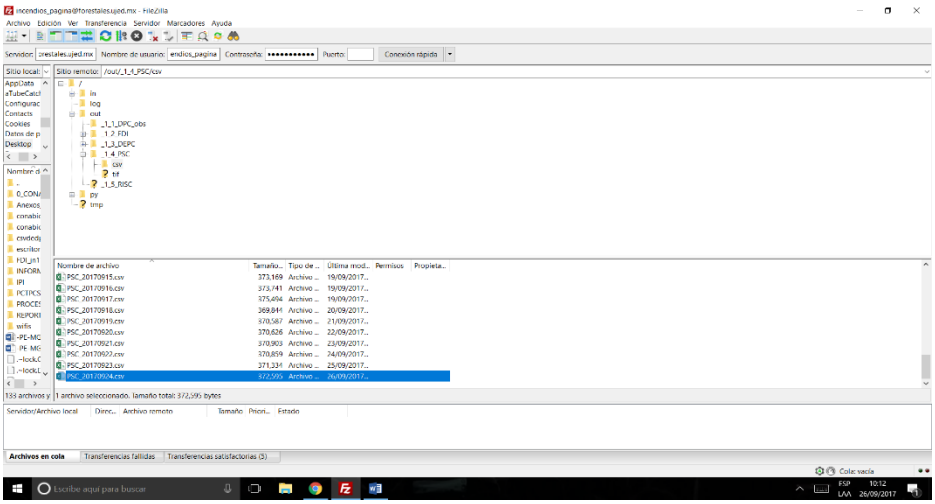
## 1_4_PSC:

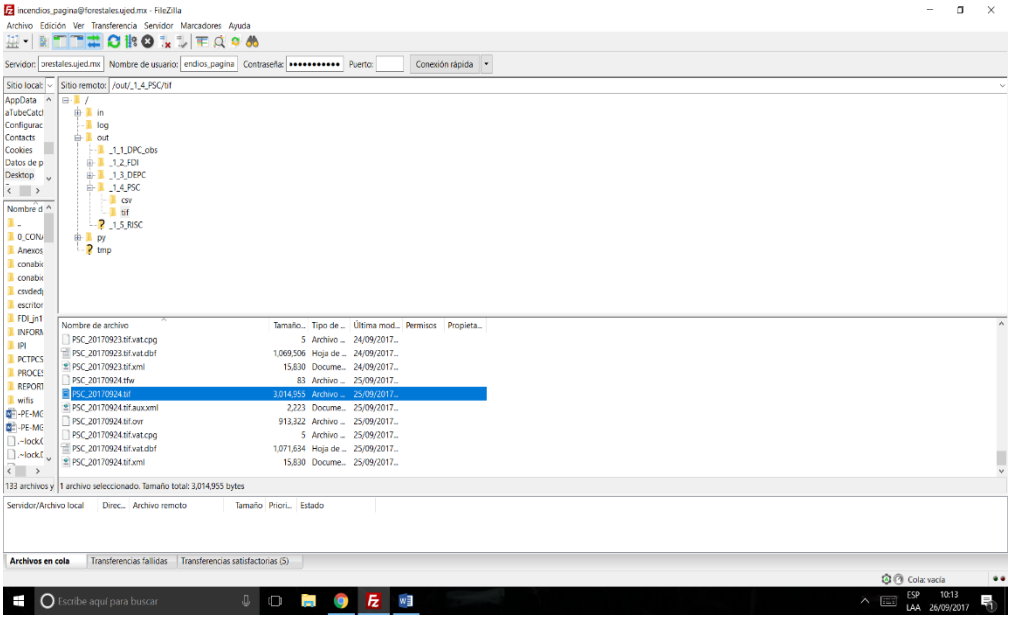
**Ruta:**

- "rutaPSCsalida_csv": rutaBase + "/out/_1_4_PSC/csv",
- "rutaPSCsalida_tif": rutaBase + "/out/_1_4_PSC/tif",

**Contenido:**

- **PSC.csv:** Archivo .csv diario de Porcentaje de Incendios por categorías de Sequedad de Combustible (PSC), con nombre **PSC_*Año-Mes-Dia*.csv**
- **PSC.tiff:** Archivo .tiff diario de Porcentaje de Incendios por categorías de Sequedad de Combustible (PSC), con nombre **PSC_*Año-Mes-Dia*.tiff**





## 1_5_RISC:

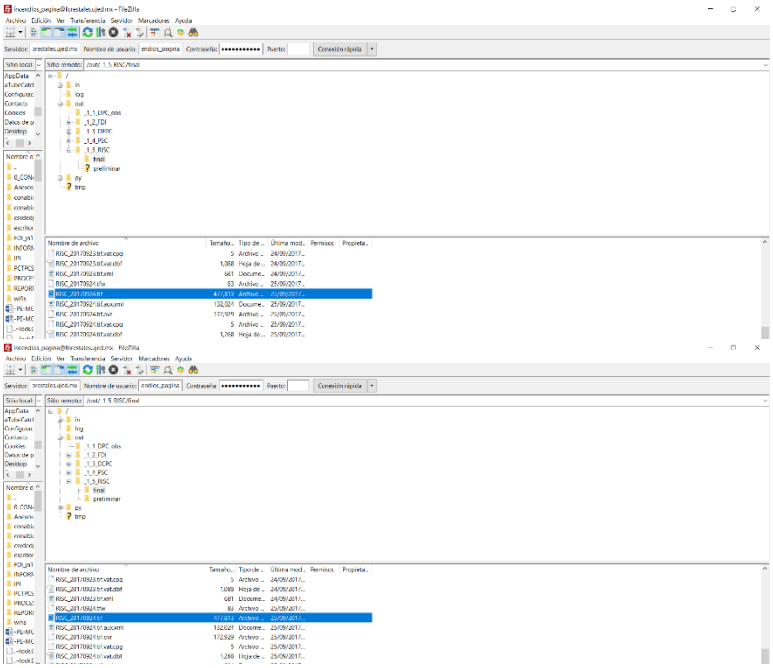
**Ruta:**

- "rutaRISCsalidaPreliminar": rutaBase + "/out/_1_5_RISC/preliminar",
- "rutaRISCsalidaFinal": rutaBase + "/out/_1_5_RISC/final",

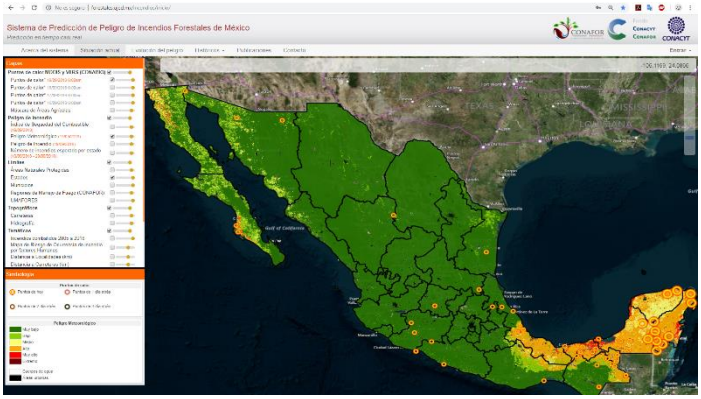
**Contenido:**

RISC preliminar: archivo .tiff de Riesgo de Ocurrencia de Incendio por Sequedad del Combustible.

**RISC final:** archivo .tiff final reclasificado de Riesgo de Ocurrencia de Incendio por Sequedad del Combustible, con nombre **RISC_*Año-Mes-Dia*.tiff**



El archivo **RISC_*Año-Mes-Dia*.tiff** se sube diariamente al visor de la plataforma en la sección Peligro de Incendio: **Peligro meteorológico** (dia/mes/año).




**Captura de la imagen Peligro meteorológico en el Sistema de Peligro de Incendios.**

## 1_6_ROI:

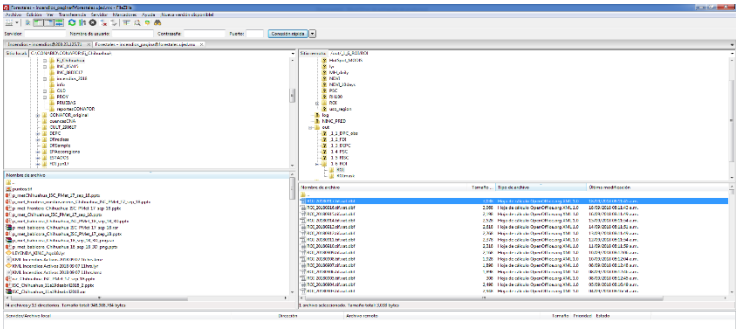
**Ruta:**

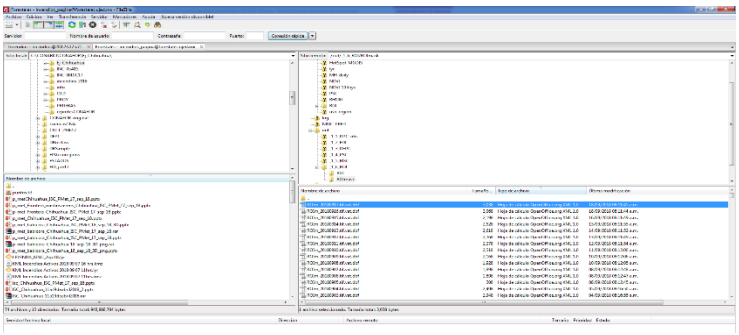
- rutaBase + "//out/_1_6_ROI/ROI
- rutaBase + "//out/_1_6_ROI/ROImask

**Contenido:**

ROI: archivo de peligro meteorológico sin filtrar áreas agrícolas.

ROIm: archivo final de peligro meteorológico

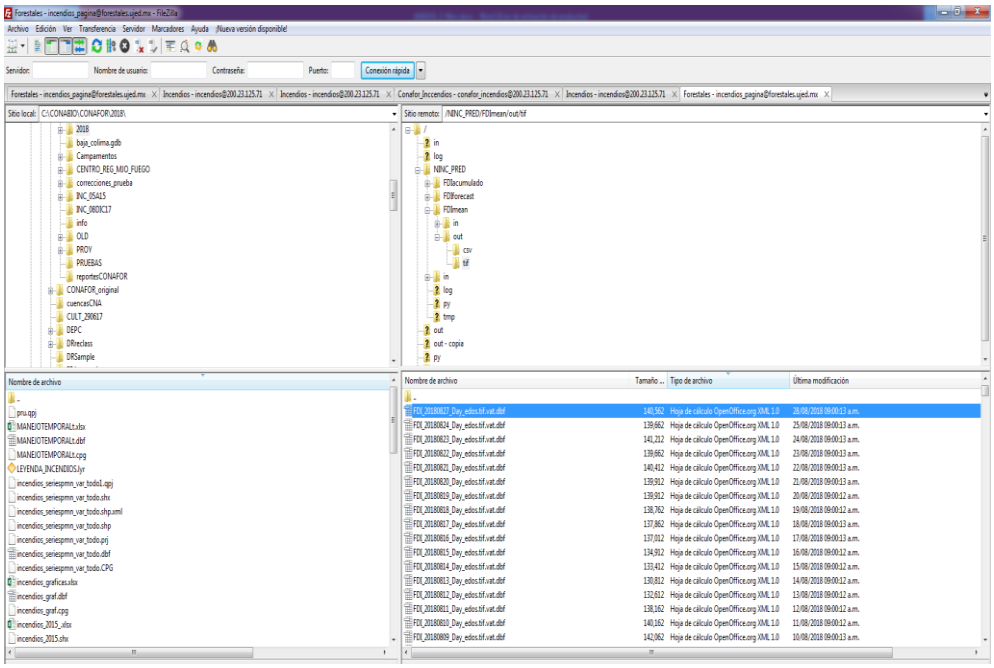




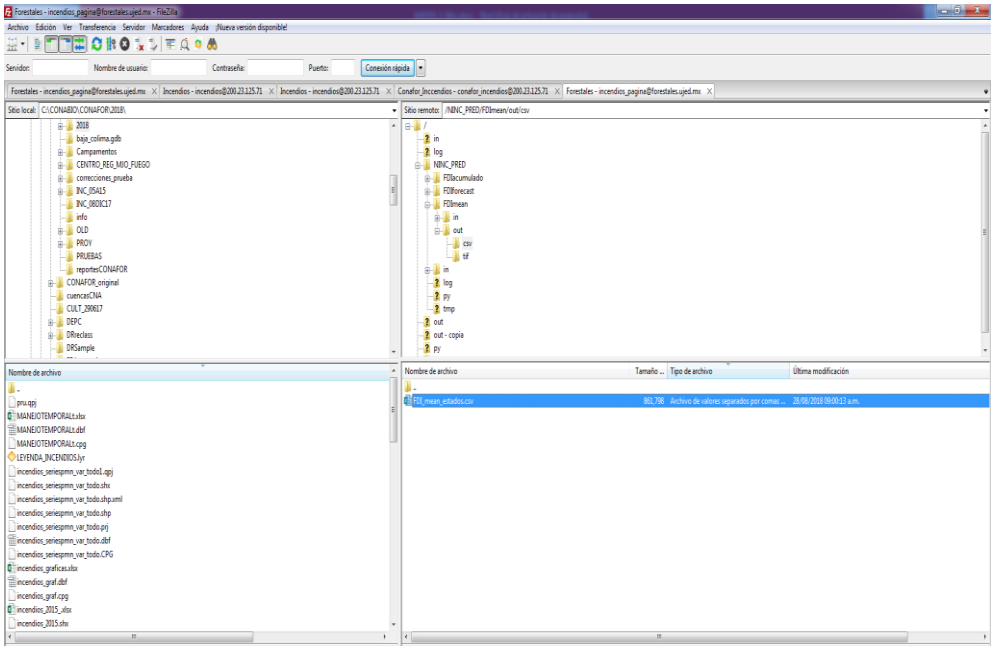
El archivo **ROI_*Año-Mes-Dia*.tiff** se sube diariamente al visor de la plataforma en la sección Peligro de Incendio: **Peligro de incendio** (dia/mes/año).


**Captura de la imagen Peligro de incendio en el Sistema de Peligro de Incendios.**

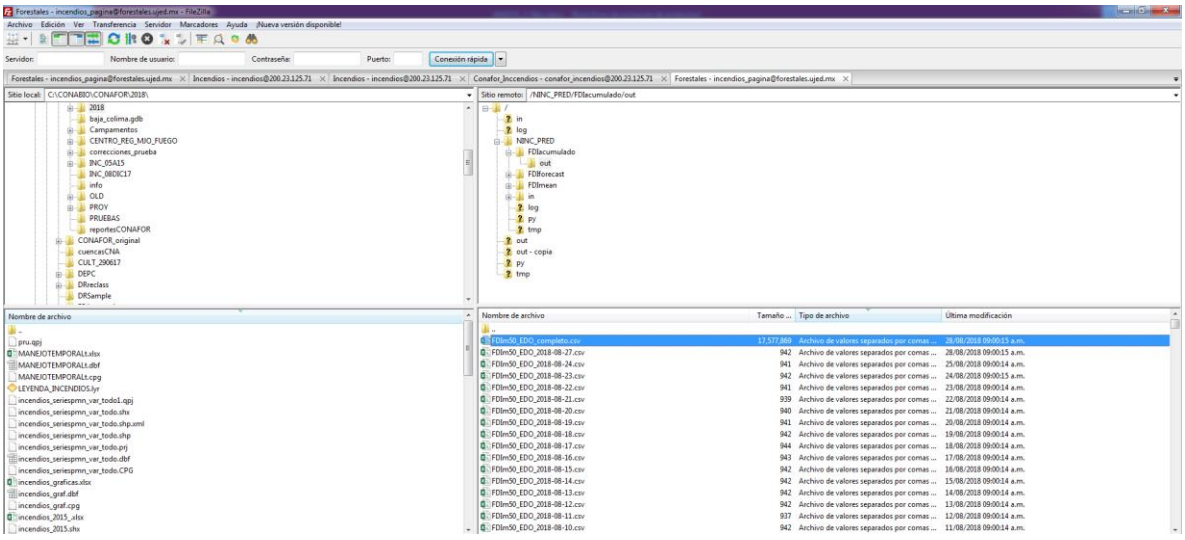
- **NINC_PRED (SALIDAS):**



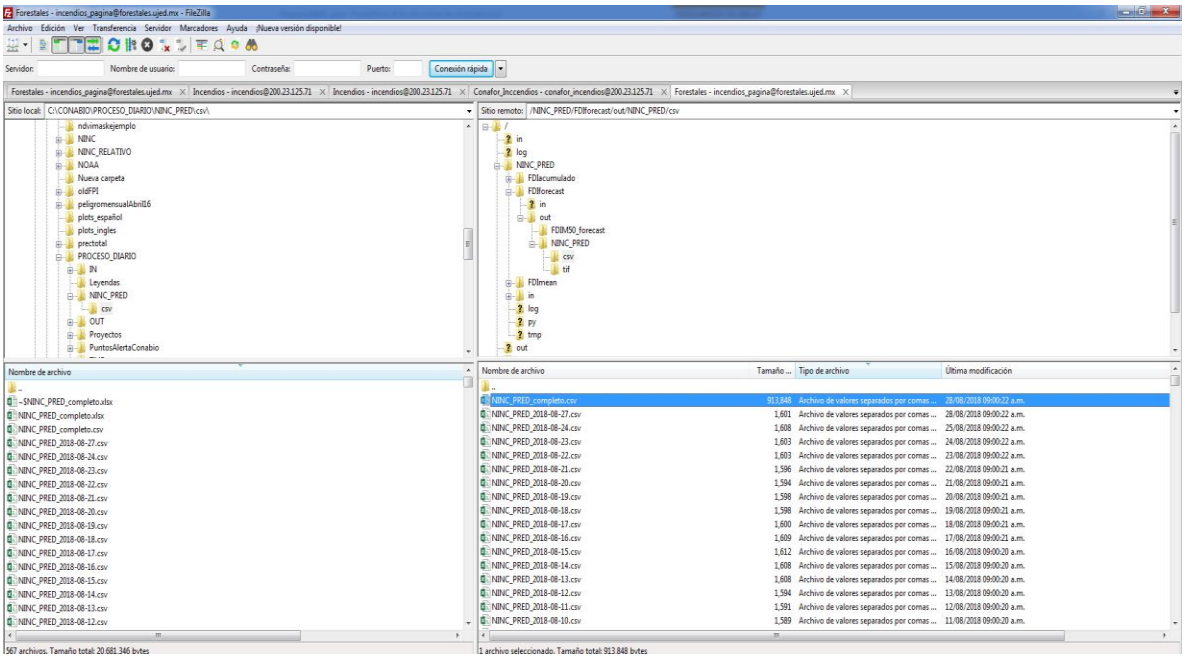
**Salida del Subproceso 1: Archivo ráster de combinación de los archivos tiff de FDI diario y estados (FDI_Day_Edos.tif)**



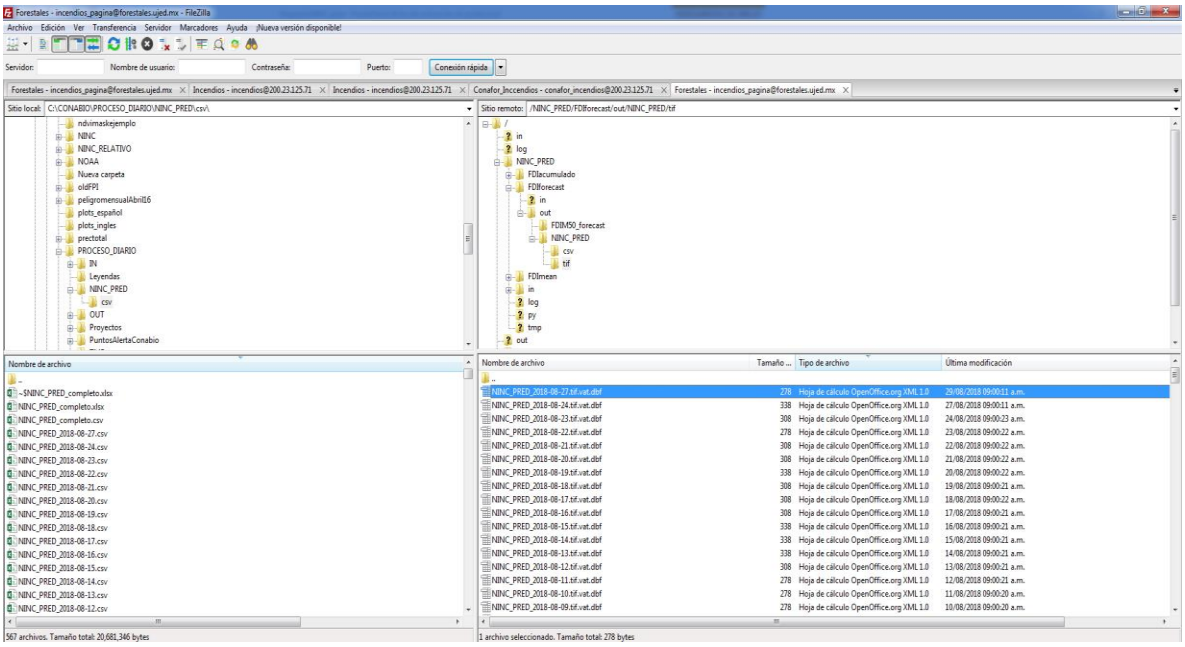
**Salida del Subproceso 2: Archivo .csv con el FDI promedio por estado y día (FDIMean_estados.csv)**



**Salida del Subproceso 3: Archivo .csv con el promedio de los 50 días antecedentes de FDI para cada estado y día (FDIm50_EDO.csv)**



**Salida del Subproceso 4: Archivo .csv con el Número de incendios esperado por estado NINC_PRED.csv**



**Salida del Subproceso 5: Archivo .tif con el Número de incendios esperado por estado NINC_PRED_Day.tif**.

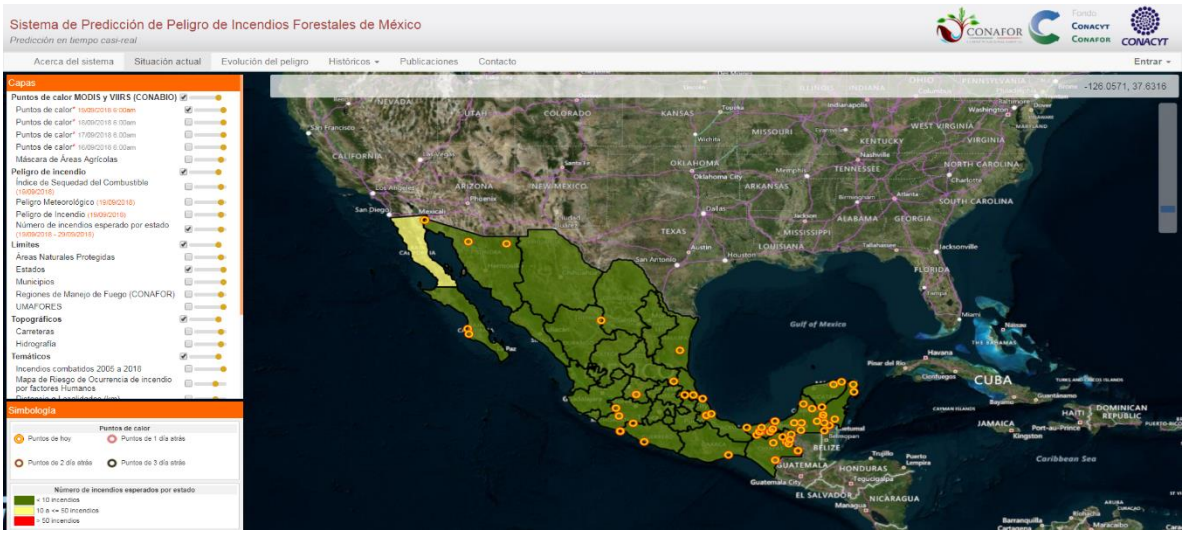
**Ruta:**

/NINC_PRED/FDIforecast/out/NINC_PRED/tif/ NINC_PRED_Day.tif.

**Contenido:**

NINC_PRED_Day.tif: archivo .tiff con el número de icnendios esperados por estado.

El archivo **NINC_PRED_*Año-Mes-Dia*.tiff** se sube diariamente al visor de la plataforma en la sección Peligro de Incendio: **Número de incendios esperado por estado** (dia/mes/año).




**Captura de la imagen Peligro de incendio en el Sistema de Peligro de Incendios.**

## 3. PROCESO DE CÁLCULO.

El sistema de peligro consta de los siguientes módulos:

1) **Proceso 1: Módulo de Peligro Meteorológico.**
   Con las siguientes salidas:
   **1.1.** FDI: **índice de sequedad del combustible.**
   1.2. DEPC: densidad esperada de puntos de calor
   1.3. PSC: Porcentaje de incendios por sequedad del combustible.
   1.4. RISC: **Peligro meteorológico.**

2) **Proceso 2: Módulo de Peligro de Incendio.**
   Con las siguientes salidas:
   2.1. ROIm: **Peligro de incendio.**
   2.2. NINC_PRED: **Número esperado de incendios por estado.**

Las salidas subrayadas se suben diariamente a la sección de Peligro de Incendio del sistema.

El diagrama continuación ilustra la integración de los dos módulos en el sistema:

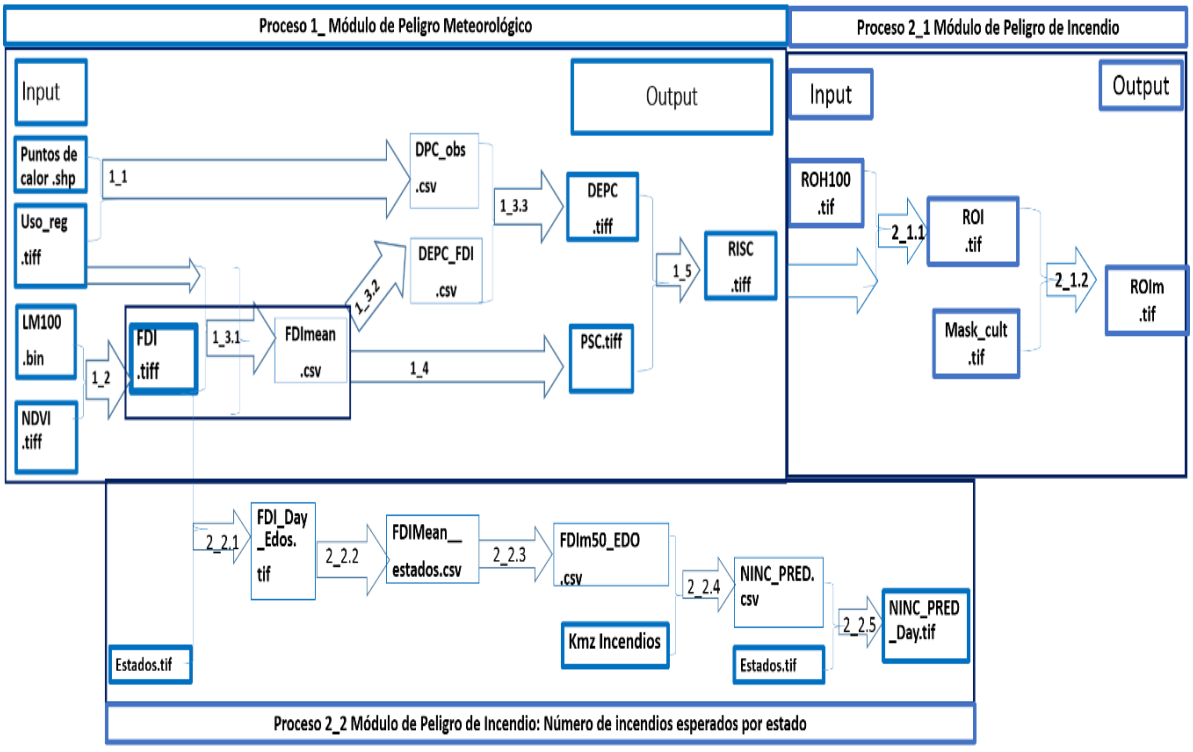



**Diagrama general de estructura del sistema de peligro.**

Se detallan a continuación los procesos de cálculo de cada módulo. Se puede encontrar más información en los anexos 1, 1.1, 1.2 y 2 del reporte de Vega et al. (2018).

## 1. PROCESO DE CÁLCULO DEL MÓDULO DE PELIGRO METEREOLÓGICO.

Se resume en el siguiente diagrama el proceso general de cálculo del módulo de peligro meteorológico de incendio.

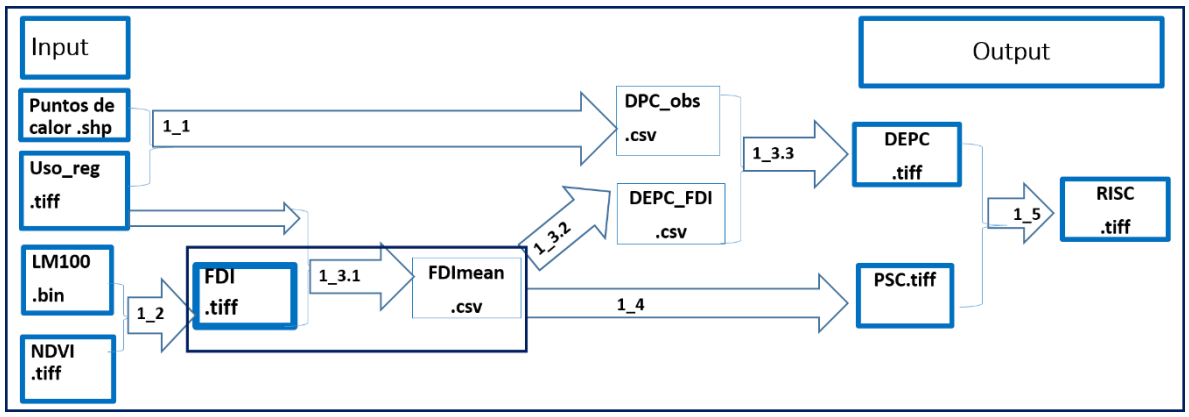



**Diagrama General: Proceso de cálculo del módulo de peligro meteorológico.**

Los números en las flechas denotan los subprocesos que serán detallados en diagramas de subprocesos a continuación.

**Entradas (inputs):**

- **Puntos de calor.shp:** Shapefile de puntos de calor depositado diariamente por CONABIO.
- **Uso_reg.tff**: archivo ráster formato .tiff de uso región. Capa fija.
- **LM100.bin**: archivo ráster formato .bin de humedad de 100 horas depositado diariamente por CONABIO.
- **NDVI.tiff**: archivo ráster formato .tiff de compuesto de 10 días de NDVI depositado semanalmente (cada 10 días) por CONABIO.

**Salidas (outputs):**

- **FDI (Fuel dryness Index):** Archivo ráster en formato .tiff del **Índice de Sequedad del Combustible (ISC).** Se calcula un archivo diario con la fecha del día.
- **DEPC (Densidad Esperada de Puntos de Calor):** Archivo ráster en formato .tiff con la densidad esperada de puntos de calor para cada uso región. Se calcula un archivo diario con la fecha del día.
- **PSC (Porcentaje de incendios por categorías de sequedad del combustible):** Archivo ráster en formato .tiff con el porcentaje de incendios por categorías del índice de sequedad del combustible. Se calcula un archivo diario con la fecha del día.
- **RISC (Riesgo de Incendio por Sequedad del Combustible)-Peligro meteorológico:** Archivo ráster en formato .tiff con el Peligro metereológico. Se calcula un archivo diario con la fecha del día.

## PROCESO 1. Calculo de FDI y de DEPC.

Se resume el proceso 1 en el diagrama a continuación:

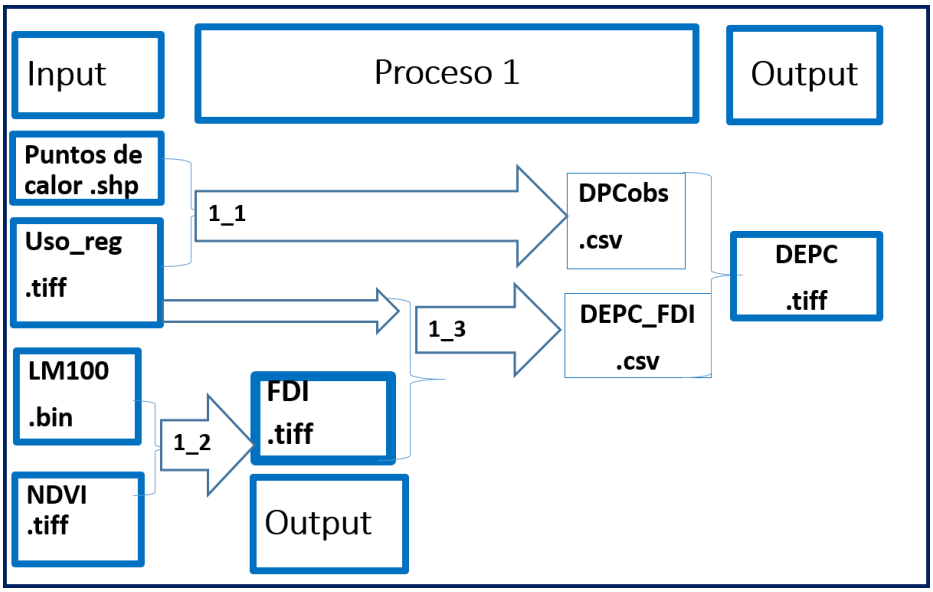



**Diagrama del proceso 1.**

Consta de los siguientes subprocesos:

**1_1: Cálculo de DPC obs**

**1_2: Cálculo de FDI.**

**1_3.1. Y 1_3.2: Cálculo de DEPC_FDI y de DEPC.**

A continuación se detallan los subprocesos del proceso 1.

### 1_1: Cálculo de DPC obs.

El diagrama 1_1 a continuación resume el subproceso de cálculo de DPCobs.

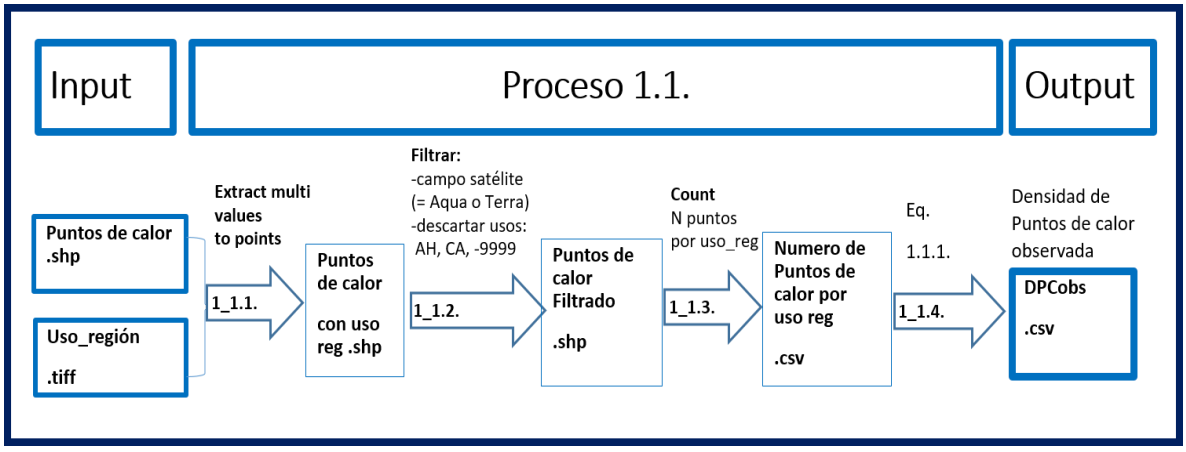



**Diagrama de proceso 1_1: Calculo de DPCobs**

**Entradas (inputs):**

- **Puntos de calor.shp:** Shapefile de puntos de calor depositado diariamente por CONABIO.
- **Uso_reg.tff**: archivo ráster formato .tiff de uso región. Capa fija.

**Salidas (outputs):**

- **DPCobs (Densidad observada de Puntos de Calor):** Archivo ráster en formato .tiff con la densidad observada de puntos de calor para cada uso región cada día. Se calcula un archivo diario con la fecha del día.

Los números en las flechas del diagrama denotan las operaciones detalladas a continuación.

**Proceso.**

**1_1.1. Extract multi values to points.** Extraer el valor del .tiff uso_region al shape de puntos de calor.

**1_1.2. Filtrar puntos de calor.**

-Filtrar por campo satélite= Aqua o Terra;

-Filtrar descartar usos AH, CA, -9999

**1_1.3. Calcular número de puntos por uso_region (N puntos).**

**1_1.4. Calcular DPCobs (Densidad de Puntos de Calor observada) para cada uso región para cada dìa según eq. 1.1.**

| |
|---|
| Eq. 1.1.: DPCobs= Npuntos/ Sup* 200000 |
| Donde: Npuntos (número de puntos de calor observados en el día para cada uso_región. |
| Sup: Superficie de la uso_región en $km^2$ (count de celdas de 1 $km^2$ de cada uso_región) |

**Salida del proceso**: archivo DPCobs en formato CSV con fecha, uso_region, valor DPCobs

**1_2: Calculo de FDI.**

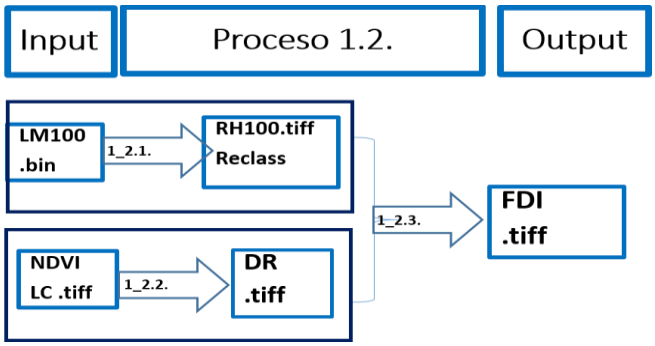



**Diagrama general de proceso 1_2: Calculo de FDI**

El proceso 1.2. consta de los siguientes subprocesos:

1_2.1: Calculo de RH100.

1_2.2: Calculo de DR.

1_2.3: Calculo de FDI

Se detallan a continuación los subprocesos.

**1_2.1: Calculo de RH100 (H100 Ratio)**

El diagrama 1_2_1 a continuación resume el subproceso de cálculo de RH100.

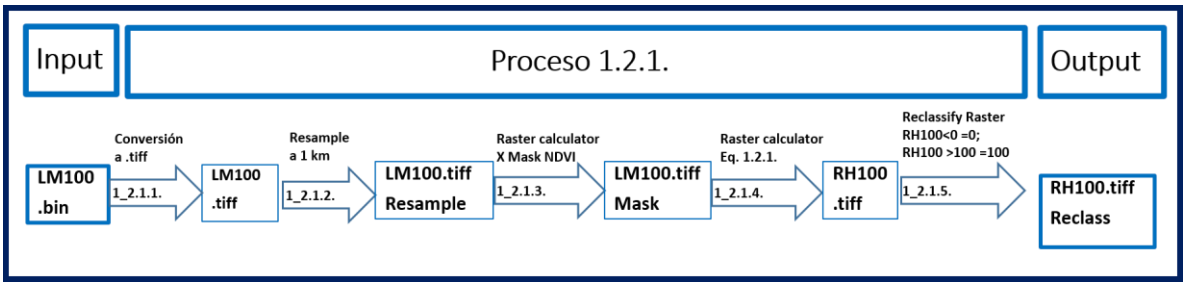



**Diagrama de subproceso 1_2.1: Calculo de RH100.**

Los números en las flechas del diagrama denotan las operaciones detalladas a continuación.

**Entrada (inputs):**

**LM100.bin.** 1 archivo por cada dia. Depositado diariamente por CONABIO

**Proceso.**

**1_2.1.1. Conversión de archivo LM100 de .bin a .tiff**

**1_2.1.2. Resample a 1km de LM100.tiff**

**1_2.1.3. Raster calculator x Máscara de NDVI**

**1_2.1.4. Raster calculator. Calcular RH100 (ecuación 1.2.1.1).**

| |
|---|
| Ecuación 1.2.1.: **RH100=** 100- (LM100 - HMIN) / (HMAX - HMIN) * 100<br>Donde: LM100: Imagen de Humedad de 100h .tiff diaria depositada por CONABIO<br>HMAX y HMIN: Capas fijas en formato .tiff con el máximo y mínimo de los históricos de LM100 |

**1_2.1.5. Reclassify raster:** RH100<0 =0; RH100 >100 =100.

**Salidas (outputs):**

**RH100reclass_dia.tiff.** Raster en formato .tiff de RH100 reclasificado. 1 archivo por cada día. **1_2.2: Calculo de DR (Dry Ratio).**

El diagrama 1_1_5a continuación resume el subproceso de cálculo de DR

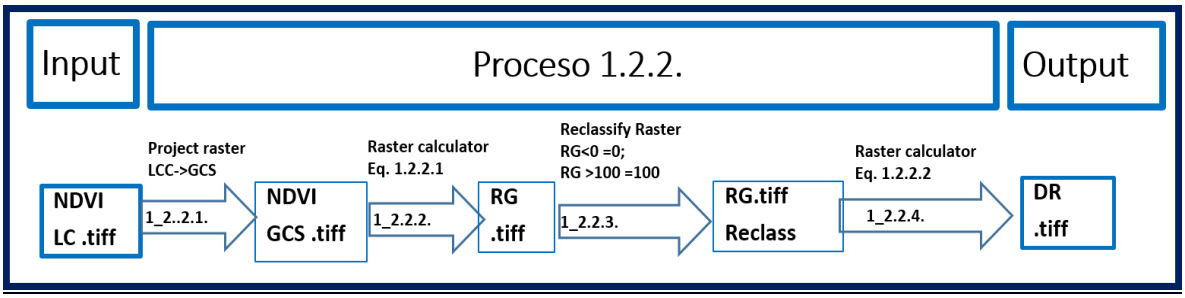



**Diagrama de subproceso 1_2.2: Calculo de DR.**

Los números en las flechas del diagrama denotan las operaciones detalladas a continuación.

**Entradas (inputs):**

**NDVI LC .tiff.** Archivo .tiff de NDVI en Lambert Conic (LCC). Depositado cada 10 días por
CONABIO.

**Proceso.**

**1_2.2.1. Project Raster.** Proyectar de LCC a GCS.

**1_2.2.2. Raster calculator. Calcular RG (Relative Greeness) (Ecuación 1.2.1.2.)**

Ecuación 1.2.2.1:

RG (Relative Greenness) = (NDVI_GCS - NDVI_min) / (NDVI_max-NDVI_min ) *100

Donde: NDVI_GCS: Archivo .tiff del compuesto de 10 días de NDVI proyectado a GCS (paso 1_2.2.1);

NDVImax y NDVImin: Capas fijas en formato .tiff con el máximo y mínimo de los históricos de los compuestos de NDVI de 10 días de CONABIO.

**1_2.2.3. Reclassify Raster:** RG<0 =0; RG >100 =100

**1_2.2.4. Raster calculator. Calcular DR (Dry Ratio) (Ecuación 1.2.2.)**

Ecuación 1.2.2.2.: DR=100-(RGreclass *LRmax/100)

Donde: RG: Relative Greeness reclasificado (salida proceso 1_2.2.3.)

LRmax= 30 +30 x (NDVImax-125)/(255-125)

Donde: NDVImax: Capas fija en formato .tiff con el máximo NDVI histórico.

**Salidas (outputs):**

**DR. Tiff.** Raster en formato .tiff de DR (Dry Ratio). 1 archivo por cada día.

**1_2.3: Calculo de FDI (Fuel Dryness Index) -Índice de Sequedad del Combustible (ISC).** El diagrama 1_2_3 a continuación resume el subproceso de cálculo de FDI

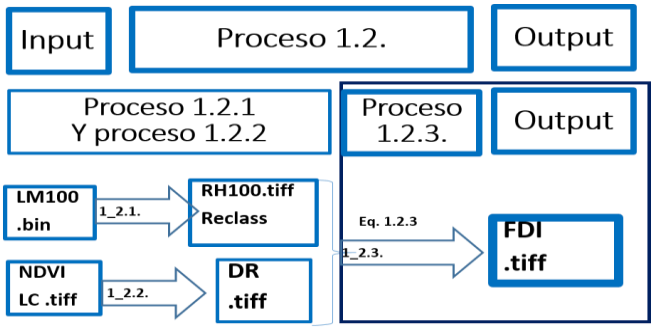



**Diagrama de subproceso 1_2.3: Calculo de FDI.**

Los números en las flechas del diagrama denotan las operaciones detalladas a continuación.

**Entradas (inputs):**

- **RH100.tiff :** salida del proceso 1.2.1.
- **DR.tiff:** salida del proceso 1.2.2. **Proceso.**

| |
|---|
| Ecuación 1.2.3: FDI= RH100 *DR/100<br><br>Donde: RH100: salida proceso 1_2.1; DR: salida proceso 1_2.2 |

**Salidas (outputs):**

**FDI. Tiff.** Raster en formato .tiff de FDI (Fuel Dryness Index)- Indice de Sequedad del Combustible (ISC). 1 archivo por cada día.

**1_3. 1. Cálculo de DEPC_FDI.**

El diagrama 1_3.1 a continuación resume el subproceso de cálculo de DEPC_FDI.

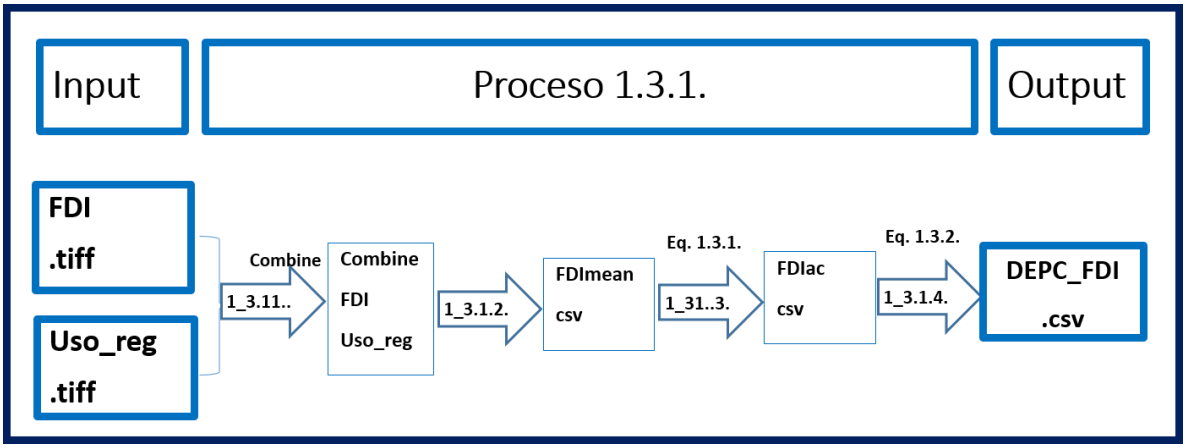



**Diagrama de subproceso 1_3.1: Calculo de DEPC_FDI.**

Los números en las flechas del diagrama denotan las operaciones detalladas a continuación.

**Entradas (inputs):**

- **FDI. Tiff.** salida del proceso 1.2.3.
- **Uso_reg.tff**: archivo ráster formato .tiff de uso región. Capa fija.

**Proceso.**

**1_3.1. Combine FDI.tiff y Uso_reg.tiff**

**1_3.2. Calcular FDImean.csv.** FDImean es el promedio de FDI para cada uso región, ponderado por la superficie observada de cada valor de FDI dividido entre la superficie total de cada uso región, calculado cada día.

**1_3.3. Calcular FDIac (FDI acumulado).**

1_3.3.1. Calcular FDImean-FDImin para cada dia

Donde FDImin= 50 para S; 60 para NO, NE, C (NBJ, N).

1_3.3.2. Calcular la suma de (FDImean-FDImin) para los 90 días anteriores.

1_3.3.3. Calcular FDIac (FDI acumulado) según Eq. 1.3.1.:

Eq. 1.3.1.: $FDIac = \sum(FDImean - FDImin) / 10$

Donde $\sum(FDI - FDImin)$ es la suma de los valores diarios de FDImean-FDImin en el periodo de 90 días anteriores a la fecha actual (1.3.3.2)

FDIac: FDIacumulado

**1_3.4. Calcular DEPC_FDI según Eq. 1.3.2.:**

Eq. 1.3.2.: $DEPC\_FDI = a * FDIac^{b}$

Donde: FDIac: FDIacumulado (calculado en 1.3.3.) a y b son coeficientes para cada uso región según tabla 1 adjunta.

**Salidas (outputs):**

**DEPC_FDI. Csv** Archivo .csv con la densidad de puntos de calor esperada-1.

DEPC_FDI**.** representa la fracción de la densidad de puntos de calor esperada debida a la sequedad del combustible (FDI). Se combina con la DEPCobs (densidad de puntos de calor observada) de los 3 días previos (salida del proceso 1.1) para el cálculo de la DEPC (Densidad Esperada de Puntos de Calor) diaria, en el paso 1.3.2. descrito a continuación.

## 1.3.2. Cálculo de DEPC (Densidad Esperada de Puntos de Calor).

El proceso 1.3.2. toma como insumos las salidas de los procesos 1.1. y 1.3.1., tal y como se detalla en el diagrama a continuación:

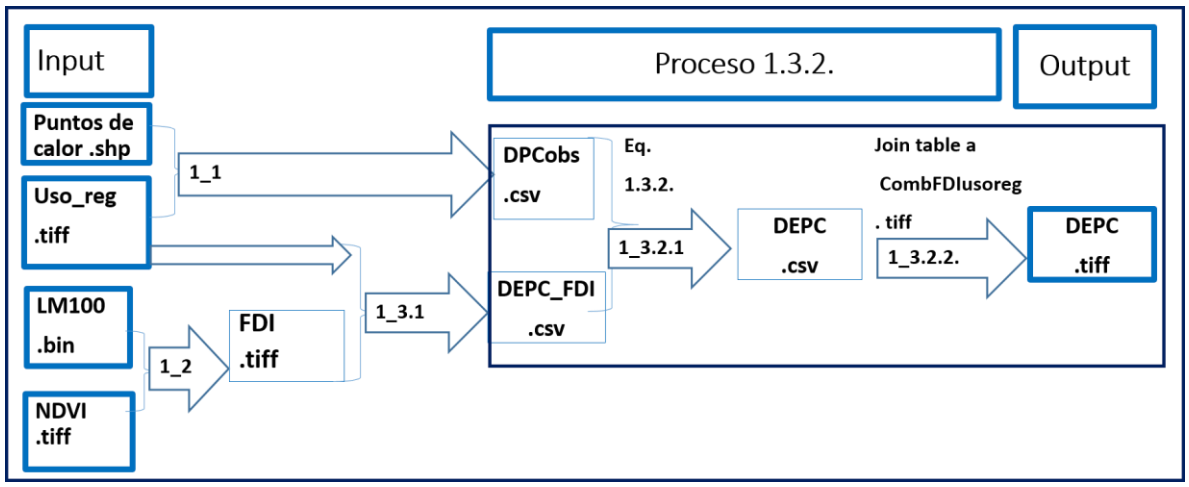



**Diagrama de subproceso 1_3.2: Calculo de DEPC**

Los números en las flechas del diagrama denotan las operaciones detalladas a continuación.

**Entradas (inputs):**

- **DPCobs. Csv.** salida del proceso 1.1
- **DEPC_FDI. Csv.** salida del proceso 1.2.3.

**Proceso.**

### 1.3.2.1. Calculo de DEPC (Densidad Esperada de Puntos de Calor) según eq.1.3.2.

Eq. 1.3.2: **DEPC**= DEPC_FDI + c * DPCobs3

Donde: DEPC_FDI: Salida del proceso 1.3.1. (DEPC_FDI.csv)

DPCobs3= promedio de DPCobs para cada uso región de 3 días anteriores de DPCobs.csv (salida del proceso 1_1).

C son coeficientes para cada uso región según tabla 1 adjunta.

La salida es un csv por día del día actual) con: Fecha,uso_reg, DPC1,DPCobs_diaAnterior,DPCpred

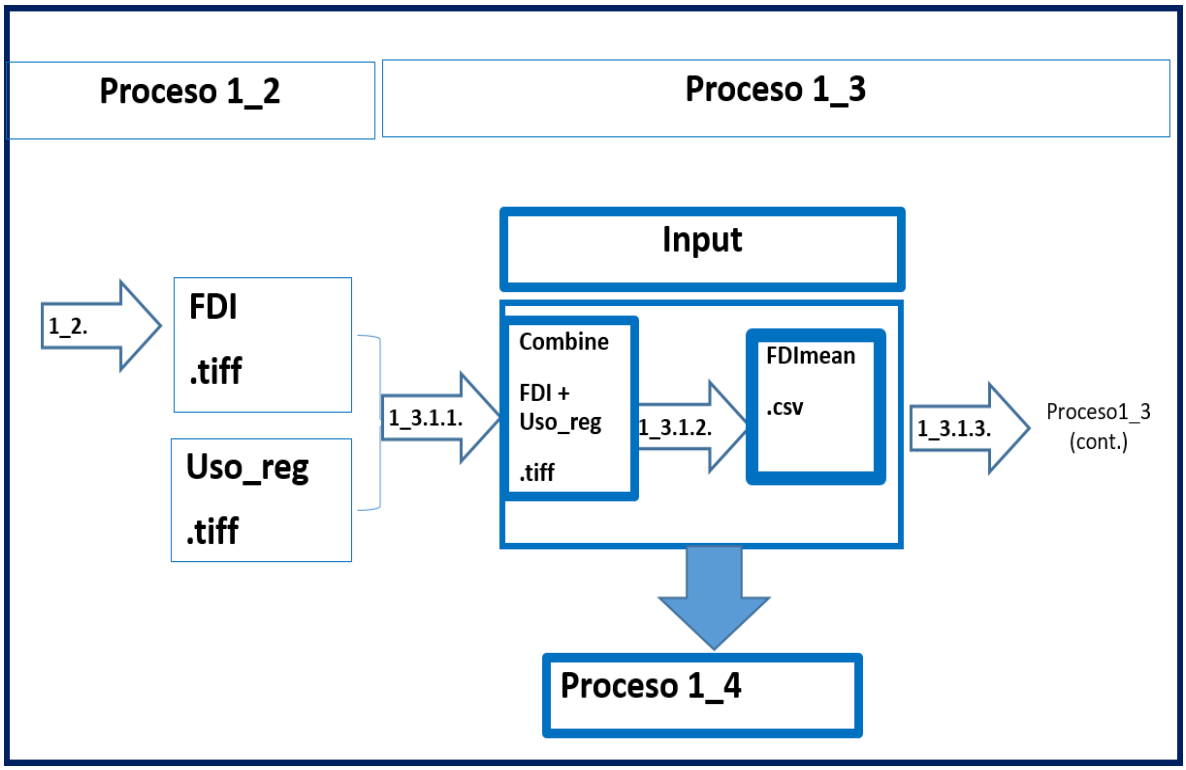
1.3.2.2. **Join table a CombFDIusoreg. Tiff**. Se une el DPEC.cvs al archivo combine de FDI del día con uso región (1.3.1.1.)

**Salida del proceso**

**DEPC.tiff.** Archivo raster en formato .tiff con la Densidad Esperada de Puntos de Calor para cada uso región cada día. Se genera un archivo por día.

## 1.4. Cálculo del PSC (Porcentaje de incendios por categorías de Sequedad del Combustible).

La figura a continuación ilustra la conexión del proceso 2_1 con el proceso 1_3 que proporcionan los insumos de este subproceso.




**Conexión del proceso 1_4 con el proceso anterior 1_3.**

El diagrama 1.4 a continuación detalla las operaciones del proceso 1.4de cálculo de PSC.

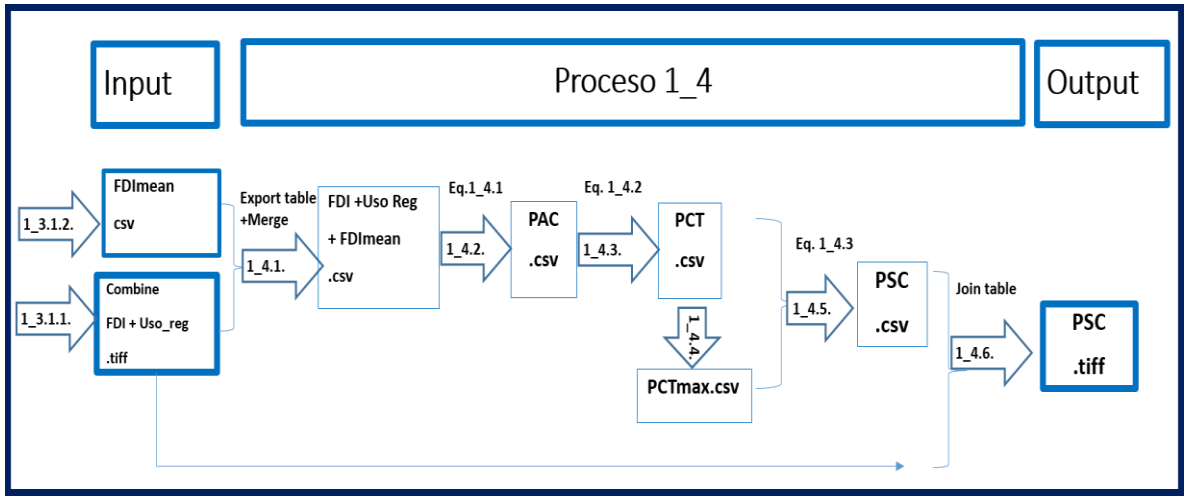



**Diagrama de subproceso 1.4: Calculo de PSC.**

Los números en las flechas del diagrama 1.4 denotan las operaciones detalladas a continuación.

**Entradas (inputs):**

Tal y como se detalla en los dos diagramas arriba, las entradas del proceso 1.4son:

- Combine FDI +Uso_reg.tiff. Salida del proceso 1_3.1.1.
- FDImean.csv. Salida del subproceso 1_3.1.2.

**Proceso.**

**1.4.1. Unir la tabla de Combine FDI + Uso_reg .tiff y el FDImean.csv.**

Se realizan las dos siguientes operaciones:

1.4.1.1. Export table Combine FDI + Uso_reg .tiff. Guardarla como .csv.

1.4.1.2. Merge table Combine FDI + Uso_reg .tiff y FDImean.csv

**1.4.2. Cálculo de PAC (Porcentaje de incendios Acumulado) según eq. 1.4.1.**

El cálculo consta de los siguientes procesos sobre el archivo FDI+usoreg+FDImean.csv:

1.4.1.4. Ordenar por uso_reg , luego por FDI_fechadia

1.4.2.2. Unir tabla de coeficientes (c0, c2, b0, b2) por uso región (tabla 2 adjunta).

1.4.2.3. Filtrar y eliminar registros con columna c0 vacia (usos de AH y CA).

A continuación se aplican las eqs. 1.4.1.1y 1.4.1.2 para el cálculo de coeficientes por uso región para cada día y 1.4.1.3 para el cálculo de PAC para cada valor de FDI y uso región cada día.

**1.4.2.3. Calcular los coeficientes c y b para cada uso región ( eq. 1.4.1.1 y 1.4.1.2)**

**Cálculo de coeficiente c:**

**eq. 1.4.1.1:** $c = c0 + c2 * FDImean$

donde: c0 y c2 son coeficientes por uso región (tabla 2); FDImean: es el valor promedio ponderado por la superficie de FDI para cada uso región en el día observado (1.3.1.2)

**Cálculo de coeficiente b:**

**eq. 1.4.1.2:** $b = b2 + Ln(FDImean)$ donde: b0 y b2 son

coeficientes por uso región (tabla 2)

Ln(FDImean) = logaritmo neperiano de FDImean (1.3.1.2)

1.4.2.4. **Calcular Porcentaje Acumulado de Incendios (PAC)** según ecuación 1.4.1.3:

Eq. 1.4.1.3: $PAC = 1 - \exp(-((FDI/b)^c))$

Donde: FDI: valores de FDI observados, para cada uso región cada día.

b y c son coeficientes calculados en el paso 1.4.2.3. según las eqs. 1.4.1.1y 1.4.1.2

El resultado es un archivo .csv con una estima de PAC (Porcentaje acumulado de Incendios) para cada valor observado del índice de sequedad del combustible FDI en cada uso región, calculado cada día.

1.4.3. **Calcular PCT (Porcentaje predicho de incendios por categorías de sequedad del combustible).** Se desacumula la variable anterior (porcentaje acumulado) para valores consecutivos de FDI. Para ello se realizan las operaciones:

1.4.3.1. Ordenar los datos por uso región y valor de FDI

1.4.3.2. Restar valores consecutivos (desacumular la variable PAC).

1.4.3.3. En el cambio de uso región (primera observación de PAC para cada uso región) no se resta al anterior, el PCT vale el primer valor de PAC observado.

1.4.4. **Calcular PCTmax**. A partir de cada archivo diario de PCT.csv, se calcula el máximo predicho de PCT para cada uso región.

**1.4.5. Calcular PSC (Porcentaje de Incendios por categorías de Sequedad de Combustible**).csv según eq. 1.4.3.

eq. 1.4.3: PSC= PCT/PCTmax *200

Donde:

PCT: Porcentaje predicho de incendios por categorías de sequedad del combustible (paso 1.4.3) PCTmax: valor máximo de PCT para cada uso región cada día (paso 1.4.4)

La salida es un .csv con los valores predichos de Porcentaje de Incendios por categorías de Sequedad de Combustible (PSC) para cada valor observado del índice de sequedad del combustible FDI, para cada uso región. Se calcula un archivo PSC.csv cada día.

**1.4.6. Unir la tabla del archivo PSC.cvs al ráster .tiff del combine diario de FDI y uso región.**

- **<u>Salida del proceso</u> PSC.tiff.**

La salida es un ráster en formato .tiff con los valores predichos de Porcentaje de Incendios por categorías de Sequedad de Combustible (PSC) para cada valor observado del índice de sequedad del combustible FDI, para cada uso región. Se calcula un archivo PSC.tiff cada día.

**1_5. Cálculo del Peligro meteorológico (RISC).**

El diagrama a continuación detalla el proceso 1_5para el cálculo del índice de peligro meteorológico RISC.

**Entradas (inputs):**

- **DEPC (Densidad Esperada de Puntos de Calor)**. Archivo .tiff. Salida del proceso 1.
- **PSC (Porcentaje de Incendios por categorías de Sequedad del Combustible)**. Archivo .tiff. Salida del proceso 1.4.

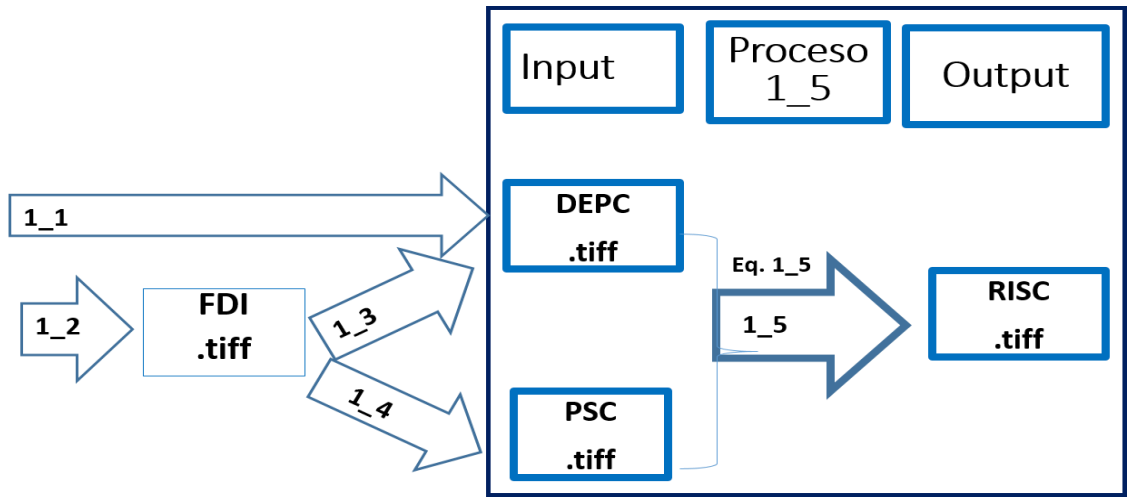



**Diagrama del proceso 1_5 para el cálculo del índice de peligro meteorológico RISC.**

**Proceso.**

**1_5. Cálculo de Índice peligro meteorológico RISC.** Se aplica en ráster calculator la ecuación 1.5

| |
|---|
| Eq. 1.5: |
| RISC= DEPC*PSC /100 |
| Donde: |
| DEPC (Densidad Esperada de Puntos de Calor). Salida del proceso 1.3; |
| PSC (Porcentaje Incendios por categorías de Sequedad del Combustible). Salida del proceso 1.4. |

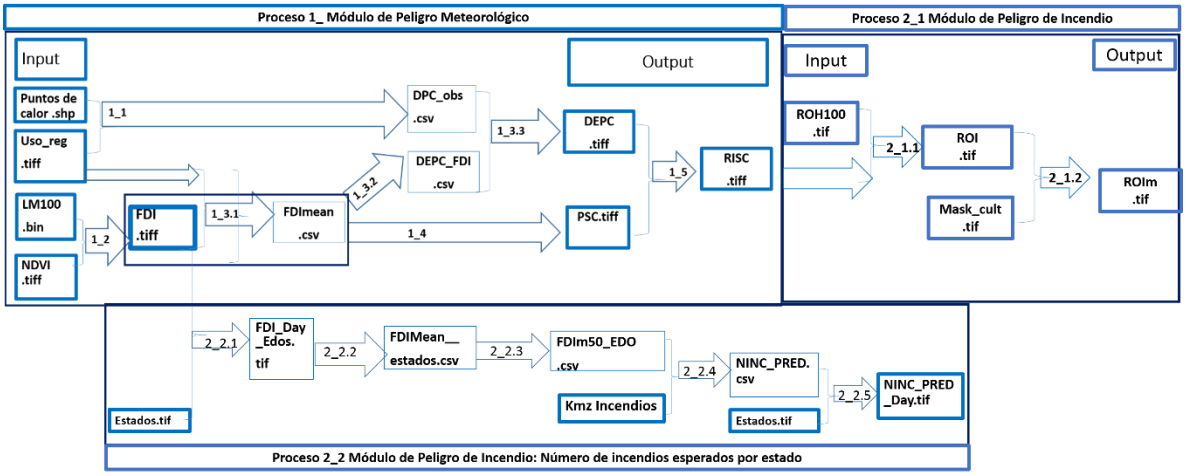
**Salida del proceso:**

**RISC. Tiff.** Archivo ráster en formato .tiff con el Índice RISC de peligro meteorológico para cada valor de uso región e índice ISC de cada día. Se calcula un archivo por día.

## 2. PROCESO DE CÁLCULO DEL MODULO DE PELIGRO DE INCENDIO.

El módulo de peligro de incendio consta de dos subprocesos, tal y como se ilustra en el diagrama a continuación:

**2.1. Cálculo del peligro de incendio.**
**2.2. Cálculo del número de incendios esperados por estado.**




**Integración del módulo de peligro de incendio con el módulo de peligro meteorológico.**

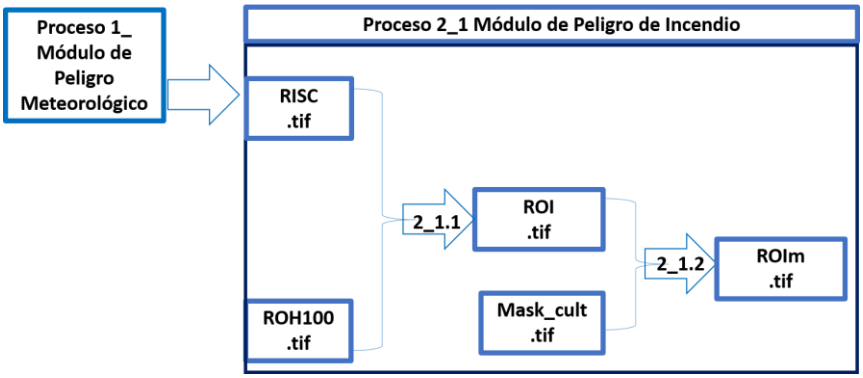
### 1.1. CÁLCULO DEL PELIGRO DE INCENDIO.

El índice diario de **peligro de incendio** se calcula diariamente a partir de las salidas de los módulos de peligro meteorológico y del mapa del módulo de riesgo de incendio. En concreto, utiliza como **entradas:**

-índice de **Peligro Meteorológico RISC** del módulo meteorológico (descrito arriba en 1.5.).

-Mapa de **Riesgo de Incendio ROH100**.tif

La figura a continuación ilustra el proceso.




**Proceso para el cálculo del índice de Peligro de Incendio.**

**Proceso.**

El cálculo del peligro de incendio consta de los siguientes subprocesos:

**2.1.1. Cálculo del ROI.**

Se emplea la ecuación 2.1.1.:

Eq. 2.1.1.:

ROI= RISC X ROH100 /100

Donde:

ROI: Peligro de Incendio sin filtrar, archivo raster en formato .tiff de Peligro de incendio sin filtrar áreas agrícolas, calculado diariamente.

RISC: Peligro Meteorológico, archivo raster en formato .tiff de Peligro Meteorológico, calculado diariamente en el módulo meteorológico (1.5)

ROH100: Riesgo de Incendio: mapa de riesgo de incendio por factores humanos en formato .tiff

2**.1.1. Cálculo del ROIm.**

Se emplea la ecuación 2.1.1.:

Eq. 2.1.1.:

ROIm= ROIxmask_cult

Donde:

ROIm: Peligro de Incendio, archivo raster en formato .tiff de Peligro de incendio, calculado diariamente

ROI: Peligro de Incendio sin filtrar, salida preoceso 2.1.1.

Mask_cult: archivo .tiff con mascara de cultivos agrícolas.

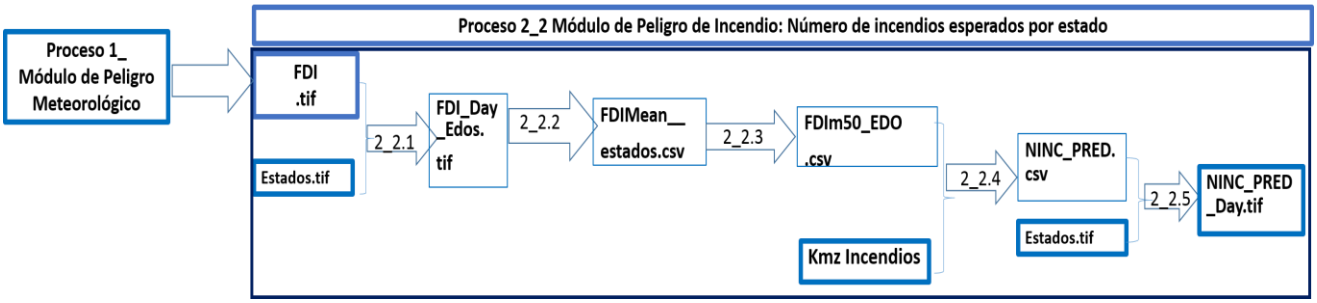
**Salida:**

ROIm.tiff: **Peligro de Incendio**, archivo raster en formato .tiff de Peligro de incendio, calculado diariamente

La salida ROIm_dia.tiff es subida diariamente a la sección de Peligro Incendio-> **Peligro de incendio.**

## 2.2. CÁLCULO DEL NÚMERO DE INCENDIOS ESPERADO POR ESTADO.

El diagrama a continuación describe los procesos intermedios para el cálculo del Número de incendios esperado por estado.




**Proceso de cálculo del Número de incendios esperado por estado**

**Entradas:**

**FDI_Day.tiff:** Archivo diario en formato tiff del Índice de Sequedad del Combustible del módulo de riesgo meteorológico, descrito en el apartado 1_2.3, donde Day= fecha.

**Estados. tif:** Capa fija en formato .tiff a resolución de 1 km con los 32 estados de la Republica Mexicana.

**Kmz incendios:** archivo diario de incendios activos, depositado por CONAFOR.

**Proceso.**

Se describe a continuación el proceso:

**2.2.1. Combinación de los archivos ráster de FDI diario y estados.tif**
**2.2.2. Cálculo del FDI promedio por estado y día**
**2.2.3. Cálculo del FDI promedio de los 50 días antecedentes.**
**2.2.4. Cálculo del .csv de Número de incendios esperado por estado.**
**2.2.5. Cálculo del .tif de Número de incendios esperado por estado.**

Se describen a continuación los subprocesos:

### 2.2.1. Combinación de los archivos ráster de FDI diario y estados.

Mediante la herramienta combine de ArcGIS, se obtiene un archivo ráster que combina el índice de sequedad del combustible diario (**FDI_Day.tif**) con los estados de la república, **FDI_Day_Edos.tif,** donde Day es la fecha.

### 2.2.2. Cálculo del FDI promedio por estado y día.

A partir del archivo ráster **FDI_Day_Edos.tif** se calcula el promedio por estado del índice de sequedad del combustible diario en un archivo formato .csv, **FDIMean_estados.csv.** El archivo contiene el promedio del índice de sequedad del combustible para cada día y estado.

### 2.2.3. Cálculo del FDI promedio de los 50 días antecedentes.

Para cada estado y día, se calcula el promedio de los 50 días antecedentes del índice de sequedad del combustible diario, según la eq 2.2.3:

Eq. 2.2.3:

$FDIm50_i = \sum_{i=n-50}^{n}(FDI_i)/50$

**Donde:**

FDIm50: Promedio de 50 días del índice de sequedad del combustible para cada estado;

$FDI_i$: índice de sequedad del combustible promedio para cada estado en el día *i*.

El resultado es un archivo **FDIm50_EDO.csv** con el promedio de los 50 días antecedentes de FDI para cada estado y día.

### 2.2.4. Cálculo del .csv de Número de incendios esperado por estado.

Para el cálculo del número de incendios esperados por estado, se emplea la eq. 2.2.4:

Eq. 2.2.4:

$NINC_{ij} = a * FDIm50_{ij}^{\ b} + c * NINC_{i-10\,j}$

Donde:

$NINC_{ij}$: Número de incendios esperado por periodo de 10 días *i* en el estado *j;*

$FDIm50_{ij}$ : Índice de sequedad promedio de 50 días por periodo de 10 días *i* en el estado *j* (eq. 1, proceso 3);

$NINC_{i-1\,j}$ : Número de incendios observado en los 10 días anteriores *i* -10 en el estado *j;*

a, b y c son coeficientes del modelo. La tabla 3 en anexos resume los coeficientes obtenidos para la predicción de número de incendios esperados por estado,

La salida del proceso es un archivo .csv con el número de incendios esperados por estado, **NINC_PRED.csv**

### 2.2.5. Cálculo del .tif de Número de incendios esperado por estado.

Mediante la herramienta combine de ArcGIS, se combina el archivo **NINC_PRED.csv** de proceso 4 con el archivo .tiff de estados, para obtener un ráster en formato .tiff con el pronóstico del **Número de incendios esperado por estado** para los próximos 10 días, **NINC_PRED_Day.tif**.

**Salida:**

**NINC_PRED_Day.tif**, con el pronóstico de número de incendios esperados por estado. El pronóstico se sube diariamente al sistema dentro de la sección de **Peligro de Incendio-> Número de incendios esperado por estado**.

**ANEXOS.**

**Tabla 1. Coeficientes a y b para el cálculo de DEPC_FDI y coeficientes c para el cálculo de DEPC predicho para cada uso región.**

| | Coeficientes a y b para el cálculo | | Coeficientes c para cálculo de |
|---|---|---|---|
| **Uso Region** | **a** | **b** | **c** |
| BOSQ____C | 63.5 | 1.63 | 0.89 |
| BOSQ____S | 150.3 | 1.84 | 0.63 |
| BOSQ___NC | 3.6 | 3.77 | 0.62 |
| BOSQ___NE | 91.2 | 1.57 | 0.91 |
| BOSQ___NO | 21.0 | 3.23 | 0.72 |
| BOSQ__NBJ | 57.4 | 0.86 | 0.39 |
| BTEH____S | 171.1 | 0.80 | 0.92 |
| BTES____S | 238.0 | 0.96 | 0.88 |
| BTH_____S | 176.0 | 0.74 | 0.82 |
| BTS_____C | 51.6 | 1.23 | 0.93 |
| BTS_____S | 238.0 | 0.96 | 0.88 |
| BTS____NE | 103.6 | 0.50 | 0.52 |
| BTS____NO | 28.7 | 1.44 | 0.80 |
| BTS___NBJ | 27.0 | 1.17 | 0.84 |
| CHAP__NBJ | 45.0 | 3.20 | 0.90 |
| CULT____C | 39.2 | 1.28 | 0.89 |
| CULT____S | 51.6 | 0.96 | 0.95 |
| CULT___NC | 11.8 | 0.67 | 0.82 |
| CULT___NE | 95.3 | 0.62 | 0.79 |
| CULT___NO | 19.0 | 1.54 | 0.68 |
| CULT__NBJ | 29.8 | 0.63 | 0.83 |
| HUM____S | 204.0 | 1.90 | 1.03 |
| MAT_____C | 38.1 | 0.68 | 0.69 |
| MAT____NC | 4.3 | 2.83 | 0.34 |
| MAT____NE | 13.1 | 2.93 | 0.71 |
| MAT___NBJ | 12.0 | 0.98 | 0.24 |
| PASNAT__C | 18.8 | 1.82 | 0.92 |
| PASNAT_NC | 5.4 | 1.55 | 0.56 |

Donde: a, b y c son coeficientes; BOSQ= Boque templado, BTEH=Bosque tropical estacionalmente húmedo, BTES= Bosque Tropical estacionalmente seco, BTH= Bosque tropical húmedo, BTS= Bosque tropical seco, CHAP= Chaparral, CULT= cultivo agrícola, MAT= matorral, HUM= humedal, PASNAT= pastizal natural; C, NC, NO, NBJ, S, NE son regiones definidas para el sistema de peligro donde: C=Centro y occidente, S=Sur, NC=Norte, NE=Noreste, NO=Noroeste-Sierra Madre Occidental, NBJ=Baja California.

**Tabla 2. Coeficientes para el cálculo de los coeficientes c y b para el cálculo de PSC.**

| | Coeficientes c0 y c2 | | Coeficientes b0 y b2 | |
|---|---|---|---|---|
| **Uso región** | **c0** | **c2** | **b0** | **b2** |
| BOSQ____C | 0.07 | 1 | 0.6 | 23.87 |
| BOSQ____S | 0.13 | 0.88 | 0.8 | 20.66 |
| BOSQ___NC | 0.07 | 1.4 | 0.69 | 26.45 |
| BOSQ___NE | 0.13 | 0.91 | 0.69 | 24.6 |
| BOSQ___NO | 0.11 | 1.06 | 0.73 | 24.62 |
| BOSQ__NBJ | 0.07 | 1.4 | 0.69 | 26.45 |
| BTEH____S | 0.84 | 0.5 | 0.78 | 22.47 |
| BTES____S | 0.44 | 0.58 | 0.79 | 21.37 |
| BTH_____S | 0.44 | 0.58 | 0.79 | 21.37 |
| BTS_____C | 0.09 | 1.19 | 0.77 | 25.37 |
| BTS_____S | 0.44 | 0.58 | 0.79 | 21.37 |
| BTS____NE | 0.19 | 0.76 | 0.77 | 20.78 |
| BTS____NO | 0.07 | 1.4 | 0.69 | 26.45 |
| BTS___NBJ | 0.07 | 1.4 | 0.69 | 26.45 |
| CHAP__NBJ | 0.07 | 1.4 | 0.69 | 26.45 |
| CULT____C | 0.08 | 1.22 | 0.9 | 15.15 |
| CULT____S | 0.24 | 0.73 | 0.76 | 22.29 |
| CULT___NC | 0.08 | 1.21 | 0.7 | 28.08 |
| CULT___NE | 0.19 | 0.8 | 0.83 | 17.14 |
| CULT___NO | 0.48 | 0.69 | 0.49 | 45.5 |
| CULT__NBJ | 0.07 | 1.4 | 0.69 | 26.45 |
| HUM_____S | 0.08 | 1.18 | 1.06 | 3.72 |
| MAT_____C | 0.07 | 1.4 | 0.69 | 26.45 |
| MAT____NC | 0.07 | 1.4 | 0.69 | 26.45 |
| MAT____NE | 0.07 | 1.4 | 0.69 | 26.45 |
| MAT___NBJ | 0.07 | 1.4 | 0.69 | 26.45 |
| PASNAT__C | 0.15 | 0.97 | 0.88 | 15.81 |
| PASNAT_NC | 0.07 | 1.4 | 0.69 | 26.45 |

Donde:

c0, c2, b0, b2 son coeficientes; BOSQ= Boque templado, BTEH=Bosque tropical estacionalmente húmedo, BTES= Bosque Tropical estacionalmente seco, BTH= Bosque tropical húmedo, BTS= Bosque tropical seco, CHAP= Chaparral, CULT= cultivo agrícola, MAT= matorral, HUM= humedal, PASNAT= pastizal natural; C, NC, NO, NBJ, S, NE son regiones definidas para el sistema de peligro donde: C=Centro y occidente, S=Sur, NC=Norte, NE=Noreste, NO=Noroeste-Sierra Madre Occidental, NBJ=Baja California.

**Tabla 3. Coeficientes a, b y c para la predicción del el número de incendios esperados por estado según la ecuación 2.2.3.**

| Estado | a | b | c |
|---|---|---|---|
| Aguascalientes | 10.0 | 1.5 | 0.27 |
| Baja California | 34.7 | 3.2 | 0.66 |
| Baja California Sur | 9.7 | 1.6 | 0.26 |
| Campeche | 27.4 | 2.4 | 0.40 |
| Coahuila | 36.3 | 3.7 | 0.73 |
| Colima | 78.9 | 5.5 | 0.27 |
| Chiapas | 103.2 | 3.2 | 0.73 |
| Chihuahua | 280.0 | 6.3 | 0.82 |
| Ciudad de México | 645.2 | 7.1 | 0.73 |
| Durango | 99.6 | 5.1 | 0.69 |
| Guanajuato | 41.0 | 5.9 | 0.07 |
| Guerrero | 25.5 | 1.0 | 0.59 |
| Hidalgo | 78.7 | 2.7 | 0.64 |
| Jalisco | 77.5 | 2.7 | 0.85 |
| México | 654.9 | 6.1 | 0.73 |
| Michoacán | 276.8 | 5.4 | 0.77 |
| Morelos | 37.7 | 2.8 | 0.51 |
| Nayarit | 12.9 | 1.0 | 0.73 |
| Nuevo León | 17.6 | 2.0 | 0.73 |
| Oaxaca | 99.6 | 4.2 | 0.70 |
| Puebla | 161.3 | 4.3 | 0.57 |
| Querétaro | 102.9 | 7.4 | 0.32 |
| Quintana Roo | 25.0 | 1.5 | 0.65 |
| San Luis Potosí | 107.7 | 8.9 | 0.46 |
| Sinaloa | 33.9 | 4.4 | 0.21 |
| Sonora | 24.2 | 3.1 | 0.35 |
| Tabasco | 25.0 | 1.5 | 0.65 |
| Tamaulipas | 25.5 | 3.4 | 0.25 |
| Tlaxcala | 90.8 | 3.6 | 0.65 |
| Veracruz | 150.3 | 3.9 | 0.61 |
| Yucatán | 109.1 | 5.5 | 0.29 |
| Zacatecas | 20.3 | 3.6 | 0.56 |

Donde: a, b y c son coeficientes para la predicción del el número de incendios esperados por estado según la ecuación 2.2.3.

## ANEXO 2.

## CÓDIGO DEL SISTEMA DE PREDICCIÓN DE PELIGRO DE INCENDIOS FORESTALES PARA MÉXICO.

```
# -*- coding: utf-8 -*-
#
# Proceso diario de incendios
#
# Autor: Jaime Briseño Reyes <jaime.briseno@gmail.com>
# Fecha: 09/Sep/2017
#

import os
import shutil
import glob
import time
import funciones
import arcpy

# Habilita el uso de la extensón Spatial de ArcGis
arcpy.CheckOutExtension("Spatial")

today = time.strftime("%d/%m/%Y")

rutaBase = "D:/proyectos/incendios/proceso_diario"
rutaBasePaginaWEB = "C:/xampp/htdocs/incendios"
rutaFTP = "D:/ftp/conabio_incendios/data"

#
# Diccionario para almacenar las variables de configuración
# estas variables deben cambiarse si hay un cambio de servidor
#
dicParam = {
    "rutaBase": rutaBase,
    "rutaBasePaginaWEB": rutaBasePaginaWEB,
    "rutaFTP": rutaFTP,
    "fechaHoy": today,
    "fechaHoyDia": today.split("/")[0],
    "fechaHoyMes": today.split("/")[1],
    "fechaHoyAnio": today.split("/")[2],
    # puntos MODIS
    "nombreArchivoMODIS": "puntoscalormodis_viirs.shp",
    # _1_1_DPC
    # "rutaFTPpuntosMODIS": rutaFTP + "/HotSpot_MODIS/daily",
    "rutaFTPpuntosMODIS": rutaFTP + "/HotSpot_MODIS_VIIRS/daily",
    "rutaFTPLM100": rutaFTP + "/MH_daily",
    "rutaEntradaPuntosMODIS": rutaBase + "/in/HotSpot_MODIS/daily",
    "rutaEntradaLM100bin": rutaBase + "/in/MH_daily/bin",
    "archivoLM100binHDR": rutaBase + "/in/MH_daily/bin_hdr/LM100h_Day.bin.hdr",
    "rutaEntradaLM100tif": rutaBase + "/in/MH_daily/tif",
    "rutaSalidaDPC": rutaBase + "/out/_1_1_DPC_obs",
    # _1_2_FDI
    "rutaFTPNDVI": rutaFTP + "/NDVI_10days/filled",
    "rutaEntradaNDVI": rutaBase + "/in/NDVI_10days",
    "NDVImin": rutaBase + "/in/NDVI/NDVI_min/NDVI_min.tif",
    "NDVIdiff": rutaBase + "/in/NDVI/NDVI_diff/NDVI_diff.tif",
    "rutaSalidasFDI": rutaBase + "/out/_1_2_FDI",
    "FDI_H100_HMIN": rutaBase + "/in/RH100/HMIN/HMIN.tif",
    "FDI_H100_HMAX": rutaBase + "/in/RH100/HMAX/HMAX.tif",
    "FDI_H100_mask_NDVI_1": rutaBase + "/in/RH100/mask_NDVI_1/mask_NDVI_1.tif",
    "rutaFDI_RH100": rutaBase + "/out/_1_2_FDI/RH100",
    "rutaFDI_DR_RG": rutaBase + "/out/_1_2_FDI/RG",
    "DR_LR_max": rutaBase + "/in/DR/LR_max/LR_max.tif",
    "rutaFDI_DR_DR": rutaBase + "/out/_1_2_FDI/DR",
    "rutaFDI_FDI": rutaBase + "/out/_1_2_FDI/FDI",
    # _1_3_DEPC
    "FDI_mean_csv": rutaBase + "/out/_1_3_DEPC/1b_FDI_mean_csv/FDI_mean.csv",
    "DEPC_FDI_combines_tif": rutaBase + "/out/_1_3_DEPC/1a_FDI_combines_tif",
    "l_uso_reg_a_omitir": [1, 3, 8, 9, 14, 15, 20, 21, 34, 36, 26, 28],
    "l_dias_uso_region_sur": [25, 27, 28, 29, 30, 31, 32, 33],
    "DEPC_archivo_parametros": rutaBase + "/in/DEPC/FDIac_30ecuaciones.csv",
    "DEPC_FDIac_csv": rutaBase + "/out/_1_3_DEPC/2_DEPC_FDIac_csv/DEPC_FDIac_FechA.csv",
    "rutaDEPC_FDIac_csv": rutaBase + "/out/_1_3_DEPC/2_DEPC_FDIac_csv",
    # 4 DEPC
    "DEPC_contante_c": rutaBase + "/in/DEPC/tabla_c.csv",
    "rutaDEPCsalida_csv": rutaBase + "/out/_1_3_DEPC/3a_DEPC_csv",
    "rutaDEPCsalida_tif": rutaBase + "/out/_1_3_DEPC/3b_DEPC_tif",
    # _1_4_PSC
    "archivoPSCEntradaFDImean": rutaBase + "/out/_1_3_DEPC/1b_FDI_mean_csv/FDI_mean.csv",
    "archivoPSCcoeficientes": rutaBase + "/in/PSC/Coeficientes.csv",
    "rutaPSCsalida_csv": rutaBase + "/out/_1_4_PSC/csv",
    "rutaPSCsalida_tif": rutaBase + "/out/_1_4_PSC/tif",
    # _1_5_RISC
    "rutaRISCsalidaPreliminar": rutaBase + "/out/_1_5_RISC/preliminar",
    "rutaRISCsalidaFinal": rutaBase + "/out/_1_5_RISC/final",
    # _1_6_ROI
    "ROI_inROI100": rutaBase + "/in/ROI/ROI100/ROH100.tif",
```

```
    "ROI_inROImask": rutaBase + "/in/ROI/Mask/mask_cult.tif",
    "ROI_outROIweb": rutaBasePaginaWEB + "/cartografia/capas/diarias/ROI/ROI.tif",


    # carpetas de salida para la pagina WEB
    "archivoTIFF_FDIsalidaDiaria": rutaBasePaginaWEB +
"/cartografia/capas/diarias/indice_de_combustible_seco/indice_de_combustible_seco.tif",
    "archivoTIFF_RISCsalidaDiaria": rutaBasePaginaWEB +
"/cartografia/capas/diarias/riesgo_de_ocurrencia_de_incendio/riesgo_de_ocurrencia_de_incendio.tif",


    "TIFFusoRegion": rutaBase + "/in/uso_region/uso_region.tif",
    "dirTMP": rutaBase + "/tmp",
    "cmd_gdal": "C:/OSGeo4W64/bin/gdal_translate.exe"
}


class INCENDIOS:

    dicParam = {}

    def __init__(self, dicParam):
        self.dicParam = dicParam

        #
        # Copia archivos diarios de FTP a ruta de trabajo
        #
        self.actualizaArchivosLocalesDesdeFTP()

        # Proceso 1.1 DPC
        #
        self._1_1_DPC()

        #
        # proceso 1.2 FDI
        #
        self._1_2_FDI()

        #
        # proceso 1.3 FDI acumulado
        #
        self._1_3_DEPC()

        #
        # Proceso 1.4 PSC
        #
        self._1_4_PSC()

        #
        # Proceso 1.4 PSC
        #
        self._1_5_RISC()

        #
        # Actualiza los archivos finales en la pagina WEB
        #
        self.actualizaPaginaWEB()

        #
        # Proceso 1.6 ROI
        #
        self._1_6_ROI()


    #
    # Actualiza archivos en la ruta de trabajo
    # Los copia de la ruta del FTP a la ruta de trabajo
    # Como se requiere modificar los archivos, esto no se debe hacer sobre los archivos originales del FTP
    #
    # 29/Ago/2017
    #
    # Parámetros: Entrada:shape de entrada, columna
    #
    def actualizaArchivosLocalesDesdeFTP(self):
        print "Actualizando carpetas locales desde sitio FTP..."
        funciones.actualizaArchivosLocalesDesdeFTP_fn(dicParam)

    #
    # Proceso DPC
    #
    # Proceso: DPC_2017-07-06.py
    def _1_1_DPC(self):
        print "___Procesando _1_1_DPC..."
        # Anteriomente se hizo con historicos,
        # ahora se hará con los 3 últimos días (3 últimas lecturas existentes [no son 3 días atrás]): ej: día 29 tomar 28,27 y 26
        #
        carpetas = glob.glob(self.dicParam['rutaEntradaPuntosMODIS'] + "/*")
        carpetas.sort() # Los ordena alfabeticamente que coincide cronológicamente
        ultimos3dias = carpetas[-3:]

        for rutaDePuntosDeCalorDiario in ultimos3dias:
            rutaDePuntosDeCalorDiario = rutaDePuntosDeCalorDiario.replace("\\", "/")
            archivoDePuntosDeCalorDiario = rutaDePuntosDeCalorDiario + "/puntoscalormodis.shp"
            fechaArchivo = rutaDePuntosDeCalorDiario.split("/")[-1].replace("-", "")
            dCountImageUsoRegion = funciones._1_1_DPC_obtieneCountImagenUsoRegion(self.dicParam['TIFFusoRegion'])
```

```
            dCountPuntosPorUsoRegion = funciones._1_1_DPC_obtieneCountDePuntosPorUsoRegion(archivoDePuntosDeCalorDiario,
self.dicParam['TIFFusoRegion'])
            funciones._1_1_DPC_combinaCounts(dCountImageUsoRegion, dCountPuntosPorUsoRegion, fechaArchivo,
self.dicParam['rutaSalidaDPC'])


    #
    # Proceso FDI
    #
    #
    def _1_2_FDI(self):
        # calculo_FDI_2017-06-09.py
        print "___Procesando _1_2_FDI..."
        # Prepara carpetas de salida
        funciones._1_2_FDI_prepara_carpeta_salida(self.dicParam["fechaHoyAnio"], self.dicParam["rutaSalidasFDI"])
        # Convierte .bin a .tif
        funciones._1_2_FDI_bin_a_tif(self.dicParam)

        # Recorre los archivos tif de entrada
        filesLMtiff = glob.glob(self.dicParam["rutaEntradaLM100tif"] + "/*.tif")
        for fileLMtiff in filesLMtiff:
            fileLMtiff = fileLMtiff.replace('\\', '/')
            # Obtiene la fecha del nombre del TIF para poder derminar la fecha del archivo NDVI utilizar
            filenameLM = fileLMtiff.split("/")[-1]

            # Si ya existe el archivo de salida omite este proceso
            archivoFDIsalida = self.dicParam["rutaFDI_FDI"] + "/" + (filenameLM.replace("LM100h_", "FDI_"))
            if not os.path.isfile(archivoFDIsalida):
                print "fileLMtiff:", fileLMtiff
                fechaLM = filenameLM.split("_")[1]
                # Solo se consideran los archivos .tif de entrada >= 11Ene2011 porque no tenemos NDVI para ellos
                if fechaLM > "20110110":
                    # Limpia carpeta temporal
                    funciones.borra_contenido_carpeta(self.dicParam["dirTMP"])

                    #RH
                    #
                    funciones._1_2_FDI_calculoRH(self.dicParam, fileLMtiff)

                    #DR
                    #
                    #fechaNDVI = funciones.obtieneFechaNDVI(fechaLM)
                    #diaNDVI = fechaNDVI[-2:]
                    #if diaNDVI == '01' or diaNDVI == '11' or diaNDVI == '21':
                    #    funciones._1_2_FDI_calculoDR(self.dicParam, fileLMtiff, fechaNDVI)
                    funciones._1_2_FDI_calculoDR(self.dicParam, fileLMtiff)

                    # FDI
                    #
                    funciones._1_2_FDI_ecuacion_4(self.dicParam, fileLMtiff)


    #
    # Proceso DEPC
    #
    def _1_3_DEPC(self):
        print "___Procesando _1_3_DEPC..."
        #
        # 1 FDI_combine
        # combine_uso_region_FDI_2017-06-14_Ok.py
        funciones._1_3_1_DEPC_FDI_combine(self.dicParam)

        # 1a FDI_mean
        # Si va
        #Xfunciones._1_3_1a_DEPC_FDI_mean(self.dicParam)

        # 2 FDI_acumulado   (Con ecuacion)
        # DPC1_2017-07-11.py
        funciones._1_3_2_DEPC_FDI_acumulado(self.dicParam)

        # 3 DEPC
        # DPCpred_2017-07-14.py
        funciones._1_3_3_DEPC(self.dicParam)



    #
    # Proceso 1.4 PSC
    #
    # 2017-07-25_PSC.py
    def _1_4_PSC(self):
        print "___Procesando _1_4_PSC..."
        funciones._1_4_PSC(self.dicParam)

    #
    # Proceso 1.5 RISC
    #
    # 2017-08-02_RISC.py
    def _1_5_RISC(self):
        print "___Procesando _1_5_RISC..."
        funciones._1_5_RISC(self.dicParam)

    #
    # Actualiza los archivos finales de salida en la pagina WEB
    #
```

```
    def actualizaPaginaWEB(self):
        print "___Actualizando pagina WEB..."
        funciones.actualizaPaginaWEB(dicParam)


    #
    # Proceso ROI
    #
    # 2018-06-01
    def _1_6_ROI(self):
        print "Proceso ROI..."
        funciones._1_6_ROI(dicParam)


################################################################################################
#
# Se instancia el objeto INCENDIOS para iniciar el proceso
#
obj = INCENDIOS(dicParam)
```

## Código de funciones

```
# -*- coding: utf-8 -*-
#
# Funciones de
# Proceso diario de incendios
#
# Autor: Jaime Briseño Reyes <jaime.briseno@gmail.com>
# Fecha: 09/Ago/2017
#

import os
import shutil
import glob
import datetime
from datetime import timedelta
import math
import arcpy


#############################################################################################
#
# Funciones comunes
#


#
# Borra una columna de un archivo shape
#
# 26/Jul/2017
#
# Parámetros: Entrada:shape de entrada, columna
#
def borraColumnaShape(archivoShape, columna):
    try:
        arcpy.DeleteField_management(archivoShape, [columna])
    except:
        pass


def creaColumnaUsoRegionEnShape(archivoShape):
    columna = "uso_region"
    borraColumnaShape(archivoShape, columna)
    arcpy.AddField_management(archivoShape, columna, "SHORT", "4", "", "", "", "NULLABLE", "NON_REQUIRED", "")


#
#  Determina la fecha del archivo NDVI que se tomará según a fecha del archivo LM (1, 11 o 21)
#
def obtieneFechaNDVI(fechaLM):
    anioLM = fechaLM[0:4]
    mesLM = fechaLM[-4:-2]
    diaLM = fechaLM[-2:]
    # Inicializa el año y el mes NDVI con los datos del LM
    anioNDVI = anioLM
    mesNDVI = mesLM
    # El 1° se usa el NDVI del dia 21
    if int(diaLM) >= 1 and int(diaLM) <= 10:
        diaNDVI = "21"
        # Si el mes actual es 1, le resta 1 al año y el mes es 12
        if int(mesLM) == 1:
            anioNDVI = str(int(anioLM) - 1)
            mesNDVI = 12
        else:
            mesNDVI = int(mesLM) - 1
    if int(diaLM) >= 11 and int(diaLM) <= 20:
        diaNDVI = "01"
    elif (int(diaLM) >= 21 and int(diaLM) <= 31):
```

```
        diaNDVI = "11"

    mesNDVI = ("0" + str(mesNDVI))[-2:]  # Rellena el cero a la izquierda por si le falta
    fechaNDVI = "%s%s%s" % (anioNDVI, mesNDVI, diaNDVI)
    return fechaNDVI


def borra_archivo(shapefile):
    if arcpy.Exists(shapefile):
        arcpy.Delete_management(shapefile)

def borra_contenido_carpeta(carpeta):
    archivos = glob.glob(carpeta + "/*")
    for archivo in archivos:
        os.remove(archivo)



#
# Para dias julianos
#
def leap_year(year):
    year = "%s" % year
    year = int(year)
    return (year % 4 == 0 and year % 100 != 0) or year % 400 == 0
def dia_juliano_a_fecha(anio, dia):
    year = anio
    leap = leap_year(year)
    days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    if leap == True:
        days[1] = 29
    day = dia
    sum = 0
    for k, item in enumerate(days):
        sum += item
        if sum >= day:
            dd = item - (sum - day)
            if k + 1 < 10 or dd < 10:
                mm = str(k + 1).zfill(2)
                dd = str(dd).zfill(2)
            else:
                mm = str(k + 1)
                dd = str(dd)
            formatted_day = str(year) + "-" + mm + "-" + dd
            return formatted_day
def creaDiccionario_diasJulianos_fechas(anio):
    leap = leap_year(anio)
    if leap == True:
        numeroDias = 366
    else:
        numeroDias = 365
    dicDiaJuliano_fecha = {}
    dicFecha_DiaJuliano = {}
    for diaJuliano in range(1, numeroDias + 1):
        fecha = dia_juliano_a_fecha(anio, diaJuliano)
        dicDiaJuliano_fecha[diaJuliano] = fecha
        dicFecha_DiaJuliano[fecha] = diaJuliano
    return {"dicDiaJuliano_fecha": dicDiaJuliano_fecha, "dicFecha_DiaJuliano":dicFecha_DiaJuliano}



#
# Terminan Funciones comunes
#
################################################################################################


################################################################################################################
#
# actualizaArchivosLocalesDesdeFTP_fn
#
def actualizaArchivosLocalesDesdeFTP_fn(dicParam):
    rutaInMODIS = dicParam["rutaFTPpuntosMODIS"]
    rutaOutMODIS = dicParam["rutaEntradaPuntosMODIS"]
    rutaInLM = dicParam["rutaFTPLM100"]
    rutaOutLM = dicParam["rutaEntradaLM100bin"]
    rutaInNDVI = dicParam["rutaFTPNDVI"]
    rutaOutNDVI = dicParam["rutaEntradaNDVI"]
    rutaDestinoSistema = dicParam["rutaBasePaginaWEB"]  + "/cartografia/capas/diarias/puntos_de_calor_observados/DIA"
    # Actualizacion 2018-07-07 08:00
    # La CONABIO cambio el nombre del archivo que deposita en el servidor FTP,
    # tanto de nombre como de carpeta
    # este cambio es para dejar el nombre del archivo depositado tal como se encontraba anteriormente
    # sin las letras "_viirs"
    #nombreArchivoMODISin = dicParam["nombreArchivoMODIS"]
    #nombreArchivoMODISout = nombreArchivoMODISin.replace("_viirs", "")

    nombreArchivoMODIS = dicParam["nombreArchivoMODIS"]

    #
    # Actualiza puntos MODIS a publicar en sitio WEB
    #
    print "Actualizando puntos MODIS en sitio WEB..."
    carpetas = glob.glob(rutaInMODIS + "/*")
    carpetas.sort()
```

```
    if len(carpetas) > 0:
        carpeta_hoy = carpetas[-1]
        carpeta_1_dia_atras = carpetas[-1]
        carpeta_2_dias_atras = carpetas[-1]
        carpeta_3_dias_atras = carpetas[-1]
    if len(carpetas) > 1:
        carpeta_hoy = carpetas[-1]
        carpeta_1_dia_atras = carpetas[-2]
        carpeta_2_dias_atras = carpetas[-2]
        carpeta_3_dias_atras = carpetas[-2]
    if len(carpetas) > 2:
        carpeta_hoy = carpetas[-1]
        carpeta_1_dia_atras = carpetas[-2]
        carpeta_2_dias_atras = carpetas[-3]
        carpeta_3_dias_atras = carpetas[-3]
    if len(carpetas) > 3:
        carpeta_hoy = carpetas[-1]
        carpeta_1_dia_atras = carpetas[-2]
        carpeta_2_dias_atras = carpetas[-3]
        carpeta_3_dias_atras = carpetas[-4]

    carpeta_hoy = carpeta_hoy.replace("\\", "/")
    carpeta_1_dia_atras = carpeta_1_dia_atras.replace("\\", "/")
    carpeta_2_dias_atras = carpeta_2_dias_atras.replace("\\", "/")
    carpeta_3_dias_atras = carpeta_3_dias_atras.replace("\\", "/")

    # Fecha hoy
    print "Actualizando Hoy..."
    carpeta_fecha = carpeta_hoy.split("/")[-1]
    archivo_entrada = carpeta_hoy + "/" + nombreArchivoMODIS
    # Al final de l siguiente linea se elimina "_viirs", al nombre del archivo
    # debido a que conabio cambio el nombre de archivo en el servidor FTP
    # el día 30-jun-2018
    archivo_salida = archivo_entrada.replace("/" + carpeta_fecha, "").replace(rutaInMODIS, rutaDestinoSistema).replace('/DIA',
"/fecha_hoy").replace("_viirs", "")
    print "archivo_entrada:", archivo_entrada
    print "archivo_salida:", archivo_salida
    if arcpy.Exists(archivo_salida):
        arcpy.Delete_management(archivo_salida)
    arcpy.CopyFeatures_management(archivo_entrada, archivo_salida)

    # 1 dia atras
    print "Actualizando 1 día atrás..."
    carpeta_fecha = carpeta_1_dia_atras.split("/")[-1]
    archivo_entrada = carpeta_1_dia_atras + "/" + nombreArchivoMODIS
    # Al final de l siguiente linea se elimina "_viirs", al nombre del archivo
    # debido a que conabio cambio el nombre de archivo en el servidor FTP
    # el día 30-jun-2018
    archivo_salida = archivo_entrada.replace("/" + carpeta_fecha, "").replace(rutaInMODIS, rutaDestinoSistema).replace('/DIA',
"/fecha_1_dia_atras").replace("_viirs", "")
    print "archivo_entrada:", archivo_entrada
    print "archivo_salida:", archivo_salida
    if arcpy.Exists(archivo_salida):
        arcpy.Delete_management(archivo_salida)
    arcpy.CopyFeatures_management(archivo_entrada, archivo_salida)

    # 2 dia atras
    print "Actualizando 2 días atrás..."
    carpeta_fecha = carpeta_2_dias_atras.split("/")[-1]
    archivo_entrada = carpeta_2_dias_atras + "/" + nombreArchivoMODIS
    archivo_salida = archivo_entrada.replace("/" + carpeta_fecha, "").replace(rutaInMODIS, rutaDestinoSistema).replace('/DIA',
"/fecha_2_dias_atras").replace("_viirs", "")
    print "archivo_entrada:", archivo_entrada
    print "archivo_salida:", archivo_salida
    if arcpy.Exists(archivo_salida):
        arcpy.Delete_management(archivo_salida)
    arcpy.CopyFeatures_management(archivo_entrada, archivo_salida)

    # 3 dia atras
    print "Actualizando 3 días atrás..."
    carpeta_fecha = carpeta_3_dias_atras.split("/")[-1]
    archivo_entrada = carpeta_3_dias_atras + "/" + nombreArchivoMODIS
    archivo_salida = archivo_entrada.replace("/" + carpeta_fecha, "").replace(rutaInMODIS, rutaDestinoSistema).replace('/DIA',
"/fecha_3_dias_atras").replace("_viirs", "")
    print "archivo_entrada:", archivo_entrada
    print "archivo_salida:", archivo_salida
    if arcpy.Exists(archivo_salida):
        arcpy.Delete_management(archivo_salida)
    arcpy.CopyFeatures_management(archivo_entrada, archivo_salida)

    # Actualizacion del historial de Archivos de puntos MODIS
    print "Actualizando historial de puntos MODIS..."
    carpetas = glob.glob(rutaInMODIS + "/*")
    for carpeta in carpetas:
        carpeta = carpeta.replace("\\", "/")
        carpeta_origen = carpeta.split("/")[-1]
        if not os.path.isdir(rutaOutMODIS + "/" + carpeta_origen):
            print "Actualizando: ", rutaOutMODIS + "/" + carpeta_origen
            shutil.copytree(carpeta, rutaOutMODIS + "/" + carpeta_origen)
            # renombra archivo copiado para quitarle "_viirs"
            # 10-Jul-2018
            # Cambio realizado ya que CONABIO cambio los nombres de los archivos
            # que deposita en el servidor FTP
            shp_antiguo = rutaOutMODIS + "/" + carpeta_origen + "/" + "puntoscalormodis_viirs.shp"
            shp_nuevo = shp_antiguo.replace("_viirs.shp", ".shp")
```

```
            if arcpy.Exists(shp_antiguo):
                arcpy.CopyFeatures_management(shp_antiguo, shp_nuevo)
                arcpy.Delete_management(shp_antiguo)

    #
    # Actualizacion de Archivos .bin LM100h
    #
    archivos = glob.glob(rutaInLM + "/*.*")
    for archivo in archivos:
        archivo = archivo.replace("\\", "/")
        nombre_archivo = archivo.split("/")[-1]
        if not os.path.isfile(rutaOutLM + "/" + nombre_archivo):
            print "Actualizando:", rutaOutLM + "/" + nombre_archivo
            shutil.copy(archivo, rutaOutLM + "/" + nombre_archivo)
    #
    # Actualizacion de Archivos NDVI
    #
    archivos = glob.glob(rutaInNDVI + "/composite_NDVI_*_merge.tif")
    for archivo in archivos:
        archivo = archivo.replace("\\", "/")
        nombre_archivo = archivo.split("/")[-1]
        if not os.path.isfile(rutaOutNDVI + "/" + nombre_archivo):
            print "Actualizando:", rutaOutNDVI + "/" + nombre_archivo
            shutil.copy(archivo, rutaOutNDVI + "/" + nombre_archivo)


# Termina actualizaArchivosLocalesDesdeFTP_fn
#########################################################################################################


#########################################################################################################
#
# _1_1_DPC
#
def _1_1_DPC_obtieneCountImagenUsoRegion(archivoTIFFusoRegion):
    print "DPC: ObtieneCountImagenUsoRegion..."
    dCountImageUsoRegion = {}
    rows = arcpy.SearchCursor(archivoTIFFusoRegion, "", "", "VALUE; COUNT", "VALUE")
    for row in rows:
        value = row.VALUE
        count = row.COUNT
        dCountImageUsoRegion[value] = count
    return dCountImageUsoRegion

def _1_1_DPC_obtieneCountDePuntosPorUsoRegion(archivoDePuntosDeCalorDiario, TIFFusoRegion):
    print "DPC: ObtieneCountDePuntosPorUsoRegion..."
    fechaArchivo = archivoDePuntosDeCalorDiario.split("/")[-2]
    anio = int(fechaArchivo.split("-")[0])

    # Revisa si el archivo de puntos tiene registros
    # Si no tiene registros devuelve un diccionario vacío
    rows = arcpy.SearchCursor(archivoDePuntosDeCalorDiario, "", "", "FID;", "")
    n = 0
    for row in rows:
        n += 1
    dCountImageUsoRegion = {}
    if n > 0:
        # Extrayendo puntos de calor
        borraColumnaShape(archivoDePuntosDeCalorDiario, "uso_region")
        arcpy.gp.ExtractMultiValuesToPoints_sa(archivoDePuntosDeCalorDiario, TIFFusoRegion, "NONE")
        # Lee puntos y los filtra por satélite
        # solo se deben considerar AQUA y TERRA, fuera los demás
        # A los 2011 a 2014 no filtrar por satélite, solo de 2015 a la fecha
        if anio >= 2015:
            condicion = " SATELITE = 'AQUA' OR SATELITE = 'TERRA' "
        else:
            condicion = ""
        rows = arcpy.SearchCursor(archivoDePuntosDeCalorDiario, condicion, "", "FID; USO_REGION", "USO_REGION")
        i = 0
        for row in rows:
            uso_region = row.USO_REGION
            # Se quitan usos AH y CA y -9999, no se usan
            # Las claves de AH son: 1=AH_____C, 8=AH_____NC, 15=AH_____NO, 21=AH____NBJ, 26=AH_____S, 36=AH_____NE
            # Las claves de CA son: 3=CA_____C, 9=CA_____NC, 14=CA_____NO, 20=CA____NBJ, 28=CA_____S, 34=CA_____NE
            lClavesAomitir = [1, 8, 15, 21, 26, 36, 3, 9, 14, 20, 28, 34, -9999]
            if not uso_region in lClavesAomitir:
                if uso_region in dCountImageUsoRegion:
                    dCountImageUsoRegion[uso_region] += 1
                else:
                    dCountImageUsoRegion[uso_region] = 1

                i += 1
    return dCountImageUsoRegion

def _1_1_DPC_combinaCounts(dCountImageUsoRegion, dCountPuntosPorUsoRegion, fechaArchivo, rutaSalidaDPC):
    print "DPC: combinaCounts..."
    # Crea archivo de salida
    archivo = open(rutaSalidaDPC + "/DPCobs_" + fechaArchivo + ".csv", 'w')
    # encabezados
    archivo.write("fecha,uso_region,count_pc,count_ur,DPC_obs\n")

    for uso_reg in dCountImageUsoRegion:
        if uso_reg in dCountPuntosPorUsoRegion:
            valor_pc = dCountPuntosPorUsoRegion[uso_reg]
            valor = round((dCountPuntosPorUsoRegion[uso_reg]*1.0) / (dCountImageUsoRegion[uso_reg]*1.0) * 200000, 1)
```

```
        else:
            valor_pc = 0
            valor = 0

        renglon = "%s,%s,%s,%s,%s\n" % (fechaArchivo, uso_reg, valor_pc, dCountImageUsoRegion[uso_reg], valor)
        archivo.write(renglon)
    archivo.close()

# Termina _1_1_DPC
#############################################################################################################

#############################################################################################################
#
# _1_2_FDI
#


def _1_2_FDI_prepara_carpeta_salida(anio, dir_salidas):
    print "FDI: _1_2_FDI_prepara_carpeta_salida..."
    # Elimina contenido de la carpeta actual de salida en forma recursiva
    # Corrobora que la ruta este completa antes de borrar recursivamente
    #if anio in dir_salidas:
    #    if os.path.exists(dir_salidas):
    #        print "Eliminando contenido de carpeta de salida..."
    #        shutil.rmtree(dir_salidas)
    # Crea estructura
    if not os.path.exists(dir_salidas):
        os.mkdir(dir_salidas)
    if not os.path.exists(dir_salidas + "/RH100"):
        os.mkdir(dir_salidas + "/RH100")
    if not os.path.exists(dir_salidas + "/RG"):
        os.mkdir(dir_salidas + "/RG")
    if not os.path.exists(dir_salidas + "/DR"):
        os.mkdir(dir_salidas + "/DR")
    if not os.path.exists(dir_salidas + "/FDI"):
        os.mkdir(dir_salidas + "/FDI")
    if not os.path.exists(dir_salidas + "/salidas"):
        os.mkdir(dir_salidas + "/salidas")

def _1_2_FDI_bin_a_tif(dicParams):
    print "FDI: _1_2_FDI_bin_a_tif..."
    #anio = dicParams["fechaHoyAnio"]
    rutaEntradaLM100bin = dicParams["rutaEntradaLM100bin"]
    rutaEntradaLM100tif = dicParams["rutaEntradaLM100tif"]
    archivoLM100binHDR = dicParams["archivoLM100binHDR"]

    cmd_gdal = dicParams["cmd_gdal"]
    opciones = ' -ot Float32 -b 1 '

    archivos=glob.glob(rutaEntradaLM100bin + "/*.bin")
    for archivoBIN in archivos:
        archivoBIN = archivoBIN.replace("\\", "/")
        archivoHDR = archivoBIN.replace(".bin", ".hdr")
        #si no existe el archivo de HDR, se lo creo
        if not os.path.isfile(archivoHDR):
            shutil.copy2(archivoLM100binHDR, archivoHDR)
        archivoTIFF = archivoBIN.replace(rutaEntradaLM100bin, rutaEntradaLM100tif).replace(".bin", ".tif")
        anio_tiff = archivoTIFF.split("/LM100h_")[1][0:4]
        #obtiene dia juliano para cambiarle el nombre, fecha en lugar de dia juliano
        dicDiasJulianos = creaDiccionario_diasJulianos_fechas(anio_tiff)
        l_ArchivoTIFF = archivoTIFF.split("/LM100h_%s" % anio_tiff)
        dia_juliano = l_ArchivoTIFF[1][0:3]
        fechaTIFF = dicDiasJulianos["dicDiaJuliano_fecha"][int(dia_juliano)].replace('-', '')
        archivoTIFF = "%s/LM100h_%s_Day.tif" % (l_ArchivoTIFF[0], fechaTIFF)
        if not os.path.isfile(archivoTIFF):
            if os.path.isfile(archivoBIN) and os.path.isfile(archivoHDR):
                print "convirtiendo:", archivoTIFF
                comando = "%s %s %s %s" % (cmd_gdal, opciones, archivoBIN, archivoTIFF)
                if os.path.isfile(cmd_gdal):
                    os.system(comando)
                else:
                    print "ERROR: No existe el archivo ejecutable de conversión: %s" % cmd_gdal
                    exit(0)

##### Inicia calculoRH ########

def _1_2_FDI_RH100_resample_1km(dicParam, fileLMtiff):
    print "FDI: _1_2_FDI_RH100_resample_1km..."
    entrada = fileLMtiff
    salida = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace(".tif", "_resample.tif")
    tif_base_para_resample = dicParam["FDI_H100_HMIN"]
    borra_archivo(salida)
    #arcpy.Resample_management(entrada, salida, tif_base_para_resample, "NEAREST")
    arcpy.Resample_management(entrada, salida, "8.99928006200001E-03", "NEAREST")
    #arcpy.Resample_management(entrada, salida, "0.0089992801 0.0089992801", "NEAREST")


def _1_2_FDI_RH100_mask_1(dicParam, fileLMtiff):
    print "FDI: _1_2_FDI_RH100_mask_1..."
    LM_Resample = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace(".tif", "_resample.tif")
    Mask_NDVI_1 = arcpy.Raster(dicParam["FDI_H100_mask_NDVI_1"])

    MASK_1 = LM_Resample * Mask_NDVI_1

    archivo_salida = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace(".tif", "_NDVI_Mask_1.tif")
```

```
    borra_archivo(archivo_salida)
    MASK_1.save(archivo_salida)


def _1_2_FDI_RH100_ecuacion_1(dicParam, fileLMtiff):
    print "FDI: _1_2_FDI_RH100_ecuacion_1..."
    # 100 - ("LM100h_20150620_Day.tif" - "HMIN.tif") / ("HMAX.tif" - "HMIN.tif") * 100
    LM = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace(".tif", "_NDVI_Mask_1.tif")
    HMIN = arcpy.Raster(dicParam["FDI_H100_HMIN"])
    HMAX = arcpy.Raster(dicParam["FDI_H100_HMAX"])

    Resta1 = LM - HMIN
    Resta2 = HMAX - HMIN
    Multiplica = Resta1 * 100
    Division = Multiplica / Resta2
    RH100 = 100 - Division

    # Almacena archivos temporales
    archivo_resta1 = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_", "/RH100_Resta1_")
    borra_archivo(archivo_resta1)
    Resta1.save(archivo_resta1)
    archivo_resta2 = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_", "/RH100_Resta2_")
    borra_archivo(archivo_resta2)
    Resta2.save(archivo_resta2)
    archivo_multiplica = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_",
"/RH100_Multiplica_")
    borra_archivo(archivo_multiplica)
    Multiplica.save(archivo_multiplica)
    archivo_division = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_",
"/RH100_Division_")
    borra_archivo(archivo_division)
    Division.save(archivo_division)
    # Almacena archivo final de resultados
    archivo_salida = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_", "/RH100_Raw_")
    borra_archivo(archivo_salida)
    RH100.save(archivo_salida)


def _1_2_FDI_RH100_reclass_RH100(dicParam, fileLMtiff):
    print "FDI: _1_2_FDI_RH100_reclass_RH100..."
    entrada = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_", "/RH100_Raw_")
    salida = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["rutaFDI_RH100"]).replace("/LM100h_", "/RH100_")
    borra_archivo(salida)
    arcpy.gp.Reclassify_sa(entrada, "Value",
                           "-9999999 0 0;1 1;2 2;3 3;4 4;5 5;6 6;7 7;8 8;9 9;10 10;11 11;12 12;13 13;14 14;15 15;16 16;17 17;18 18;19
19;20 20;21 21;22 22;23 23;24 24;25 25;26 26;27 27;28 28;29 29;30 30;31 31;32 32;33 33;34 34;35 35;36 36;37 37;38 38;39 39;40 40;41
41;42 42;43 43;44 44;45 45;46 46;47 47;48 48;49 49;50 50;51 51;52 52;53 53;54 54;55 55;56 56;57 57;58 58;59 59;60 60;61 61;62 62;63
63;64 64;65 65;66 66;67 67;68 68;69 69;70 70;71 71;72 72;73 73;74 74;75 75;76 76;77 77;78 78;79 79;80 80;81 81;82 82;83 83;84 84;85
85;86 86;87 87;88 88;89 89;90 90;91 91;92 92;93 93;94 94;95 95;96 96;97 97;98 98;99 99;100 9999999 100",
                           salida, "DATA")


def _1_2_FDI_calculoRH(dicParam, fileLMtiff):
    print "FDI: _1_2_FDI_calculoRH..."
    _1_2_FDI_RH100_resample_1km(dicParam, fileLMtiff)
    _1_2_FDI_RH100_mask_1(dicParam, fileLMtiff)
    _1_2_FDI_RH100_ecuacion_1(dicParam, fileLMtiff)
    _1_2_FDI_RH100_reclass_RH100(dicParam, fileLMtiff)

##### Termina calculoRH ########

##### Inicia calculo DR ########

def _1_2_FDI_DR_ecuacion_2(dicParam, fileLMtiff):
    print "FDI: _1_2_FDI_DR_ecuacion_2..."
    # Ajusta el año de la ruta de entrada de los NDVI segun la fechaNDVI
    #anioNDVI = fechaNDVI[0:4]
    #archivo_NDVI_entrada = dicParam["rutaEntradaNDVI"] + "/composite_NDVI_" + fechaNDVI + ".tif"
    archivosNDVI = glob.glob(dicParam["rutaEntradaNDVI"] + "/composite_NDVI_*_merge.tif")
    # Los ordena alfabeticamente que coincide cronológicamente
    archivosNDVI.sort()
    # Lee el último
    archivo_NDVI_entrada = archivosNDVI[-1]

    # Crea objetos raster
    NDVI = arcpy.Raster(archivo_NDVI_entrada)
    NDVI_Min = arcpy.Raster(dicParam["NDVImin"])
    NDVI_Diff = arcpy.Raster(dicParam["NDVIdiff"])
    # No se puede hacer la ecuacion en un solo paso, hay que hacerla por partes
    # Borra contenido de carpeta de salida
    # borra_contenido_carpeta(self.DR_dir_RG)
    # Realiza la ecuacion en tres pasos

    tmp_resta = NDVI - NDVI_Min
    tmp_multiplica = tmp_resta * 100
    tmp_resultado = tmp_multiplica / NDVI_Diff

    # Guarda los archivos temporales .tif
    archivo_resta = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_", "/RG_resta_")
    borra_archivo(archivo_resta)
    tmp_resta.save(archivo_resta)
    archivo_multiplica = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_",
"/RG_multiplica_")
    borra_archivo(archivo_multiplica)
    tmp_multiplica.save(archivo_multiplica)
```

```
    # Guarda archivo final de salida
    archivo_salida = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_", "/RG_raw_")
    borra_archivo(archivo_salida)
    tmp_resultado.save(archivo_salida)


def _1_2_DR_reclass_raster(dicParam, fileLMtiff):
    print "FDI: _1_2_DR_reclass_raster..."
    entrada = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_", "/RG_raw_")
    salida = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["rutaFDI_DR_RG"]).replace("/LM100h_", "/RG_")
    borra_archivo(salida)
    arcpy.gp.Reclassify_sa(entrada, "Value",
                           "-9999999 0 0;1 1;2 2;3 3;4 4;5 5;6 6;7 7;8 8;9 9;10 10;11 11;12 12;13 13;14 14;15 15;16 16;17 17;18 18;19
19;20 20;21 21;22 22;23 23;24 24;25 25;26 26;27 27;28 28;29 29;30 30;31 31;32 32;33 33;34 34;35 35;36 36;37 37;38 38;39 39;40 40;41
41;42 42;43 43;44 44;45 45;46 46;47 47;48 48;49 49;50 50;51 51;52 52;53 53;54 54;55 55;56 56;57 57;58 58;59 59;60 60;61 61;62 62;63
63;64 64;65 65;66 66;67 67;68 68;69 69;70 70;71 71;72 72;73 73;74 74;75 75;76 76;77 77;78 78;79 79;80 80;81 81;82 82;83 83;84 84;85
85;86 86;87 87;88 88;89 89;90 90;91 91;92 92;93 93;94 94;95 95;96 96;97 97;98 98;99 99;100 9999999 100",
                           salida, "DATA")


def _1_2_DR_ecuacion_3(dicParam, fileLMtiff):
    print "FDI: _1_2_DR_ecuacion_3..."
    # Crea objetos raster
    archivo_entrada = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["rutaFDI_DR_RG"]).replace("/LM100h_", "/RG_")
    RG_reclass = arcpy.Raster(archivo_entrada)
    LR_Max = arcpy.Raster(dicParam["DR_LR_max"])
    # No se puede hacer la ecuacion en un solo paso, hay que hacerla por partes
    # Realiza la ecuacion en tres pasos

    tmp_multiplica = RG_reclass * LR_Max
    tmp_divide = tmp_multiplica / 100
    tmp_resultado = 100 - tmp_divide

    # Guarda archivos temporales
    archivo_multiplica = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_",
"/DR_multiplica")
    borra_archivo(archivo_multiplica)
    tmp_multiplica.save(archivo_multiplica)
    archivo_divide = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["dirTMP"]).replace("/LM100h_", "/DR_divide")
    borra_archivo(archivo_divide)
    tmp_divide.save(archivo_divide)
    # Guarda archivo final de resultado
    archivo_resultado = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["rutaFDI_DR_DR"]).replace("/LM100h_", "/DR_")
    borra_archivo(archivo_resultado)
    tmp_resultado.save(archivo_resultado)


def _1_2_FDI_calculoDR(dicParam, fileLMtiff):
    print "FDI: _1_2_FDI_calculoDR..."
    _1_2_FDI_DR_ecuacion_2(dicParam, fileLMtiff)
    _1_2_DR_reclass_raster(dicParam, fileLMtiff)
    _1_2_DR_ecuacion_3(dicParam, fileLMtiff)

##### Termina calculo DR ########


##### Inicia calculo FDI ########

def _1_2_FDI_ecuacion_4(dicParam, fileLMtiff):
    print "FDI: _1_2_FDI_ecuacion_4..."
    # Crea objetos raster
    archivo_RH100 = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["rutaFDI_RH100"]).replace("/LM100h_", "/RH100_")
    archivo_DR = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["rutaFDI_DR_DR"]).replace("/LM100h_", "/DR_")
    RH100 = arcpy.Raster(archivo_RH100)
    DR = arcpy.Raster(archivo_DR)

    # Realiza ecuacion
    Multiplica = RH100 * DR
    FDI = Multiplica / 100

    archivo_FDI = fileLMtiff.replace(dicParam["rutaEntradaLM100tif"], dicParam["rutaFDI_FDI"]).replace("/LM100h_", "/FDI_")
    # Borra archivo de salida por si ya existe
    borra_archivo(archivo_FDI)
    # Guarda la salida en archivos .tif
    FDI.save(archivo_FDI)

##### Termina calculo FDI ########


# Termina _1_2_FDI
#############################################################################################################


#############################################################################################################
#
# _1_3_DEPC
#

##### Inicia calculo FDI_combine ########

def _1_3_DEPC_obtiene_dias_del_archivo_FDI_mean_csv(dicParam):
    print "DEPC: _1_3_DEPC_obtiene_dias_del_archivo_FDI_mean_csv..."
    archivo_entrada = dicParam["FDI_mean_csv"]
    f = open(archivo_entrada, 'r')
    lineas = f.readlines()
```

```
    i = 0
    dicFechas = []
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            fecha = l_linea[0]
            if not fecha in dicFechas:
                dicFechas.append(fecha)
        i += 1
    # Cierra archivo
    f.close()

    if 'f' in locals():
        del f
    if 'lineas' in locals():
        del lineas

    return dicFechas


def _1_3_DEPC_combina(dicParam, archivoTIFF):
    print "DEPC: _1_3_DEPC_combina %s..." % archivoTIFF
    # Crea cadena de fecha a partir del nombre del archivo LM
    tmp = archivoTIFF.split("/FDI_")
    tmp = tmp[1].split("_Day.tif")
    tmp = tmp[0]
    anio = tmp[0:4]
    fecha = tmp[6:8] + '/' + tmp[4:6] + '/' + anio

    # Si no existe la carpeta con el año para la salida de los combine, la crea
    #ruta_combine_salida = ruta_combines_salida + '/' + anio
    #if not os.path.isdir(ruta_combine_salida):
    #    os.mkdir(ruta_combine_salida)

    ruta_combine_salida = dicParam["DEPC_FDI_combines_tif"]
    tmp = archivoTIFF.replace('.tif', '_combine_usoreg.tif')
    ltmp = tmp.split('/')
    nombre_archivo_combine = ltmp[-1]
    img_combine_salida = ruta_combine_salida + '/' + nombre_archivo_combine
    #if arcpy.Exists(img_combine_salida):
    #    arcpy.Delete_management(img_combine_salida)
    # Si ya existe ya no la genera
    if not arcpy.Exists(img_combine_salida):
        arcpy.gp.Combine_sa(dicParam["TIFFusoRegion"] + ";" + archivoTIFF, img_combine_salida)


    # Lee tabla de datos de la imagen combine
    #rows = arcpy.SearchCursor(img_combine_salida, "", "", "OID; COUNT,LM10H_" + anio + ",UOSREGWP17", "UOSREGWP17,LM10H_" + anio)
    # Los primeros 10 caracteres del nombre del tiff son los que le pone a la columna
    ltmnombreArchivoTIFF = archivoTIFF.split('/')[-1]
    nombreColTIFF = ltmnombreArchivoTIFF[0:10]
    rows = arcpy.SearchCursor(img_combine_salida, "", "", "OID; COUNT,"+nombreColTIFF+",USO_REGION", "USO_REGION,"+nombreColTIFF)
    i = 0
    dicSumasUsoReg = {}
    dicSumaCount = {}
    listaLlaves = []
    for row in rows:
        i += 1

        #ren = "%s,%s,%s\n" % (row.COUNT, row.UOSREGWP17, row.LM10H_2012)
        #csv_combine_iteraciones.write(ren)

        if row.USO_REGION in dicSumasUsoReg:
            dicSumasUsoReg[row.USO_REGION] += row.COUNT * eval("row."+nombreColTIFF)
            dicSumaCount[row.USO_REGION] += row.COUNT
        else:
            listaLlaves.append(row.USO_REGION)
            dicSumasUsoReg[row.USO_REGION] = row.COUNT * eval("row."+nombreColTIFF)
            dicSumaCount[row.USO_REGION] = row.COUNT

    #  Agrega datos de este tiff al archivo CSV
    FDI_mean_csv = open(dicParam["FDI_mean_csv"], 'a')

    # ordena los datos antes de mandarlos al archivo de salida
    for llave in listaLlaves:
        # uso_reg,Suma(count * LM10H_2012), contador de registros con el mismo uso_reg, promedio_ponderado
        ren = "%s,%s,%s,%s,%s\n" % (fecha, llave, dicSumasUsoReg[llave], dicSumaCount[llave],
dicSumasUsoReg[llave]/dicSumaCount[llave])
        FDI_mean_csv.write(ren)

    # Cierra archivo
    FDI_mean_csv.close()


def _1_3_1_DEPC_FDI_combine(dicParam):
    print "DEPC: _1_3_1_DEPC_FDI_combine..."
    # asigna nombre al archivo de salida
    csv_salida_combine_suma_ponderada = dicParam["FDI_mean_csv"]


    # Si ya existe el archivo de salida lo borra
    #if os.path.isfile(csv_salida_combine_suma_ponderada):
    #    os.remove(csv_salida_combine_suma_ponderada)
```

```
    if os.path.isfile(csv_salida_combine_suma_ponderada):
        # Con el propósito de generar únicamente los dias que no estan en el archivo FDI_csv,
        # Se generará un diccionario con los dias ya incluidos en este archivo y
        # solo se generarán los dias que no estén en ese archivo
        # Obtiene la lista de fechas en formato dd/mm/aaa
        dicDiasFDImeanCSV = _1_3_DEPC_obtiene_dias_del_archivo_FDI_mean_csv(dicParam)
    else:
        # crea archivos CSV
        # csv_combine_iteraciones = open(csv_salida_combine_iteraciones,'w')
        FDI_mean_csv = open(csv_salida_combine_suma_ponderada, 'w')
        # abre archivos
        # csv_combine_iteraciones = open(csv_salida_combine_iteraciones, 'a')
        FDI_mean_csv = open(csv_salida_combine_suma_ponderada, 'a')
        # pone encabezados
        # FDImean = suma(count * FDI) / suma(count)
        ren = "fecha,uso_region,suma(count * FDI),suma(count),FDImean\n"
        FDI_mean_csv.write(ren)
        # Cierra archivo
        FDI_mean_csv.close()

        dicDiasFDImeanCSV = []

    ruta_entrada_TIFF = dicParam["rutaFDI_FDI"]
    archivosTIFF = glob.glob(ruta_entrada_TIFF + "/*.tif")
    for archivoTIFF in archivosTIFF:
        archivoTIFF = archivoTIFF.replace("\\", "/")
        # FDI_20170111_Day.tif
        fechaArchivoTIFF = archivoTIFF.split("/")[-1].split("_")[1]
        fechaArchivoTIFF = fechaArchivoTIFF[6:] + "/" + fechaArchivoTIFF[4:6] + "/" + fechaArchivoTIFF[0:4]
        # Si la fecha de este archivo TIFF ya está en el archivo FDI_meas_csv yo ho hace el combine
        # solo se agregan a este archivo los registros generados del combine de los TIFF nuevos
        if not fechaArchivoTIFF in dicDiasFDImeanCSV:
            # proyecta(archivoTIFF)
            _1_3_DEPC_combina(dicParam, archivoTIFF)


##### Termina calculo FDI_combine ########



##### Inicia calculo FDI_acumulado ########

def _1_3_2_DEPC_FDI_acumulado__archivos_CSV_a_tabla(dicParam, d_tabla):
    print "DEPC: _1_3_2_DEPC_FDI_acumulado__archivos_CSV_a_tabla..."
    l_dias = []
    archivo_entrada = dicParam["FDI_mean_csv"]
    f = open(archivo_entrada, 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            fecha = l_linea[0]
            l_fecha = fecha.split("/")
            fecha = l_fecha[2] + "-" + l_fecha[1] + "-" + l_fecha[0]
            uso_region = int(l_linea[1])
            # Filtra solo los uso_reg útiles
            if not uso_region in dicParam["l_uso_reg_a_omitir"]:
                fdimean = int(round(float(l_linea[4]), 0))

                if not fecha in l_dias:
                    l_dias.append(fecha)
                    d_tabla[fecha] = {}
                d_tabla[fecha][uso_region] = fdimean
        i += 1
    # Cierra archivo
    f.close()

    if 'l_dias' in locals():
        del l_dias
    if 'f' in locals():
        del f
    if 'lineas' in locals():
        del lineas

    return d_tabla


def _1_3_2_DEPC_FDI_acumulado__tabla_menos_60(dicParam, d_tabla):
    print "DEPC: _1_3_2_DEPC_FDI_acumulado__tabla_menos_60..."
    for fecha in d_tabla:
        for uso_reg in d_tabla[fecha]:
            # Se da tratamiento diferente a la zona sur
            if uso_reg in dicParam["l_dias_uso_region_sur"]:
                cantidad_a_restar = 50
            else:
                cantidad_a_restar = 60

            if d_tabla[fecha][uso_reg] > cantidad_a_restar:
                d_tabla[fecha][uso_reg] = d_tabla[fecha][uso_reg] - cantidad_a_restar
            else:
                d_tabla[fecha][uso_reg] = 0
    return d_tabla
```

```
#
# Acumula el valor de uso_region de 90 dias atrás a la fecha indicada
#
def acumulado(fecha_final, uso_reg, d_tabla):
    #print "fecha_final:", fecha_final
    fecha_tmp = datetime.datetime(int(fecha_final[0:4]), int(fecha_final[5:7]), int(fecha_final[8:10]))
    dias = timedelta(days=90)
    fecha_inicial = fecha_tmp - dias
    fecha_inicial_txt = ("%s" % fecha_inicial)[0:10]
    fecha_final_txt = "%s" % fecha_final
    #print "fecha_inicial_txt:", fecha_inicial_txt
    #print "fecha_final_txt:", fecha_final_txt

    suma = 0
    for fecha in d_tabla:
        if fecha >= fecha_inicial_txt and fecha <= fecha_final_txt:
            suma += d_tabla[fecha][uso_reg]

    return suma
#
# Genera tabla con el aculumado de 90 dias
#
def _1_3_2_DEPC_FDI_acumulado__acumulado_90_dias(d_tabla, d_acumulado):
    print "DEPC: _1_3_2_DEPC_FDI_acumulado__acumulado_90_dias..."
    l_dias = []
    #fecha_90_dias = "2011-04-01"

    # Obtiene la fecha de los últimos 90 dias anteriores disponibles
    # Obtiene las fechas de la tabla
    l_fechas = d_tabla.keys()
    # Ordena las fechas cronologicamente
    l_fechas.sort()
    if len(l_fechas) >= 90:
        # Obtiene la fecha de la lectura 90 hacia atrás
        fecha_90_dias = l_fechas[-90]
    else:
        # Obtiene la primera fecha disponible
        fecha_90_dias = l_fechas[0]

    for fecha in d_tabla:
        for uso_reg in d_tabla[fecha]:
            if fecha >= fecha_90_dias:
                valor = acumulado(fecha, uso_reg, d_tabla)
                if not fecha in l_dias:
                    l_dias.append(fecha)
                    d_acumulado[fecha] = {}
                d_acumulado[fecha][uso_reg] = round((valor / 10), 0)

    if 'l_fechas' in locals():
        del l_fechas

    return d_acumulado



def _1_3_2_DEPC_FDI_acumulado__aplica_ecuaciones(dicParam, d_acumulado):
    print "DEPC: _1_3_2_DEPC_FDI_acumulado__aplica_ecuaciones..."
    # Carga archivo de parámetros a diccionario "d_parametros"
    f = open(dicParam["DEPC_archivo_parametros"], 'r')
    lineas = f.readlines()
    i = 0
    d_parametros = {}
    d_acumulado_salida = {}
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            uso_reg = int(l_linea[0])
            param_a = float(l_linea[1])
            param_b = float(l_linea[2])
            d_parametros[uso_reg] = {'a': param_a, 'b': param_b}
        i += 1
    # Aplica ecuacion a valores
    for fecha in d_acumulado:
        d_acumulado_fecha = {}
        for uso_reg in d_acumulado[fecha]:
            #d_acumulado[fecha][uso_reg] = round(d_parametros[uso_reg]['a'] * (d_acumulado[fecha][uso_reg] / 100) **
d_parametros[uso_reg]['b'], 0)
            #d_acumulado_salida[fecha][uso_reg]['FDIac'] = d_acumulado[fecha][uso_reg]
            #d_acumulado_salida[fecha][uso_reg]['a'] = d_parametros[uso_reg]['a']
            #d_acumulado_salida[fecha][uso_reg]['b'] = d_parametros[uso_reg]['b']
            #d_acumulado_salida[fecha][uso_reg]['DEPC_FDIac'] = round(d_parametros[uso_reg]['a'] * (d_acumulado[fecha][uso_reg] /
100) ** d_parametros[uso_reg]['b'], 0)
            FDIac = d_acumulado[fecha][uso_reg]
            a = d_parametros[uso_reg]['a']
            b = d_parametros[uso_reg]['b']
            DEPC_FDIac = round(d_parametros[uso_reg]['a'] * (d_acumulado[fecha][uso_reg] / 100) ** d_parametros[uso_reg]['b'], 0)
            d_acumulado_fecha[uso_reg] = {'FDIac': FDIac, 'a': a, 'b': b, 'DEPC_FDIac': DEPC_FDIac}
        d_acumulado_salida[fecha] = d_acumulado_fecha

    return d_acumulado_salida


def _1_3_2_DEPC_FDI_acumulado__genera_CSV_salida(dicParam, d_acumulado):
    print "DEPC: _1_3_2_DEPC_FDI_acumulado__genera_CSV_salida..."
```

```
    # ordena fechas
    l_fechas = d_acumulado.keys()
    l_fechas.sort()

    # recorre tabla de acuerdo al orden de la lista de fechas
    tmpFecha = ''
    for fecha_orden in l_fechas:

        archivo_nuevo = 0
        if fecha_orden != tmpFecha:
            if tmpFecha != '':
                # Cierra el ultimo archivo creado
                archivo.close()
            tmpFecha = fecha_orden
            archivo_nuevo = 1

        if archivo_nuevo == 1:
            # Genera un archivo de salida por día
            archivo_diario = dicParam["DEPC_FDIac_csv"].replace("FechA", fecha_orden)
            archivo = open(archivo_diario, 'w')
            # encabezados
            archivo.write("fecha,uso_region,FDIac,a,b,DEPC_FDIac\n")

        row_dia = d_acumulado[fecha_orden]
        # ordena claves de uso_region
        l_orden_uso_region = row_dia.keys()
        l_orden_uso_region.sort()
        # agrega registro a archivo CSV
        for uso_region_orden in l_orden_uso_region:
            registro = "%s,%s,%s,%s,%s,%s\n" % (fecha_orden, uso_region_orden, row_dia[uso_region_orden]['FDIac'],
row_dia[uso_region_orden]['a'],row_dia[uso_region_orden]['b'],row_dia[uso_region_orden]['DEPC_FDIac'])
            archivo.write(registro)

    # Cierra el ultimo archivo creado
    archivo.close()



def _1_3_2_DEPC_FDI_acumulado(dicParam):
    print "DEPC: _1_3_2_DEPC_FDI_acumulado..."
    d_tabla = {}
    #l_uso_reg_a_omitir = [1, 3, 8, 9, 14, 15, 20, 21, 34, 36, 26, 28]
    #l_dias_uso_region_sur = [25, 27, 28, 29, 30, 31, 32, 33]
    d_acumulado = {}

    d_tabla = _1_3_2_DEPC_FDI_acumulado__archivos_CSV_a_tabla(dicParam, d_tabla)
    #print "dtabla 1:", d_tabla
    d_tabla = _1_3_2_DEPC_FDI_acumulado__tabla_menos_60(dicParam, d_tabla)
    #print "dtabla_2:", d_tabla
    d_acumulado = _1_3_2_DEPC_FDI_acumulado__acumulado_90_dias(d_tabla, d_acumulado)
    #print "d_acumulado1:", d_acumulado
    ###_1_3_2_DEPC_FDI_acumulado__genera_CSV_salida(d_acumulado, archivo_salida_1)
    d_acumulado = _1_3_2_DEPC_FDI_acumulado__aplica_ecuaciones(dicParam, d_acumulado)
    #print "d_acumulado2:", d_acumulado
    _1_3_2_DEPC_FDI_acumulado__genera_CSV_salida(dicParam, d_acumulado)
    #exit(0)

    if 'd_tabla' in locals():
        del d_tabla
    if 'd_acumulado' in locals():
        del d_acumulado


##### Termina calculo FDI_acumulado ########



##### Inicia DEPC 3 dias ########
"""
def _1_3_3_DEPC_obtieneCountImagenUsoRegion(dicParam):
    print "DEPC: _1_3_3_DEPC_obtieneCountImagenUsoRegion..."
    dCountImageUsoRegion = {}
    rows = arcpy.SearchCursor(dicParam["TIFFusoRegion"], "", "", "VALUE; COUNT", "VALUE")
    for row in rows:
        value = row.VALUE
        count = row.COUNT
        dCountImageUsoRegion[value] = count
    return dCountImageUsoRegion


def _1_3_3_DEPC_obtieneCountDePuntosPorUsoRegion(dicParam, archivoDePuntosDeCalorDiario):
    print "DEPC: _1_3_3_obtieneCountDePuntosPorUsoRegion..."
    fechaArchivo = archivoDePuntosDeCalorDiario.split('/')[-2]
    anio = int(fechaArchivo.split("-")[0])

    # Extrayendo puntos de calor
    borraColumnaShape(archivoDePuntosDeCalorDiario, "uso_region")

    # Si el archivo de puntos no tiene registros marca error, entonces, esto solo se hace si tiene registros
    # Si no tiene registros, solo se le crea la columna "uso_region" a la tabla vacía de puntos
    rows = arcpy.SearchCursor(archivoDePuntosDeCalorDiario, "", "", "FID;", "")
    n = 0
    for row in rows:
        n += 1
    if n > 0:
```

```
        arcpy.gp.ExtractMultiValuesToPoints_sa(archivoDePuntosDeCalorDiario, dicParam["TIFFusoRegion"] + " uso_region", "NONE")
    else:
        creaColumnaUsoRegionEnShape(archivoDePuntosDeCalorDiario)

    # Lee puntos y los filtra por satélite
    # solo se deben considerar AQUA y TERRA, fuera los demás
    # A los 2011 a 2014 no filtrar por satélite, solo de 2015 a la fecha
    if anio >= 2015:
        condicion = " SATELITE = 'AQUA' OR SATELITE = 'TERRA' "
    else:
        condicion = ""
    rows = arcpy.SearchCursor(archivoDePuntosDeCalorDiario, condicion, "", "FID; FECHA; USO_REGION", "USO_REGION")
    i = 0
    dCountImageUsoRegion = {}
    for row in rows:
        # FID = row.FID
        # acq_date = row.ACQ_DATE
        uso_region = row.USO_REGION
        # Se quitan usos AH y CA y -9999, no se usan
        # Las claves de AH son: 1=AH_____C, 8=AH_____NC, 15=AH_____NO, 21=AH____NBJ, 26=AH_____S, 36=AH_____NE
        # Las claves de CA son: 3=CA_____C, 9=CA_____NC, 14=CA_____NO, 20=CA____NBJ, 28=CA_____S, 34=CA_____NE
        lClavesAomitir = [1, 8, 15, 21, 26, 36, 3, 9, 14, 20, 28, 34, -9999]
        if not uso_region in lClavesAomitir:
            if uso_region in dCountImageUsoRegion:
                dCountImageUsoRegion[uso_region] += 1
            else:
                dCountImageUsoRegion[uso_region] = 1

            i += 1
    return dCountImageUsoRegion


def _1_3_3_DEPC_combinaCounts(dicParam, dCountImageUsoRegion, dCountPuntosPorUsoRegion, archivo_salida, fechaArchivo):
    # Crea archivo de salida
    archivo = open(archivo_salida, 'w')
    # encabezados
    archivo.write("fecha,uso_region,count_pc,count_ur,DPC_obs\n")

    for uso_reg in dCountImageUsoRegion:
        if uso_reg in dCountPuntosPorUsoRegion:
            valor_pc = dCountPuntosPorUsoRegion[uso_reg]
            valor = round((dCountPuntosPorUsoRegion[uso_reg]*1.0) / (dCountImageUsoRegion[uso_reg]*1.0) * 200000, 1)
        else:
            valor_pc = 0
            valor = 0

        renglon = "%s,%s,%s,%s,%s\n" % (fechaArchivo, uso_reg, valor_pc, dCountImageUsoRegion[uso_reg], valor)
        archivo.write(renglon)
    archivo.close()



def _1_3_3_DEPC(dicParam):
    print "DEPC: _1_3_3_DEPC..."
    carpetasArchivosDePuntosDeCalorDiarios = glob.glob(dicParam["rutaDEPC_puntos_de_calor_diarios"] + "/*")
    for carpetaArchivosDePuntosDeCalorDiarios in carpetasArchivosDePuntosDeCalorDiarios:
        carpetaArchivosDePuntosDeCalorDiarios = carpetaArchivosDePuntosDeCalorDiarios.replace("\\", "/")
        archivoDePuntosDeCalorDiario = carpetaArchivosDePuntosDeCalorDiarios + "/" + dicParam["nombreDEPC_puntos_de_calor_diarios"]
        if os.path.isfile(archivoDePuntosDeCalorDiario):

            fechaArchivo = archivoDePuntosDeCalorDiario.split('/')[-2]
            archivo_salida = dicParam["rutaDEPC_salidas"] + "/DEPC_" + fechaArchivo + ".csv"
            if not os.path.isfile(archivo_salida):
                dCountImageUsoRegion = _1_3_3_DEPC_obtieneCountImagenUsoRegion(dicParam)
                dCountPuntosPorUsoRegion = _1_3_3_DEPC_obtieneCountDePuntosPorUsoRegion(dicParam, archivoDePuntosDeCalorDiario)
                _1_3_3_DEPC_combinaCounts(dicParam, dCountImageUsoRegion, dCountPuntosPorUsoRegion, archivo_salida, fechaArchivo)
"""

##### Termina DEPC 3 dias ########


##### Inicia DEPC ########

def _1_3_3_DEPC_cargaTablaDeConstante_c(dicParam):
    print "DEPC: _1_3_3_DEPC_cargaTablaDeConstante_c..."
    dic_tabla_c = {}
    f = open(dicParam["DEPC_contante_c"], 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            uso_reg = int(l_linea[0])
            valor = float(l_linea[1])
            dic_tabla_c[uso_reg] = valor
        i += 1
    # Cierra archivo
    f.close()
    return dic_tabla_c


#
# calcula promedio de los tres dias anteriores de DPCObs
#
```

```
def X_1_3_3_DEPC_calculaPromedio3diasAnterioresDPCobs(dicParam, fecha):
    print "DEPC: _1_3_3_DEPC_calculaPromedio3diasAnterioresDPCobs..."
    # Para obtener los tres días anteiores se hará lo siguiente:
    # - Se leeran todos los archivos CSV de la carpeta DPCobs
    # - La fecha que está en el nombre del archivo se mandará a una lista
    # - Se ordenará la lista de nombres en orden descendente
    # - Se obtendrán las fechas de los tres archivos con fecha menor a la actual
    # - Se revisará que la fecha mas antigua no sea mayor a 13 días, de lo contrario mandará una alerta
    # - Se lee el valor de uso_region de los archivos de los tres dias anteriores
    # - Se retorna el valor promedio de los tres días
    #
    #print "--->Fecha obs:", fecha
    # - Se leeran todos los archivos CSV de la carpeta DPCobs
    # recorre carpeta de archivos DCPobs disponibles
    lFechaArchivos = []
    archivos = glob.glob(dicParam["rutaSalidaDPC"] + "/*.csv")

    for archivo in archivos:
        archivo = archivo.replace("\\", "/")
        fechaArchivo = archivo.split("/")[-1].split("_")[1].split(".")[0]
        fechaArchivo = fechaArchivo[0:4] + "-" + fechaArchivo[4:6] + "-" + fechaArchivo[6:]

        # print "fecha_archivo:", fechaArchivo
        # - La fecha que está en el nombre del archivo se mandará a una lista
        lFechaArchivos.append(fechaArchivo)
    # - Se ordenará la lista de nombres en orden descendente
    lFechaArchivos.sort(reverse=True)

    # - Se obtendrán las fechas de los tres archivos con fecha menor a la actual
    i = 0
    lFechasValidas3dias = []
    for fechaArchivo in lFechaArchivos:
        if fechaArchivo < fecha:
            lFechasValidas3dias.append(fechaArchivo)
            i += 1
            if i >= 3:
                break

    # - Se revisará que la fecha mas antigua no sea mayor a 13 días (10 + 3), de lo contrario mandará una alerta
    fechaUltimoDiaValido = lFechasValidas3dias[-1]
    lFecha = fecha.split("-")
    fecha_tmp = datetime.datetime(int(lFecha[0]), int(lFecha[1]), int(lFecha[2]))
    dias = timedelta(days=13)
    fechaValidaMasAntigua = ("%s" % (fecha_tmp - dias))[0:10]
    if fechaUltimoDiaValido < fechaValidaMasAntigua:
        print "ATENCION: La fecha de los archivos de DPCobs es mas antigua de 10 días !"

    # - Se lee el promedio del valor de uso_region de los archivos de los tres dias anteriores
    dicAcumulado = {}
    lFechasValidas3dias.sort()

    for lFechaValidas3dias in lFechasValidas3dias:
        #archivoEntradaDPCobs = dicParam["rutaDEPC_FDIac_csv"] + "/DEPC_FechA.csv"
        archivoEntradaDPCobs = dicParam["rutaDEPC_FDIac_csv"] + "/DEPC_FDIac_FechA.csv"
        nombreArchivo = archivoEntradaDPCobs.replace("FechA", lFechaValidas3dias)
        print "nombreArchivo:", nombreArchivo
        f = open(nombreArchivo, 'r')
        lineas = f.readlines()
        i = 0
        for linea in lineas:
            # Ignora la primer linea porque es de encabezados
            if i > 0:
                l_linea = linea.split(",")
                tmp_fecha = l_linea[0]
                tmp_uso_region = int(float(l_linea[1]))
                tmp_valor = float(l_linea[5])
                dicDia = {tmp_uso_region: tmp_valor}
                if tmp_fecha in dicAcumulado:
                    if tmp_uso_region in dicAcumulado[tmp_fecha]:
                        dicAcumulado[tmp_fecha][tmp_uso_region] += tmp_valor
                    else:
                        dicAcumulado[tmp_fecha][tmp_uso_region] = tmp_valor
                else:
                    dicAcumulado[tmp_fecha] = dicDia
            i += 1
        # Cierra archivo
        f.close()

    dicUsoRegPromedio = {}
    totDias = len(dicAcumulado) * 1.0
    for fechaAcumulado in dicAcumulado:
        for ur in dicAcumulado[fechaAcumulado]:
            if ur in dicUsoRegPromedio:
                if dicAcumulado[fechaAcumulado][ur] > 0:
                    dicUsoRegPromedio[ur] += dicAcumulado[fechaAcumulado][ur]
            else:
                if dicAcumulado[fechaAcumulado][ur] > 0:
                    dicUsoRegPromedio[ur] = dicAcumulado[fechaAcumulado][ur]
                else:
                    dicUsoRegPromedio[ur] = 0

    for ur in dicUsoRegPromedio:
        dicUsoRegPromedio[ur] = dicUsoRegPromedio[ur] / totDias
```

```
    # Redondea
    for ur in dicUsoRegPromedio:
        dicUsoRegPromedio[ur] = round(dicUsoRegPromedio[ur], 2)
    return dicUsoRegPromedio


#
# calcula promedio de los tres dias anteriores de DPCObs
#
def _1_3_3_DEPC_calculaPromedio3diasAnterioresDPCobs(dicParam, fecha):
    print "DEPC: _1_3_3_DEPC_calculaPromedio3diasAnterioresDPCobs..."
    # Para obtener los tres días anteiores se hará lo siguiente:
    # - Se leeran todos los archivos CSV de la carpeta DPCobs
    # - La fecha que está en el nombre del archivo se mandará a una lista
    # - Se ordenará la lista de nombres en orden descendente
    # - Se obtendrán las fechas de los tres archivos con fecha menor a la actual
    # - Se revisará que la fecha mas antigua no sea mayor a 13 días, de lo contrario mandará una alerta
    # - Se lee el valor de uso_region de los archivos de los tres dias anteriores
    # - Se retorna el valor promedio de los tres dias
    #
    #print "--->Fecha obs:", fecha
    # - Se leeran todos los archivos CSV de la carpeta DPCobs
    # recorre carpeta de archivos DCPobs disponibles
    lFechaArchivos = []
    archivos = glob.glob(dicParam["rutaSalidaDPC"] + "/*.csv")
    archivos.sort()
    lista_ultimos_3_dias = archivos[-3:]

    """
    print "archivos:", archivos
    print "lista_ultimos_3_dias:", lista_ultimos_3_dias
    exit(0)

    for archivo in archivos:
        archivo = archivo.replace("\\", "/")
        fechaArchivo = archivo.split("/")[-1].split("_")[1].split(".")[0]
        fechaArchivo = fechaArchivo[0:4] + "-" + fechaArchivo[4:6] + "-" + fechaArchivo[6:]

        # print "fecha_archivo:", fechaArchivo
        # - La fecha que está en el nombre del archivo se mandará a una lista
        lFechaArchivos.append(fechaArchivo)
    # - Se ordenará la lista de nombres en orden descendente
    lFechaArchivos.sort(reverse=True)

    # - Se obtendrán las fechas de los tres archivos con fecha menor a la actual
    i = 0
    lFechasValidas3dias = []
    for fechaArchivo in lFechaArchivos:
        if fechaArchivo < fecha:
            lFechasValidas3dias.append(fechaArchivo)
            i += 1
            if i >= 3:
                break

    # - Se revisará que la fecha mas antigua no sea mayor a 13 días (10 + 3), de lo contrario mandará una alerta
    fechaUltimoDiaValido = lFechasValidas3dias[-1]
    lFecha = fecha.split("-")
    fecha_tmp = datetime.datetime(int(lFecha[0]), int(lFecha[1]), int(lFecha[2]))
    dias = timedelta(days=13)
    fechaValidaMasAntigua = ("%s" % (fecha_tmp - dias))[0:10]
    if fechaUltimoDiaValido < fechaValidaMasAntigua:
        print "ATENCION: La fecha de los archivos de DPCobs es mas antigua de 10 días !"

    # - Se lee el promedio del valor de uso_region de los archivos de los tres dias anteriores
    dicAcumulado = {}
    lFechasValidas3dias.sort()
    """

    dicAcumulado = {}
    for archivo_dia in lista_ultimos_3_dias:
        archivo_dia = archivo_dia.replace("\\", "/")
        #fechaArchivo = archivo_dia.split("/")[-1].split("_")[1].split(".")[0]
        #fechaArchivo = fechaArchivo[0:4] + "-" + fechaArchivo[4:6] + "-" + fechaArchivo[6:]

        #archivoEntradaDPCobs = dicParam["rutaDEPC_FDIac_csv"] + "/DEPC_FechA.csv"
        #archivoEntradaDPCobs = dicParam["rutaDEPC_FDIac_csv"] + "/DEPC_FDIac_FechA.csv"
        #nombreArchivo = archivoEntradaDPCobs.replace("FechA", lFechaValidas3dias)
        #print "nombreArchivo:", archivo_dia
        f = open(archivo_dia, 'r')
        lineas = f.readlines()
        i = 0
        for linea in lineas:
            # Ignora la primer linea porque es de encabezados
            if i > 0:
                l_linea = linea.split(",")
                tmp_fecha = l_linea[0]
                tmp_uso_region = int(float(l_linea[1]))
                tmp_valor = float(l_linea[4])
                dicDia = {tmp_uso_region: tmp_valor}
                if tmp_fecha in dicAcumulado:
                    if tmp_uso_region in dicAcumulado[tmp_fecha]:
                        dicAcumulado[tmp_fecha][tmp_uso_region] += tmp_valor
                    else:
                        dicAcumulado[tmp_fecha][tmp_uso_region] = tmp_valor
                else:
```

```
                    dicAcumulado[tmp_fecha] = dicDia
            i += 1
        # Cierra archivo
        f.close()

    dicUsoRegPromedio = {}
    #totDias = len(dicAcumulado) * 1.0
    totDias = 3.0
    for fechaAcumulado in dicAcumulado:
        for ur in dicAcumulado[fechaAcumulado]:
            if ur in dicUsoRegPromedio:
                if dicAcumulado[fechaAcumulado][ur] > 0:
                    dicUsoRegPromedio[ur] += dicAcumulado[fechaAcumulado][ur]
            else:
                if dicAcumulado[fechaAcumulado][ur] > 0:
                    dicUsoRegPromedio[ur] = dicAcumulado[fechaAcumulado][ur]
                else:
                    dicUsoRegPromedio[ur] = 0.0

    for ur in dicUsoRegPromedio:
        # divide y redondea a 1 decimal
        dicUsoRegPromedio[ur] = float("%.1f" % (dicUsoRegPromedio[ur] / totDias))

    return dicUsoRegPromedio


def _1_3_3_DEPC_cargaArchivoCSV_DEPC_FDIac(archivo):
    print "DEPC: _1_3_3_DEPC_cargaArchivoCSV_DEPC_FDIac..."
    dic_archivo = {}
    f = open(archivo, 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            fecha = l_linea[0]
            uso_region = int(l_linea[1])
            valor = float(l_linea[5])
            dic_archivo[uso_region] = [fecha, valor]
        i += 1
    # Cierra archivo
    f.close()
    return dic_archivo


#
# Agrega columnas de DPCpred a USOREG para hacer el merge de datos
#
def _1_3_3_DEPC_agregaColumnasAusoreg(archivoTiffUsoRegion):
    print "DEPC: _1_3_3_DEPC_agregaColumnasAusoreg..."
    # fecha
    try:
        arcpy.DeleteField_management(archivoTiffUsoRegion, "fecha")
    except:
        pass
    arcpy.AddField_management(archivoTiffUsoRegion, "fecha", "TEXT", "", "", "10", "", "NULLABLE", "NON_REQUIRED", "")
    # DPC1
    try:
        arcpy.DeleteField_management(archivoTiffUsoRegion, "DEPC_FDIac")
    except:
        pass
    arcpy.AddField_management(archivoTiffUsoRegion, "DEPC_FDIac", "SHORT", "3", "", "", "", "NULLABLE", "NON_REQUIRED", "")
    # DPCobs_da
    try:
        arcpy.DeleteField_management(archivoTiffUsoRegion, "DPCobs_da")
    except:
        pass
    arcpy.AddField_management(archivoTiffUsoRegion, "DPCobs_da", "DOUBLE", "5", "1", "", "", "NULLABLE", "NON_REQUIRED", "")
    # DPCpred
    try:
        arcpy.DeleteField_management(archivoTiffUsoRegion, "DPCpred")
    except:
        pass
    arcpy.AddField_management(archivoTiffUsoRegion, "DPCpred", "DOUBLE", "10", "6", "", "", "NULLABLE", "NON_REQUIRED", "")



#
# Combina DPC1 y DPCobs
#
def _1_3_3_DEPC_DEPCFDIac_DPCobs(dicParam, dic_tabla_c):
    print "DEPC: _1_3_3_DEPC_DEPCFDIac_DPCobs..."
    #archivos = glob.glob(dicParam["rutaDEPC_salida_acumulado_con_ecuacion"] + "/*.csv")

    archivos = glob.glob(dicParam["rutaDEPC_FDIac_csv"] + "/*.csv")
    for archivo in archivos:
        archivo = archivo.replace("\\", "/")
        fecha = archivo.split("/")[-1].split("_")[2].split(".")[0]

        # Obtiene fecha del archivo de 3 lecturas anteriores
        lFechas = []
        for diasLectura in archivos:
            diasLectura = diasLectura.replace("\\", "/")
            fechaLectura = diasLectura.split("/")[-1].split("_")[2].split(".")[0]
```

```
            if not fechaLectura in lFechas:
                lFechas.append(fechaLectura)
        lFechas.sort()
        fechaInicial = lFechas[-3]

        # La fecha de los archivos leidos debe ser mayor a la fecha inicial
        # La fecha inicial debe ser mayor al periodo de los 3 días
        if fecha >= fechaInicial:
            print "Procesando fecha: %s" % fecha
            # Obtiene promedio de DPCobs de los 3 dias anteriores
            dicDPCobsPromedio3diasAnteriores = _1_3_3_DEPC_calculaPromedio3diasAnterioresDPCobs(dicParam, fecha)

            # Solo se procesa si encuentra DPC observado
            if len(dicDPCobsPromedio3diasAnteriores) == 0:
                print "No se encontraron archivos de DPCobs para la fecha ", fecha
            else:

                # Genera un archivo de salida por día
                archivo_diario = (dicParam["rutaDEPCsalida_csv"] + "/DEPCpred_FechA.csv").replace("FechA", fecha)
                f = open(archivo_diario, 'w')
                # encabezados
                f.write("fecha,uso_region,DEPC_FDIac,constante_c,DPCobs_prom3diaAnteriores,DEPCpred\n")

                # Dicconario para almacenar temporalmente los valores DPCpred
                dicTmpValoresDPCpred = {}

                # Lee archivo csv DEPC_FDIac (tiene un renglon por cada uso_region)
                archivoDPC1 = dicParam["rutaDEPC_FDIac_csv"] + "/DEPC_FDIac_%s.csv" % fecha
                dicDEPC_FDIac = _1_3_3_DEPC_cargaArchivoCSV_DEPC_FDIac(archivoDPC1)
                #dicDPCobs = dicDPCobsPromedio3diasAnteriores

                # ordena claves de uso_region
                l_orden_uso_region = []
                for clave_uso_region in dicDEPC_FDIac:
                    l_orden_uso_region.append(clave_uso_region)
                l_orden_uso_region.sort()

                for uso_region_DEPC_FDIac in l_orden_uso_region:
                    DEPC_FDIac = dicDEPC_FDIac[uso_region_DEPC_FDIac][1]
                    c = dic_tabla_c[uso_region_DEPC_FDIac]
                    if uso_region_DEPC_FDIac in dicDPCobsPromedio3diasAnteriores:
                        DPCobsDiasAnteriores = dicDPCobsPromedio3diasAnteriores[uso_region_DEPC_FDIac]
                    else:
                        DPCobsDiasAnteriores = 0.0

                    DPCpred = DEPC_FDIac + (c * DPCobsDiasAnteriores)

                    # Almacena temporalmente el valor de DPCpred para pasarlos al mapa
                    dicTmpValoresDPCpred[uso_region_DEPC_FDIac] = DPCpred
                    # Agrega registroa archivo CSV
                    registro = "%s,%s,%s,%s,%s,%s\n" % (fecha, uso_region_DEPC_FDIac, DEPC_FDIac, c, DPCobsDiasAnteriores, DPCpred)
                    f.write(registro)
                # Cierra archivo CSV
                f.close()


                # Combina la tabla de DPCpred con tabla de TIFF de USOREG
                # Resultado: un TIFF de DPCpred para cada USOREG
                # Un archivo TIFF de cada día con el nombre DPCpred_dia.tif
                #
                # Combina mapa USOREG con DPCpred
                #
                # Crea copia de mapa uso_reg para personalizarlo a este dia
                archivoTiffUsoRegionDeEsteDia = dicParam["rutaDEPCsalida_tif"] + "/DEPCpred_%s.tif" % fecha
                if not arcpy.Exists(archivoTiffUsoRegionDeEsteDia):
                    #arcpy.Delete_management(archivoTiffUsoRegionDeEsteDia)
                    arcpy.CopyRaster_management(dicParam["TIFFusoRegion"], archivoTiffUsoRegionDeEsteDia)

                _1_3_3_DEPC_agregaColumnasAusoreg(archivoTiffUsoRegionDeEsteDia)
                # rows = arcpy.UpdateCursor(archivoTiffUsoRegionDeEsteDia, "", "", "OID; VALUE; FECHA; USO_REGION; DPC1; DPCOBS_DA;
DPCPRED  ", "VALUE")
                # actualizaTIF(archivoTiffUsoRegionDeEsteDia, fecha, uso_region_DPC1, DPC1, DPCobsDiaAnterior, DPCpred)
                cursor = arcpy.UpdateCursor(archivoTiffUsoRegionDeEsteDia, "", "", "VALUE; fecha; DEPC_FDIac; DPCobs_da; DPCpred",
"VALUE;")
                for row in cursor:
                    VALUE = row.VALUE

                    DPC1 = 0
                    DPCobs_promedio = 0
                    DPCobs_da = 0
                    DPCpred = 0
                    if VALUE in dicDEPC_FDIac:
                        DEPC_FDIac = dicDEPC_FDIac[VALUE][1]
                    if VALUE in dicDPCobsPromedio3diasAnteriores:
                        DPCobs_promedio = dicDPCobsPromedio3diasAnteriores[VALUE]
                    if VALUE in dicTmpValoresDPCpred:
                        DPCpred = dicTmpValoresDPCpred[VALUE]

                    row.fecha = fecha
                    row.DEPC_FDIac = DEPC_FDIac
                    row.DPCobs_da = DPCobs_promedio
                    row.DPCpred = DPCpred
                    cursor.updateRow(row)
                if 'row' in locals():
                    del row
```

```
                if 'cursor' in locals():
                    del cursor


def _1_3_3_DEPC(dicParam):
    print "DEPC: _1_3_3_DEPC..."
    # Carga tabla de parametros de c en diccionario "dic_tabla_c"
    dic_tabla_c = _1_3_3_DEPC_cargaTablaDeConstante_c(dicParam)
    # Combina DEPCFDIac y DPCobs
    _1_3_3_DEPC_DEPCFDIac_DPCobs(dicParam, dic_tabla_c)

##### Termina DEPC ########


#
# Termina _1_3_DEPC
#############################################################################################################



#############################################################################################################
#
# _1_4_PSC
#

def _1_4_PSC_FDImean_to_dic(archivoEntradaFDImean):
    print "PSC: _1_4_PSC_FDImean_to_dic..."
    dicFDImean = {}
    f = open(archivoEntradaFDImean, 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        if i > 0:
            lLinea = linea.split(',')
            lFecha = lLinea[0].split('/')
            fecha = lFecha[2] + lFecha[1] + lFecha[0]
            ur = int(lLinea[1])
            FDImean = float(lLinea[4])
            llave = "%s-%s" % (fecha, ur)
            dicFDImean[llave] = FDImean
        i += 1
    # Cierra archivo CSV
    f.close()
    if 'f' in locals():
        del f
    if 'lineas' in locals():
        del lineas

    return dicFDImean


def _1_4_PSC_coeficientes_a_dic(archivoEntradaCoeficientes):
    print "PSC: _1_4_PSC_coeficientes_a_dic..."
    dicCoeficientes = {}
    f = open(archivoEntradaCoeficientes, 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        if i > 0:
            lLinea = linea.split(',')
            dicCoeficientes[int(lLinea[0])] = {"USO_REG": lLinea[1],
                                               "c0": float(lLinea[2]),
                                               "c2": float(lLinea[3]),
                                               "b0": float(lLinea[4]),
                                               "b2": float(lLinea[5].strip())
                                               }
        i += 1
    # Cierra archivo CSV
    f.close()
    if 'f' in locals():
        del f
    if 'lineas' in locals():
        del lineas

    return dicCoeficientes



# Graba el contenido del diccionario de datos en un archivo CSV de salida
def _1_4_PSC_dic_a_CSV(dicDatos, nombreArchivoCSV):
    print "PSC: _1_4_PSC_..."
    # crea archivo CSV de salida
    archivoCSV = open(nombreArchivoCSV, 'w')
    archivoCSV.write("VALUE,COUNT,FDI_DAY,USO_REGION,FDIM30,USO_REG,c0,c2,b0,b2,c,b,PAC,PCT,PCTmax,PSC\n")
    # genera llave y la ordena para mostrar datos ordenados
    dicLlaves = {}
    for value in dicDatos:
        ur = ("000%s" % dicDatos[value]['USO_REGION'])[-3:]
        fdi = ("000%s" % dicDatos[value]['FDI_DAY'])[-3:]
        llaveOrden ="%s-%s" % (ur, fdi)
        dicLlaves[llaveOrden] = value
    listaLlaves = dicLlaves.keys()
    listaLlaves.sort()
    listaLlavesOrdenadas = []
```

```
    for llave in listaLlaves:
        listaLlavesOrdenadas.append(dicLlaves[llave])
    # Recorre llaves ordenadas de diccionario para generar archivo CSV de salida
    for value in listaLlavesOrdenadas:
        # Elimina los registros donde no se pudo hacer el marge con el archivo "Coeficientes.csv"
        renglon = "%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s\n" % (value,
                                    dicDatos[value]['COUNT'],
                                    dicDatos[value]['FDI_DAY'],
                                    dicDatos[value]['USO_REGION'],
                                    dicDatos[value]['FDIM30'],
                                    dicDatos[value]['USO_REG'],
                                    dicDatos[value]['c0'],
                                    dicDatos[value]['c2'],
                                    dicDatos[value]['b0'],
                                    dicDatos[value]['b2'],
                                    dicDatos[value]['c'],
                                    dicDatos[value]['b'],
                                    dicDatos[value]['pac'],
                                    dicDatos[value]['pct'],
                                    dicDatos[value]['PCTmax'],
                                    dicDatos[value]['PSC']
                                    )
        archivoCSV.write(renglon)

    # Cierra archivo CSV
    archivoCSV.close()


def _1_4_PSC_agregaColumnasAtiff(archivoTIFF):
    print "PSC: _1_4_PSC_agregaColumnasAtiff..."
    # crea columna COUNT
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_COUNT")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_COUNT", "SHORT", "", "", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna FDIDAY
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_FDIDAY")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_FDIDAY", "SHORT", "", "", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna USOREG
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_USOREG")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_USOREG", "SHORT", "3", "", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna FDIM30
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_FDIM30")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_FDIM30", "DOUBLE", "12", "8", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna PSC_UR
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_UR")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_UR", "TEXT", "", "", "30", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna c0
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_c0")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_c0", "DOUBLE", "5", "2", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna c2
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_c2")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_c2", "DOUBLE", "5", "2", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna b0
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_b0")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_b0", "DOUBLE", "5", "2", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna b2
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_b2")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_b2", "DOUBLE", "5", "2", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna c
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_c")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_c", "DOUBLE", "12", "8", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna b
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_b")
    except:
        pass
```

```
    arcpy.AddField_management(archivoTIFF, "PSC_b", "DOUBLE", "12", "8", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna PAC
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_PAC")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_PAC", "DOUBLE", "13", "10", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna PCT
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_PCT")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_PCT", "DOUBLE", "13", "10", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna PCTMAX
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_PCTMAX")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_PCTMAX", "DOUBLE", "12", "8", "", "", "NULLABLE", "NON_REQUIRED", "")
    # crea columna PSC
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC_PSC")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC_PSC", "DOUBLE", "12", "8", "", "", "NULLABLE", "NON_REQUIRED", "")



def _1_4_PSC_generaMapas(dicParam, fecha, dicDatos):
    print "PSC: _1_4_PSC_generaMapas..."
    # ------------------------------------------------------------------------------
    # Combina la tabla de PSC con tabla de TIFF del TIFF combine
    # Resultado: un TIFF de PSC para cada dia
    # Un archivo TIFF de cada dia con el nombre DPCpred_dia.tif
    #
    # Combina mapa USOREG con DPCpred
    #
    # Crea copia de mapa uso_reg para personalizarlo a este dia
    archivoTIFF_origen = dicParam["DEPC_FDI_combines_tif"] + "/FDI_%s_Day_combine_usoreg.tif" % fecha
    archivoTIFF_destino = dicParam["rutaPSCsalida_tif"] + "/PSC_%s.tif" % fecha
    if not arcpy.Exists(archivoTIFF_destino):
        #arcpy.Delete_management(archivoTIFF_destino)
        arcpy.CopyRaster_management(archivoTIFF_origen, archivoTIFF_destino)
        # - Crea columnas en nuevo TIFF
        _1_4_PSC_agregaColumnasAtiff(archivoTIFF_destino)
        # Actualiza los datos en nuevo TIFF
        cursor = arcpy.UpdateCursor(archivoTIFF_destino, "", "",
            "VALUE; PSC_COUNT; PSC_FDIDAY; PSC_USOREG; PSC_FDIM30; PSC_UR; PSC_c0; PSC_c2; PSC_b0; PSC_b2; PSC_c; PSC_b; PSC_PAC;
PSC_PCT; PSC_PCTMAX; PSC_PSC",
            "")
        for row in cursor:
            if row.VALUE in dicDatos:
                #try:
                VALUE = row.VALUE
                #print "PSC_COUNT::", dicDatos[VALUE]['COUNT']
                row.PSC_COUNT = dicDatos[VALUE]['COUNT']
                row.PSC_FDIDAY = dicDatos[VALUE]['FDI_DAY']
                row.PSC_USOREG = dicDatos[VALUE]['USO_REGION']
                row.PSC_FDIM30 = dicDatos[VALUE]['FDIM30']
                row.PSC_UR = dicDatos[VALUE]['USO_REG']
                row.PSC_c0 = dicDatos[VALUE]['c0']
                row.PSC_c2 = dicDatos[VALUE]['c2']
                row.PSC_b0 = dicDatos[VALUE]['b0']
                row.PSC_b2 = dicDatos[VALUE]['b2']
                row.PSC_c = dicDatos[VALUE]['c']
                row.PSC_b = dicDatos[VALUE]['b']
                row.PSC_PAC = dicDatos[VALUE]['pac']
                row.PSC_PCT = dicDatos[VALUE]['pct']
                row.PSC_PCTMAX = dicDatos[VALUE]['PCTmax']
                row.PSC_PSC = dicDatos[VALUE]['PSC']
                cursor.updateRow(row)
                #except:
                #    print "__ERROR:"
                #    print "COUNT:", dicDatos[VALUE]['COUNT']
                #    print "FDI_DAY:", dicDatos[VALUE]['FDI_DAY']
                #    print "USO_REGION:", dicDatos[VALUE]['USO_REGION']
                #    print "FDIM30:", dicDatos[VALUE]['FDIM30']
                #    print "USO_REG:", dicDatos[VALUE]['USO_REG']
                #    print "c0:", dicDatos[VALUE]['c0']
                #    print "c2:", dicDatos[VALUE]['c2']
                #    print "b0:", dicDatos[VALUE]['b0']
                #    print "b2:", dicDatos[VALUE]['b2']
                #    print "c:", dicDatos[VALUE]['c']
                #    print "b:", dicDatos[VALUE]['b']
                #    print "pac:", dicDatos[VALUE]['pac']
                #    print "pct:", dicDatos[VALUE]['pct']
                #    print "PCTmax:", dicDatos[VALUE]['PCTmax']
                #    print "PSC:", dicDatos[VALUE]['PSC']
                #    exit(0)

    if 'row' in locals():
        del row
    if 'cursor' in locals():
        del cursor
```

```
#
# Obtiene el promedio del FDImean de los 30 dias anteriores a la llave que se reciba como parámetro
# la llave es la fecha-uso_region, ej: 20110411-2
#
def _1_4_PSC_obtieneFDImean30(dicFDImean, llave_busqueda):
    #print "PSC: _1_4_PSC_obtieneFDImean30..."
    dias_promedio = 30
    fecha_llave_busqueda = llave_busqueda.split("-")[0]
    uso_region_llave_busqueda = llave_busqueda.split("-")[1]

    fecha_tmp = datetime.datetime(int(fecha_llave_busqueda[0:4]), int(fecha_llave_busqueda[4:6]), int(fecha_llave_busqueda[6:8]))
    dias = timedelta(days=dias_promedio)
    fecha_inicial = fecha_tmp - dias
    fecha_inicial_txt = ("%s" % fecha_inicial)[0:10]
    fecha_inicial_txt = fecha_inicial_txt.replace("-", "")

    fecha_tmp = datetime.datetime(int(fecha_llave_busqueda[0:4]), int(fecha_llave_busqueda[4:6]), int(fecha_llave_busqueda[6:8]))
    dias = timedelta(days=1)
    fecha_final = fecha_tmp - dias
    fecha_final_txt = ("%s" % fecha_final)[0:10]
    fecha_final_txt = fecha_final_txt.replace("-", "")
    #print "fecha_inicial_txt:", fecha_inicial_txt
    #print "fecha_final_txt:", fecha_final_txt

    suma = 0
    i = 0
    for llave in dicFDImean:
        fecha = llave.split("-")[0]
        uso_region = llave.split("-")[1]
        if fecha >= fecha_inicial_txt and fecha <= fecha_final_txt and uso_region == uso_region_llave_busqueda:
            suma += dicFDImean[llave]
            i += 1
    return suma/i



def _1_4_PSC_genera_reporte(dicParam, dicFDImean, dicCoeficientes):
    print "PSC: _1_4_PSC_genera_reporte..."
    archivosTIFF = glob.glob(dicParam["DEPC_FDI_combines_tif"] + "/FDI_*_Day_combine_usoreg.tif")
    archivosTIFF.sort()

    ultimoTIFF = archivosTIFF[-1]
    archivoTIFF = ultimoTIFF.replace("\\", "/")

    #for archivoTIFF in archivosTIFF:
    #    archivoTIFF = archivoTIFF.replace("\\", "/")

    fechaArchivo = archivoTIFF.split("/")[-1].split("_")[1]



    # Obtiene fecha del archivo de 30 lecturas anteriores
    #lFechas = []
    #for diasLectura in archivosTIFF:
    #    diasLectura = diasLectura.replace("\\", "/")
    #    fechaLectura = diasLectura.split("/")[-1].split("_")[1]
    #    if not fechaLectura in lFechas:
    #        lFechas.append(fechaLectura)
    #lFechas.sort()
    #####fecha_inicio = lFechas[-30]
    #fecha_inicio = lFechas[-1]



    #if fechaArchivo >= fecha_inicio:

    print "Archivo:%s" % archivoTIFF
    # Lee tabla de imagen y lo guarda en un diccionario
    dicDatos = {}
    varFDI = "FDI_"+fechaArchivo[0:6]
    rows = arcpy.SearchCursor(archivoTIFF, "", "", "VALUE; COUNT; USO_REGION; " + varFDI + ";", "USO_REGION;" + varFDI + ";")
    i = 0
    tmpPAC = 0
    dicFDIM30 = {} # Se guardan las llaves ya consultadas para tomarlas de aquí y no buscarlas de nuevo
    for row in rows:
        # Solo se consideran los registros en donde hay ecuacion
        if row.USO_REGION in dicCoeficientes:
            # print "USO_REGION:", row.USO_REGION
            dicRow = {"COUNT": row.COUNT, "FDI_DAY": eval("row." + varFDI), "USO_REGION": row.USO_REGION}

            llave = "%s-%s" % (fechaArchivo, row.USO_REGION)
            # dicRow["FDIM30"] = dicFDImean[llave]
            if llave in dicFDIM30:
                dicRow["FDIM30"] = dicFDIM30[llave]
            else:
                dicRow["FDIM30"] = _1_4_PSC_obtieneFDImean30(dicFDImean, llave)
                dicFDIM30[llave] = dicRow["FDIM30"]
            # print "FDIM30:", dicRow["FDIM30"]

            # Datos de ecuacion
            dicRow["USO_REG"] = dicCoeficientes[row.USO_REGION]["USO_REG"]
            dicRow["c0"] = dicCoeficientes[row.USO_REGION]["c0"]
            dicRow["c2"] = dicCoeficientes[row.USO_REGION]["c2"]
            dicRow["b0"] = dicCoeficientes[row.USO_REGION]["b0"]
            dicRow["b2"] = dicCoeficientes[row.USO_REGION]["b2"]
```

```
            # ecuacion para calculo de c
            dicRow["c"] = dicRow["c0"] * dicRow["FDIM30"] ** dicRow["c2"]
            # ecuacion para calculo de b
            dicRow["b"] = dicRow["b0"] * dicRow["FDIM30"] + dicRow["b2"]
            # ecuación para el calculo del porcentaje acumulado (PAC) =1-EXP(-((M2/L2)^K2))
            if dicRow["b"] > 0:
                dicRow["pac"] = 1 - math.exp(-((dicRow["FDI_DAY"] / dicRow["b"]) ** dicRow["c"]))
            else:
                dicRow["pac"] = 0
            # ecuación para el cálculo de pct   =SI((N3-N2)<0,N3,(N3-N2))
            if i > 0:
                if (dicRow["pac"] - tmpPAC) < 0:
                    dicRow["pct"] = dicRow["pac"]
                else:
                    dicRow["pct"] = dicRow["pac"] - tmpPAC
            else:
                dicRow["pct"] = 0
            #if dicRow["FDI_DAY"] > 0:
            #    print "USO_REGION:%s PAC:%s  PACant:%s  pct:%s" % (dicRow["USO_REGION"], dicRow["pac"], tmpPAC, dicRow["pct"])
            #    exit(0)
            tmpPAC = dicRow["pac"]
            dicDatos[row.VALUE] = dicRow
            i += 1

    if i == 0:
        print "Error:", row
    # Para calcular la columna "PCTmax" debo hacerlo al final, ya que debo
    # calcular el máximo para la columna PCT por cada uso region y no lo puedo
    # saber hasta que termine de procesar cada uso region.
    # calcula "PCTmax" por uso_region
    dicPctMax = {}
    for value in dicDatos:
        if dicDatos[value]['USO_REGION'] in dicPctMax:
            if dicDatos[value]['pct'] > dicPctMax[dicDatos[value]['USO_REGION']]:
                dicPctMax[dicDatos[value]['USO_REGION']] = dicDatos[value]['pct']
        else:
            dicPctMax[dicDatos[value]['USO_REGION']] = dicDatos[value]['pct']
    # Actualiza dicDatos para agregarle el PCTmax
    for value in dicDatos:
        dicDatos[value]['PCTmax'] = dicPctMax[dicDatos[value]['USO_REGION']]

    # Calcula columna PSC
    # v1
    #for value in dicDatos:
    #    if dicDatos[value]['PCTmax'] > 0:
    #        if dicDatos[value]['FDI_DAY'] < dicDatos[value]['b']:
    #            dicDatos[value]['PSC'] = dicDatos[value]['pct'] / dicDatos[value]['PCTmax'] * 200
    #        else:
    #            dicDatos[value]['PSC'] = (60 + 40 * dicDatos[value]['pct'] / dicDatos[value]['PCTmax']) * 2
    #    else:
    #        dicDatos[value]['PSC'] = 0
    # v2
    #for value in dicDatos:
    #    if dicDatos[value]['PCTmax'] > 0:
    #        dicDatos[value]['PSC'] = dicDatos[value]['pct'] / dicDatos[value]['PCTmax'] * 200
    #    else:
    #        dicDatos[value]['PSC'] = 0
    # v3
    for value in dicDatos:
        if dicDatos[value]['PCTmax'] > 0:
            if dicDatos[value]['FDI_DAY'] < dicDatos[value]['b']:
                dicDatos[value]['PSC'] = dicDatos[value]['pct'] / dicDatos[value]['PCTmax'] * 200
            else:
                dicDatos[value]['PSC'] = (20 + 80 * dicDatos[value]['pct'] / dicDatos[value]['PCTmax']) * 2
        else:
            dicDatos[value]['PSC'] = 0

    # Manda contenido de diccionario a archivo CSV
    nombreArchivoCSV = dicParam["rutaPSCsalida_csv"] + "/PSC_" + fechaArchivo + ".csv"
    #print "Escribiendo renglon en CSV..."
    _1_4_PSC_dic_a_CSV(dicDatos, nombreArchivoCSV)


    # Genera mapas
    #print "Generando mapa..."
    _1_4_PSC_generaMapas(dicParam, fechaArchivo, dicDatos)



def _1_4_PSC(dicParam):
    print "PSC: _1_4_PSC..."
    # Carga FDImean a un diccionario para hacer las busquedas rápido
    dicFDImean = _1_4_PSC_FDImean_to_dic(dicParam["archivoPSCEntradaFDImean"])
    # Carga archivo CSV de coeficientes a diccionario
    dicCoeficientes = _1_4_PSC_coeficientes_a_dic(dicParam["archivoPSCcoeficientes"])
    # Lee tabla de tiffs de FDI combines y los pasa a un diccionario de datos
    _1_4_PSC_genera_reporte(dicParam, dicFDImean, dicCoeficientes)

#
# Termina _1_4_PSC
################################################################################################################
```

```
#############################################################################################################
#
# _1_5_RISC
#

# Carga archivo CSV de PSC a diccionario
def _1_5_RISC_csv_a_dicPSC(archivoCSV):
    print "RISC: _1_5_RISC_csv_a_dicPSC..."
    dicSalida = {}
    if os.path.isfile(archivoCSV):
        f = open(archivoCSV, 'r')
        lineas = f.readlines()
        i = 0
        for linea in lineas:
            if i > 0:
                lLinea = linea.split(',')
                dicSalida[int(lLinea[0])] = {"PSC": float(lLinea[15])}
            i += 1
        # Cierra archivo CSV
        f.close()
        if 'f' in locals():
            del f
        if 'lineas' in locals():
            del lineas
    else:
        print "No existe el archivo CSV: %s" % archivoCSV

    return dicSalida


# Carga archivo CSV de DPCpred a diccionario
def _1_5_RISC_csv_a_dicDPCpred(archivoCSV):
    print "RISC: _1_5_RISC_csv_a_dicDPCpred..."
    dicSalida = {}
    if os.path.isfile(archivoCSV):
        f = open(archivoCSV, 'r')
        lineas = f.readlines()
        i = 0
        for linea in lineas:
            if i > 0:
                lLinea = linea.split(',')
                dicSalida[int(lLinea[1])] = {"DPCP": float(lLinea[5])}
            i += 1
        # Cierra archivo CSV
        f.close()
        if 'f' in locals():
            del f
        if 'lineas' in locals():
            del lineas
    else:
        print "No existe el archivo CSV: %s" % archivoCSV

    return dicSalida


def _1_5_RISC_agregaColumnasAtiff(archivoTIFF):
    print "RISC: _1_5_RISC_agregaColumnasAtiff..."
    # crea columna DPCP (DPCpred)
    try:
        arcpy.DeleteField_management(archivoTIFF, "DPCP")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "DPCP", "FLOAT", "14", "11", "", "", "NULLABLE", "NON_REQUIRED", "")

    # crea columna DPCPreclas (DPCpred reclasificado)
    try:
        arcpy.DeleteField_management(archivoTIFF, "DPCPreclas")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "DPCPreclas", "FLOAT", "14", "11", "", "", "NULLABLE", "NON_REQUIRED", "")

    # crea columna PSC
    try:
        arcpy.DeleteField_management(archivoTIFF, "PSC")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "PSC", "FLOAT", "14", "11", "", "", "NULLABLE", "NON_REQUIRED", "")

    # crea columna RISC
    try:
        arcpy.DeleteField_management(archivoTIFF, "RISC")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "RISC", "SHORT", "", "", "", "", "NULLABLE", "NON_REQUIRED", "")



def _1_5_RISC_joinFDIcombine_DPCpred(dicParam, archivoTiffFDIcombine, fecha):
    print "RISC: _1_5_RISC_joinFDIcombine_DPCpred..."
    #
    # Obtiene datos del csv de DPCpred de este dia
    #
```

```
    fechaDPCpred = "%s-%s-%s" % (fecha[0:4], fecha[4:6], fecha[6:8])
    archivoDPCpred = (dicParam["rutaDEPCsalida_csv"] + "/DEPCpred_FechA.csv").replace("FechA", fechaDPCpred)
    dicDPCpred = _1_5_RISC_csv_a_dicDPCpred(archivoDPCpred)
    # Reclasifica columna DPCpred  y la agrega al diccionario de DPCpred
    for uso_region in  dicDPCpred:
        if dicDPCpred[uso_region]['DPCP'] > 300:
            dicDPCpred[uso_region]['DPCPreclas'] = 140
        else:
            if dicDPCpred[uso_region]['DPCP'] > 100:
                dicDPCpred[uso_region]['DPCPreclas'] = 110
            else:
                dicDPCpred[uso_region]['DPCPreclas'] = dicDPCpred[uso_region]['DPCP']

    #
    # Obtiene datos del csv de PSC de este dia
    #
    archivoPSC = (dicParam["rutaPSCsalida_csv"] + "/PSC_FechA.csv").replace("FechA", fecha)
    dicPSC = _1_5_RISC_csv_a_dicPSC(archivoPSC)

    #
    # Une datos de DPCpred y PSC al mapa (join)
    #
    # Crea copia de mapa
    archivoTIFF_origen = archivoTiffFDIcombine
    archivoTIFF_destino = dicParam["rutaRISCsalidaPreliminar"] + "/RISC_%s.tif" % fecha
    borra_archivo(archivoTIFF_destino)
    arcpy.CopyRaster_management(archivoTIFF_origen, archivoTIFF_destino)
    # - Crea columnas en nuevo TIFF
    _1_5_RISC_agregaColumnasAtiff(archivoTIFF_destino)
    # Actualiza los datos en nuevo TIFF
    cursor = arcpy.UpdateCursor(archivoTIFF_destino, "", "", "VALUE; USO_REGION; DPCP; DPCPreclas; PSC; RISC;", "")
    #cursor = arcpy.UpdateCursor(archivoTIFF_destino, "", "", "VALUE; USO_REGION; DPCP; PSC; RISC;", "")
    for row in cursor:
        #try:

        DPCP = 0
        DPCPreclas = 0
        RISC = 0
        PSC = 0
        # Se le pone 1001 (AH) a RISC si USO_REGION [1, 8, 15, 21, 26, 36]
        # Se le pone 1002 (CA) a RISC si USO_REGION [3, 9, 14, 20, 28, 34]
        if int(row.USO_REGION) in [1, 8, 15, 21, 26, 36]:
            RISC = 1001
        elif int(row.USO_REGION) in [3, 9, 14, 20, 28, 34]:
            # 1002 era para a primer versión
            #RISC = 1002
            RISC = 65535
        else:
            # Columnas de DPCpred
            if row.USO_REGION in dicDPCpred:
                DPCP = dicDPCpred[row.USO_REGION]['DPCP']
                DPCPreclas = dicDPCpred[row.USO_REGION]['DPCPreclas']
            # Columna de PSC
            if row.VALUE in dicPSC:
                PSC = dicPSC[row.VALUE]['PSC']
            #if DPCP * PSC > 0:
            if DPCPreclas > 0 and PSC > 0:
                RISC = int(DPCPreclas * PSC / 100)

        # row.DPCP = DPCP
        row.DPCPreclas = DPCPreclas
        row.PSC = PSC
        row.RISC = RISC
        cursor.updateRow(row)

        #except:
        #    print "\n__ERROR: no se pudo actualizar la tabla del mapa con estos valores:"
        #    print "Mapa:", archivoTIFF_destino
        #    print "VALUE: ", row.VALUE
        #    print "row.USO_REGION: ", row.USO_REGION
        #    if row.USO_REGION in dicDPCpred:
        #        print "DPCP:", dicDPCpred[row.USO_REGION]['DPCP']
        #        print "type:", type(dicDPCpred[row.USO_REGION]['DPCP'])
        #        print "DPCPreclas:", dicDPCpred[row.USO_REGION]['DPCPreclas']
        #        print "type:", type(dicDPCpred[row.USO_REGION]['DPCPreclas'])
        #    if row.VALUE in dicPSC:
        #        print "PSC:", dicPSC[row.VALUE]['PSC']
        #        print "type:", type(dicPSC[row.VALUE]['PSC'])
        #    #if dicDPCpred[row.USO_REGION]['DPCPreclas'] > 0 & dicPSC[row.VALUE]['PSC']:
        #    if dicPSC[row.VALUE]['PSC']:
        #        print "RISC:", RISC
        #        print "type:", type(RISC)
        #    exit(0)

    if 'row' in locals():
        del row
    if 'cursor' in locals():
        del cursor


def _1_5_RISC_cambia_valor_de_VALUE_a_RISC(dicParam, fecha):
    print "RISC: _1_5_RISC_cambia_valor_de_VALUE_a_RISC..."
    archivo_entrada = dicParam["rutaRISCsalidaPreliminar"] + "/RISC_%s.tif" % fecha
    archivo_tmp_tif = archivo_entrada.replace(dicParam['rutaRISCsalidaPreliminar'], dicParam['dirTMP'])
    archivo_tmp_shp = archivo_tmp_tif.replace(".tif", ".shp")
```

```
    archivo_final= dicParam["rutaRISCsalidaFinal"] + "/RISC_%s.tif" % fecha

    borra_archivo(archivo_tmp_tif)
    arcpy.CopyRaster_management(archivo_entrada, archivo_tmp_tif)
    borra_archivo(archivo_tmp_shp)
    arcpy.RasterToPolygon_conversion(archivo_tmp_tif, archivo_tmp_shp, "NO_SIMPLIFY", "RISC")

    archivoTIFF_cellsize = archivo_entrada
    borra_archivo(archivo_final)
    arcpy.PolygonToRaster_conversion(archivo_tmp_shp, "GRIDCODE", archivo_final, "CELL_CENTER", "GRIDCODE", archivoTIFF_cellsize)




def _1_5_RISC(dicParam):
    print "RISC: _1_5_RISC..."
    archivosTIFF = glob.glob(dicParam["DEPC_FDI_combines_tif"] + "/FDI_*_Day_combine_usoreg.tif")
    for archivoTIFF in archivosTIFF:
        archivoTIFF = archivoTIFF.replace("\\", "/")
        fechaArchivo = archivoTIFF.split("/")[-1].split("_")[1]

        if fechaArchivo >= '20170918':

            # Si ya existe el RISC preliminar de salida, ya no lo genera
            archivoRISCpreliminarSalida = dicParam['rutaRISCsalidaPreliminar'] + "/RISC_" + fechaArchivo + ".tif"
            if not arcpy.Exists(archivoRISCpreliminarSalida):
                print "Archivo preliminar:%s" % archivoRISCpreliminarSalida
                #
                # Hace join de tabla del TIFF de "FDI combine" por el campo "uso_region" con CSV de "DPCpred"
                #
                _1_5_RISC_joinFDIcombine_DPCpred(dicParam, archivoTIFF, fechaArchivo)

            # Si ya existe el RISC final de salida, ya no lo genera
            archivoRISCfinalSalida = dicParam['rutaRISCsalidaFinal'] + "/RISC_" + fechaArchivo + ".tif"
            if not arcpy.Exists(archivoRISCfinalSalida):
                print "Archivo final:%s" % archivoRISCfinalSalida
                #
                # Cambia el valor de la columna RISC de la imagen al VALUE
                #
                _1_5_RISC_cambia_valor_de_VALUE_a_RISC(dicParam, fechaArchivo)
#
# Termina _1_5_RISC
#############################################################################################################################




#############################################################################################################################
#
# Actualiza los archivos finales de salida en la pagina WEB
#

def actualizaPaginaWEB(dicParam):
    #
    # Actualiza imagen FDI ( indice_de_combustible_seco )
    #
    print "Actualiza imagen FDI..."
    archivos = glob.glob(dicParam["rutaFDI_FDI"] + "/*.tif")
    if len(archivos) > 0:
        archivos.sort()
        ultimoArchivoTIFF = archivos[-1]
        ultimoArchivoTIFF = ultimoArchivoTIFF.replace("\\", "/")

        archivo_entrada = ultimoArchivoTIFF
        archivo_salida = dicParam["archivoTIFF_FDIsalidaDiaria"]

        if arcpy.Exists(archivo_salida):
            arcpy.Delete_management(archivo_salida)
        arcpy.CopyRaster_management(archivo_entrada, archivo_salida)

    #
    # Actualiza imagen RISC ( riesgo_de_ocurrencia_de_incendio )
    #
    print "Actualiza imagen RISC..."
    archivos = glob.glob(dicParam['rutaRISCsalidaFinal'] + "/*.tif")
    if len(archivos) > 0:
        archivos.sort()
        ultimoArchivoTIFF = archivos[-1]
        ultimoArchivoTIFF = ultimoArchivoTIFF.replace("\\", "/")

        archivo_entrada = ultimoArchivoTIFF
        archivo_salida = dicParam["archivoTIFF_RISCsalidaDiaria"]

        if arcpy.Exists(archivo_salida):
            arcpy.Delete_management(archivo_salida)
        arcpy.CopyRaster_management(archivo_entrada, archivo_salida)


#############################################################################################################################
#
# Ejecuta proceso ROI Modulo de Riesgo Peligro
#
```

```
def _1_6_ROI(dicParam):
    #
    # Genera TIF de Riesgo de Ocurrencia de Incendio
    #
    archivos = glob.glob(dicParam["rutaRISCsalidaFinal"] + "/*.tif")
    # Se ordenan, con la finalidad que se ordenen por fecha
    archivos.sort()
    archivo_generado = 0
    ultimo_archivo_ROIm_generado = ""
    for archivo in archivos:
        archivo = archivo.replace("\\", "/")
        nombreArchivoSalidaTifROI = archivo.replace("/out/_1_5_RISC/final/RISC_", "/out/_1_6_ROI/ROI/ROI_")
        if not arcpy.Exists(nombreArchivoSalidaTifROI):
            print "Generando ROI:", nombreArchivoSalidaTifROI
            rasterEntradaTifRISC = arcpy.Raster(archivo)
            rasterEntradaTifROI100 = arcpy.Raster(dicParam["ROI_inROI100"])

            #Raster Calculator
            # Genera archivo ROI
            multiplicacion = rasterEntradaTifRISC * rasterEntradaTifROI100
            resultado = multiplicacion / 100
            resultado.save(nombreArchivoSalidaTifROI)

        # Genera archivo ROIm (ROI x mascara)
        nombreArchivoSalidaTifROImask = nombreArchivoSalidaTifROI.replace("/out/_1_6_ROI/ROI/ROI_", "/out/_1_6_ROI/ROImask/ROIm_")
        if not arcpy.Exists(nombreArchivoSalidaTifROImask):
            print "Generando ROIm:", nombreArchivoSalidaTifROImask
            rasterEntradaROI = arcpy.Raster(nombreArchivoSalidaTifROI)
            rasterEntradaROImask = arcpy.Raster(dicParam["ROI_inROImask"])
            resultado = rasterEntradaROI * rasterEntradaROImask
            resultado.save(nombreArchivoSalidaTifROImask)

            archivo_generado = 1
            # Almacena el nombre del último archivo TIFF generado
            ultimo_archivo_ROIm_generado = nombreArchivoSalidaTifROImask

    # Si se generó un archivo nuevo, lo actualiza en el servidor
    if archivo_generado == 1:
        print "Copiando archivo ROI a carpeta WEB..."
        print "Archivo generado:", ultimo_archivo_ROIm_generado
        print "Archivo web:", dicParam["ROI_outROIweb"]
        # Borra archivo publicado en el web
        if arcpy.Exists(dicParam["ROI_outROIweb"]):
            arcpy.Delete_management(dicParam["ROI_outROIweb"])
        # Copia archivo de carpeta de trabajo a carpeta WEB
        arcpy.CopyRaster_management(ultimo_archivo_ROIm_generado, dicParam["ROI_outROIweb"])



#
# Termina - Actualiza los archivos finales de salida en la pagina WEB
#############################################################################################################
```

```
# -*- coding: utf-8 -*-
#
# Actualiza los archivos KML que deposita CONABIO en el servidor FTP de UJED
# El flujo de datos es desde la carpeta de FTP a la carpeta de entrada para el proceso
#
# Este proceso se ejecuta automáticamente cada 15 min. desde las 11:15hrs hasta las 23:30hrs
# desde el programador de tareas de windows
# debido a que el proceso es manual, varía la hora en que lo depositan, por esa razón hay que verificar cada 15 min
# rocorre toda la lista de archivos en el directorio FTP (ftp) y actualiza los que no se hayan copiado
# a la carpeta de entrada (in)
#
# Autor: Jaime Briseño Reyes <jaime.briseno@gmail.com>
# Fecha: 18/Mayo/2018
#

import os
import shutil
import glob
import arcpy
import time

rutaInAlmacenKMLs = "D:/proyectos/incendios/incendios_activos/in/KML"
rutaInAlmacenShapes = "D:/proyectos/incendios/incendios_activos/in/shapes"
rutaInFTP = "D:/ftp/conafor_incendios"
rutaOutWEB = "C:/xampp/htdocs/incendios/cartografia/capas/conafor/shape"
rutaTMP = "D:/proyectos/incendios/incendios_activos/tmp"

# Formato de archivos de entrada FTP, tal como los depositan
# KML Incendios Activos 2018 05 08 11 hr..kmz
# kml Activos 16 hrs 08-05-18.kmz
# KML Activos 2018-05-12  16 horas.kmz
# Kml Activos 20 hrs 08-05-18.kmz
# KML Activos 2018-05-12   20 horas.kmz

# Formato ded salida, con este nombre los lee el servidor de mapas Geoserver
# incendios_activos_11hr.shp
# incendios_activos_16hr.shp
# incendios_activos_20hr.shp

# Con este formato se guardan en el almacen de KMLs y SHAPEs
plantillaNombreArchivos = "incendios_activos_AAAA_MM_DD_HHhrs"
```

```
nombreArchivo11hrsWEB = "incendios_activos_11hr.shp"
nombreArchivo16hrsWEB = "incendios_activos_16hr.shp"
nombreArchivo20hrsWEB = "incendios_activos_20hr.shp"

# Sobre-escribe el archivo de salida
arcpy.env.overwriteOutput = True



# Función para borrar archivos
def borra_archivo(archivo_a_borrar):
    if arcpy.Exists(archivo_a_borrar):
        arcpy.Delete_management(archivo_a_borrar)


#
# Actualizacion del historial de Archivos de puntos MODIS
#
def actualiza_historial_KML():
    print "Actualizando historial de archivos KML..."
    archivos = glob.glob(rutaInFTP + "/*")
    for archivo_original in archivos:
        archivo_original = archivo_original.replace("\\", "/")
        nombre_archivo_original = archivo_original.split("/")[-1]

        nombre_archivo = "NO_asignado"
        # le crea un nombre estandarizado a los archivos
        if " 11 hrs." in nombre_archivo_original or " 11hrs." in nombre_archivo_original : # KML Incendios Activos 2018 05 18
11hrs.kmz
            anio = nombre_archivo_original.split(" ")[3]
            mes = nombre_archivo_original.split(" ")[4]
            dia = nombre_archivo_original.split(" ")[5]
            nombre_archivo = plantillaNombreArchivos.replace("AAAA", anio).replace("MM", mes).replace("DD", dia).replace("HH", "11")
+ ".kmz"
            nombreArchivoHrsWEB = nombreArchivo11hrsWEB
        if " 11 hr." in nombre_archivo_original: # KML Incendios Activos 2018 05 08 11 hr..kmz
            anio = nombre_archivo_original.split(" ")[3]
            mes = nombre_archivo_original.split(" ")[4]
            dia = nombre_archivo_original.split(" ")[5]
            nombre_archivo = plantillaNombreArchivos.replace("AAAA", anio).replace("MM", mes).replace("DD", dia).replace("HH", "11")
+ ".kmz"
            nombreArchivoHrsWEB = nombreArchivo11hrsWEB

        if " 16 hrs." in nombre_archivo_original: # KML Incendios Activos 2018 05 18 16hrs.kmz
            anio = nombre_archivo_original.split(" ")[3]
            mes = nombre_archivo_original.split(" ")[4]
            dia = nombre_archivo_original.split(" ")[5]
            nombre_archivo = plantillaNombreArchivos.replace("AAAA", anio).replace("MM", mes).replace("DD", dia).replace("HH", "16")
+ ".kmz"
            nombreArchivoHrsWEB = nombreArchivo11hrsWEB
        if " 16 hrs " in nombre_archivo_original: # kml Activos 16 hrs 08-05-18.kmz
            fecha = nombre_archivo_original.split(" ")[4].split(".")[0]
            anio = "20" + fecha.split("-")[2]
            mes = fecha.split("-")[1]
            dia = fecha.split("-")[0]
            nombre_archivo = plantillaNombreArchivos.replace("AAAA", anio).replace("MM", mes).replace("DD", dia).replace("HH", "16")
+ ".kmz"
            nombreArchivoHrsWEB = nombreArchivo16hrsWEB
        if " 16 horas." in nombre_archivo_original: # KML Activos 2018-05-12  16 horas.kmz
            fecha = nombre_archivo_original.split(" ")[2]
            anio = fecha.split("-")[0]
            mes = fecha.split("-")[1]
            dia = fecha.split("-")[2]
            nombre_archivo = plantillaNombreArchivos.replace("AAAA", anio).replace("MM", mes).replace("DD", dia).replace("HH", "16")
+ ".kmz"
            nombreArchivoHrsWEB = nombreArchivo16hrsWEB

        if " 20 hrs." in nombre_archivo_original: # KML Incendios Activos 2018 05 18 20hrs.kmz
            anio = nombre_archivo_original.split(" ")[3]
            mes = nombre_archivo_original.split(" ")[4]
            dia = nombre_archivo_original.split(" ")[5]
            nombre_archivo = plantillaNombreArchivos.replace("AAAA", anio).replace("MM", mes).replace("DD", dia).replace("HH", "20")
+ ".kmz"
            nombreArchivoHrsWEB = nombreArchivo11hrsWEB
        if " 20 hrs " in nombre_archivo_original: # Kml Activos 20 hrs 08-05-18.kmz
            fecha = nombre_archivo_original.split(" ")[4].split(".")[0]
            anio = "20" + fecha.split("-")[2]
            mes = fecha.split("-")[1]
            dia = fecha.split("-")[0]
            nombre_archivo = plantillaNombreArchivos.replace("AAAA", anio).replace("MM", mes).replace("DD", dia).replace("HH", "20")
+ ".kmz"
            nombreArchivoHrsWEB = nombreArchivo20hrsWEB
        if " 20 horas." in nombre_archivo_original: # KML Activos 2018-05-12   20 horas.kmz
            fecha = nombre_archivo_original.split(" ")[2]
            anio = fecha.split("-")[0]
            mes = fecha.split("-")[1]
            dia = fecha.split("-")[2]
            nombre_archivo = plantillaNombreArchivos.replace("AAAA", anio).replace("MM", mes).replace("DD", dia).replace("HH", "20")
+ ".kmz"
            nombreArchivoHrsWEB = nombreArchivo20hrsWEB


        # Actualiza almacén de archivos SHP
        # Si no existe el KML destino, lo copia desde la carpeta original de FTP
        if ".kmz" in nombre_archivo:
```

```
            archivoKMLSalidaAlmacen = rutaInAlmacenKMLs + "/" + nombre_archivo
            if not os.path.isfile(archivoKMLSalidaAlmacen):
                print "Actualizando en almacén: ", archivoKMLSalidaAlmacen
                shutil.copy(archivo_original, archivoKMLSalidaAlmacen)

                # Ahora el archivo actual es el archivo recien copiado
                archivo = archivoKMLSalidaAlmacen
                nombre_archivo = archivo.split("/")[-1]

                # Convierte KML a shape
                nombreArchivoHrsWEB = ''
                if "11hrs" in nombre_archivo:
                    nombreArchivoHrsWEB = nombreArchivo11hrsWEB
                elif "16hrs" in nombre_archivo:
                    nombreArchivoHrsWEB = nombreArchivo16hrsWEB
                elif "20hrs" in nombre_archivo:
                    nombreArchivoHrsWEB = nombreArchivo20hrsWEB
                else:
                    pass

                if nombreArchivoHrsWEB:
                    # Convierte KML a Layer
                    print "Genera layer..."
                    borra_archivo(rutaTMP + "/" + nombre_archivo + ".gdb")
                    borra_archivo(rutaTMP + "/" + archivo + ".lyr")
                    nombreArchivoWEBsinExtension = nombreArchivoHrsWEB.split(".shp")[0]
                    arcpy.KMLToLayer_conversion(archivo, rutaTMP, nombreArchivoWEBsinExtension, "NO_GROUNDOVERLAY")

                    # Convierte Layer a Shape
                    print "Genera shape..."
                    entrada = rutaTMP + "/" + nombreArchivoWEBsinExtension + ".gdb/Placemarks/Points"
                    salida = rutaTMP
                    borra_archivo(salida + "/" + "Points.shp")
                    arcpy.FeatureClassToShapefile_conversion(entrada, salida)

                    # Copia archivo "Points" a ruta de Almacen de shapes
                    borra_archivo(rutaInAlmacenShapes + "/" + nombre_archivo)
                    arcpy.CopyFeatures_management(rutaTMP + "/" + "Points.shp", rutaInAlmacenShapes + "/" + nombre_archivo)

#
#  Obtiene el nombre del ultimo archivo shape del almacen de shapes
#
def obtiene_ultimo_shape(rutaInAlmacenShapes, tipo):
    archivos = glob.glob(rutaInAlmacenShapes + "/*" + tipo + "hrs*.shp")
    ultimo_archivo = False
    if len(archivos) > 0:
        archivos.sort()
        ultimo_archivo = archivos[-1]
        ultimo_archivo = ultimo_archivo.replace("\\", "/")
    print "Ultimo archivo tipo %s: %s" % (tipo, ultimo_archivo)
    return ultimo_archivo

#
# Genera un archivo de texto con el nombre según el tipo de archivo,
# de contenido se le pone la fecha, para saber de que fecha es el último archivo creado
#
def genera_archivo_txt_con_fecha(archivo, tipo):
    print "Genera archivo txt con fecha de ultimo archivo de %s hrs..." % tipo
    #
    # Genera un archivo de texto con el nombre del último archivo encontrado
    # Para poder determinar de que dia es según el tipo (11, 16 y 20hrs)
    #
    # obtiene el puro nombre del archivo
    nombre_archivo = archivo.split("/")[-1]
    archivo_txt = open(rutaOutWEB + "/fecha_ultimo_archivo_" + tipo + '_hrs.txt', 'w')
    # Agrega contenido al archivo (fecha de último archivo de este tipo)
    anio_archivo = nombre_archivo.split("_")[2]
    mes_archivo = nombre_archivo.split("_")[3]
    dia_archivo = nombre_archivo.split("_")[4]
    linea = "%s/%s/%s" % (dia_archivo, mes_archivo, anio_archivo)
    archivo_txt.write(linea)
    # Cierra archivo
    archivo_txt.close()


# Copia el archivo shape origen al destino remplazando
def copia_archivo_shape_remplazando(origen, destino):
    if arcpy.Exists(destino):
        arcpy.Delete_management(destino)
    arcpy.CopyFeatures_management(origen, destino)


#
# Proceso principal
#
actualiza_historial_KML()

#
# Obtiene los tres últimos archivos de cada tipo (11, 16 y 20 hrs) del almacen de shapes
#
ultimo_shape_11hrs = obtiene_ultimo_shape(rutaInAlmacenShapes, "11")
ultimo_shape_16hrs = obtiene_ultimo_shape(rutaInAlmacenShapes, "16")
ultimo_shape_20hrs = obtiene_ultimo_shape(rutaInAlmacenShapes, "20")

#
```

```
# Genera un archivo de texto con el nombre según el tipo de archivo,
# de contenido se le pone la fecha, para saber de que fecha es el último archivo creado
#
genera_archivo_txt_con_fecha(ultimo_shape_11hrs, "11")
genera_archivo_txt_con_fecha(ultimo_shape_16hrs, "16")
genera_archivo_txt_con_fecha(ultimo_shape_20hrs, "20")

#
# Toma los últimos 3 shapes (11, 16 y 20) y genera los copia con un nombre generico
# a la carpeta donde los toma el geoserver para publicar en el sistema WEB
#
copia_archivo_shape_remplazando(ultimo_shape_11hrs, rutaOutWEB + "/" + nombreArchivo11hrsWEB)
copia_archivo_shape_remplazando(ultimo_shape_16hrs, rutaOutWEB + "/" + nombreArchivo16hrsWEB)
copia_archivo_shape_remplazando(ultimo_shape_20hrs, rutaOutWEB + "/" + nombreArchivo20hrsWEB)

fecha = time.strftime("%d/%m/%Y")
hora = time.strftime("%H:%M:%S")
print "Terminado: %s %s" % (fecha, hora)
```

```
# -*- coding: utf-8 -*-
#
# Proceso diario de incendios NINC PRED
# (NINC PRED: Número de incendios predichos)
# El resultado del proceso es publicado en el sitio WEB de Incendios
#
# Autor: Jaime Briseño Reyes <jaime.briseno@gmail.com>
# Fecha: 26/Mayo/2018
#


import time
today = time.strftime("%d/%m/%Y")
todayYMD = time.strftime("%Y/%m/%d")

print "Inicio: %s %s\n" % (todayYMD, time.strftime("%H:%M:%S") )

import os
import arcpy
import glob
import datetime
from datetime import timedelta


rutaBase = "D:/proyectos/incendios/proceso_diario/NINC_PRED"
rutaBasePaginaWEB = "C:/xampp/htdocs/incendios"

#
# Diccionario para almacenar las variables de configuración
# estas variables deben cambiarse si hay un cambio de servidor
#
dicParam = {
    "rutaBase": rutaBase,
    "rutaBasePaginaWEB": rutaBasePaginaWEB,
    "fechaHoy": today,
    "fechaHoyDia": today.split("/")[0],
    "fechaHoyMes": today.split("/")[1],
    "fechaHoyAnio": today.split("/")[2],
    "inTIFFestados": rutaBase + "/in/estados.tif",
    ### FDImean
    "FDImean_in_tifFDI": "D:/proyectos/incendios/proceso_diario/out/_1_2_FDI/FDI",
    "FDImean_out_tifFDI": rutaBase + "/FDImean/out/tif",
    "FDImean_out_csv": rutaBase + "/FDImean/out/csv/FDI_mean_estados.csv",
    # Si el proceso FDImean procesa un archivo nuevo, se enciende esta bandera a 1
    # Si esta bandera se enciende a 1, entonces se continua con los procesos posteriores
    "FDImean_procesa_archivo_nuevo": 0,
    ### FDIacumulado
    "FDIacumulado_out_FDIm50_csv": rutaBase + "/FDIacumulado/out/FDIm50_EDO_FechA.csv",
    ### FDIforecast
    "FDIforecast_in_archivo_parametros_forecast_csv": rutaBase + "/FDIforecast/in/tabla_COEFS_FDIm50_forecastEDO.csv",
    "FDIforecast_in_archivo_parametros_NINC_PRED_csv": rutaBase + "/FDIforecast/in/tabla_COEFS_NINC_PRED_EDO.csv",
    "FDIforecast_in_archivo_acumulado50dias_csv": rutaBase + "/FDIacumulado/out/FDIm50_EDO_completo.csv",
    "FDIforecast_out_archivo_FDIM50_forecast_csv": rutaBase + "/FDIforecast/out/FDIM50_forecast/FDIM50_forecast_FechA.csv",
    "FDIforecast_out_archivo_NINC_PRED_csv": rutaBase + "/FDIforecast/out/NINC_PRED/csv/NINC_PRED_FechA.csv",
    "FDIforecast_out_archivo_NINC_PRED_tif": rutaBase + "/FDIforecast/out/NINC_PRED/tif/NINC_PRED_FechA.tif",
    ### Mapa tif final
    "nombreArchivoTifFinalDiario": "",
    "tifNINCpaginaWEB": rutaBasePaginaWEB +
"/cartografia/capas/diarias/numero_incendios_esperados_por_estado/numero_incendios_esperados.tif",
    # Temporal
    "dirTMP": rutaBase + "/tmp"
}


#########################################################################################################################
#
# Funciones comunes
#
# suma dias a la fecha indicada, debe recibirse en formato AAAA-MM-DD
def agrega_quita_dias_a_fecha(fecha_inicial, accion, dias):
    fecha_tmp = datetime.datetime(int(fecha_inicial[0:4]), int(fecha_inicial[5:7]), int(fecha_inicial[8:10]))
    dias_delta = timedelta(days=dias)
    if accion == "AGREGAR":
        fecha_final = fecha_tmp + dias_delta
    if accion == "RESTAR":
        fecha_final = fecha_tmp - dias_delta
    fecha_final = ("%s" % fecha_final)[0:10]
```

```
    return fecha_final

def borra_archivo(shapefile):
    if arcpy.Exists(shapefile):
        arcpy.Delete_management(shapefile)
#########################################################################################################################
# Terminan funciones comunes


#########################################################################################################################
#
# Proceso FDI forecast 10 dias
#
# Convierte archivo CSV de parámetros del modelo a un diccionario
def FDIforecast_archivos_CSV_parametros_a_tabla(dicParam):
    print "FDIforecast_archivos_CSV_parametros_a_tabla..."
    d_tabla = {}
    l_dias = []
    archivo_entrada = dicParam["FDIforecast_in_archivo_parametros_forecast_csv"]
    f = open(archivo_entrada, 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            cve_estado = int(l_linea[0])
            p_a0 = l_linea[1]
            p_ar1 = l_linea[2]
            p_ar2 = l_linea[3]
            d_tabla[cve_estado] = {"p_a0": p_a0, "p_ar1": p_ar1, "p_ar2": p_ar2}
        i += 1
    # Cierra archivo
    f.close()

    if 'f' in locals():
        del f
    if 'lineas' in locals():
        del lineas

    return d_tabla


# Lee contenido de archivo CSV de 50 dias y lo pasa a un diccionario
def FDIforecast_archivos_CSV_50_dias_a_tabla(dicParam):
    print "FDIforecast_archivos_CSV_50_dias_a_tabla..."
    d_acumulado50dias = {}
    l_dias = []
    archivo_entrada = dicParam["FDIforecast_in_archivo_acumulado50dias_csv"]
    f = open(archivo_entrada, 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            fecha = l_linea[0]
            tmp = l_linea[1]
            if len(tmp) > 0:
                estado = int(l_linea[1])
            FDIm50_EDO = float(l_linea[2])
            if not fecha in l_dias:
                l_dias.append(fecha)
                d_acumulado50dias[fecha] = {}

            d_acumulado50dias[fecha][estado] = {"fecha": fecha, "estado": estado, "FDIm50_EDO": FDIm50_EDO}
        i += 1
    # Cierra archivo
    f.close()

    if 'f' in locals():
        del f
    if 'lineas' in locals():
        del lineas

    return d_acumulado50dias


# Acumula el valor para el estado indicado y del periodo de dias indicado
def FDIforecast_acumula_por_rango_de_fechas(fecha_inicial, fecha_final, estado, d_tabla, columna):
    promedio = 0
    suma = 0
    i = 0
    for fecha in d_tabla:
        if fecha >= fecha_inicial and fecha <= fecha_final:
            if fecha in d_tabla:
                if estado in d_tabla[fecha]:
                    suma += d_tabla[fecha][estado][columna]
                    i += 1
    if i > 0:
        promedio = suma / i
    return promedio


def FDIforecast_acumula_10_dias(d_acumulado50dias):
```

```
    print "FDIforecast_acumula_10_dias..."
    l_dias = []
    # determina fecha de inicio de proceso, la cual será 20 días posteriores a la fecha en que inician
    # los datos en el archivo de csv de entrada (acumulado de 50 dias

    # Determina la fecha del primer registro del archivo CSV acumulado de 50 dias
    l_fechas = d_acumulado50dias.keys()
    l_fechas.sort()
    fecha_primer_registro = l_fechas[0]
    fecha_inicial = fecha_primer_registro
    # Calcula la fecha en que puede iniciar el proceso (20 dias después del primer día)
    fecha_inicio_proceso = agrega_quita_dias_a_fecha(fecha_primer_registro, "AGREGAR", 20-1)

    # Se usa una matriz para almacenar el valor de AR2 por fecha y por estado
    # Eel propósito es que algún dia que no haya información para calcular AR2, se use el último calculado
    d_tmp_acumulado_AR2 = {}

    d_acumulado_10_dias = {}
    for fecha in l_fechas:
        for estado in d_acumulado50dias[fecha]:
            if fecha >= fecha_inicio_proceso:
                fecha_inicial_parametro_a = agrega_quita_dias_a_fecha(fecha, "RESTAR", 10-1)
                fecha_final_parametro_a = fecha
                fecha_inicial_parametro_b = agrega_quita_dias_a_fecha(fecha, "RESTAR", 20-1)
                fecha_final_parametro_b = agrega_quita_dias_a_fecha(fecha, "RESTAR", 10)

                acumulado_valor_a = FDIforecast_acumula_por_rango_de_fechas(fecha_inicial_parametro_a, fecha_final_parametro_a,
estado, d_acumulado50dias, 'FDIm50_EDO')
                acumulado_valor_b = FDIforecast_acumula_por_rango_de_fechas(fecha_inicial_parametro_b, fecha_final_parametro_b,
estado, d_acumulado50dias, 'FDIm50_EDO')
                # Si el valor de AR2=0, le pone el de día anterior
                if acumulado_valor_b == 0:
                    acumulado_valor_b = d_tmp_acumulado_AR2[estado]

                if not fecha in l_dias:
                    l_dias.append(fecha)
                    d_acumulado_10_dias[fecha] = {}
                if not estado in d_acumulado_10_dias[fecha]:
                    d_acumulado_10_dias[fecha][estado]={}

                d_acumulado_10_dias[fecha][estado]['AR1'] = acumulado_valor_a
                d_acumulado_10_dias[fecha][estado]['AR2'] = acumulado_valor_b

                # Almacena temporalmente el valor de AR2, por si el siguiente registro, el AR2=0,
                # entonces le ponemos el de este registro
                if not estado in d_tmp_acumulado_AR2:
                    d_tmp_acumulado_AR2[estado] = {}
                d_tmp_acumulado_AR2[estado] = acumulado_valor_b

    return d_acumulado_10_dias


def FDIforecast_aplica_ecuaciones_forecast(dicParam, d_acumulado):
    print "FDIforecast_aplica_ecuaciones_forecast..."
    # Carga archivo de parámetros a diccionario "d_parametros"
    f = open(dicParam["FDIforecast_in_archivo_parametros_forecast_csv"], 'r')
    lineas = f.readlines()
    i = 0
    d_parametros = {}
    d_acumulado_salida = {}
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            estado = int(l_linea[0])
            param_a0 = float(l_linea[1])
            param_AR1 = float(l_linea[2])
            param_AR2 = float(l_linea[3])
            d_parametros[estado] = {'a0': param_a0, 'AR1': param_AR1, 'AR2': param_AR2}
        i += 1
    # Aplica ecuacion a valores
    for fecha in d_acumulado:
        d_acumulado_fecha = {}
        for estado in d_acumulado[fecha]:
            promedio_AR1 = d_acumulado[fecha][estado]['AR1']
            promedio_AR2 = d_acumulado[fecha][estado]['AR2']
            a0 = d_parametros[estado]['a0']
            AR1 = d_parametros[estado]['AR1']
            AR2 = d_parametros[estado]['AR2']

            FDIM50_forecast = round(a0 + AR2 * promedio_AR2 + AR1 * promedio_AR1  ,2)
            d_acumulado_fecha[estado] = {'promedio_AR1': promedio_AR1, 'promedio_AR2': promedio_AR2, 'a0': a0, 'AR1': AR1, 'AR2':
AR2, 'FDIM50_forecast': FDIM50_forecast}
        d_acumulado_salida[fecha] = d_acumulado_fecha

    return d_acumulado_salida


def FDIforecast_genera_CSV_salida_forecast(dicParam, d_acumulado):
    print "FDIforecast_genera_CSV_salida_forecast..."
    # ordena fechas
    l_fechas = d_acumulado.keys()
    l_fechas.sort()

    #encabezados
```

```
    encabezado_csv = "fecha,estado,promedio_AR1,promedio_AR2,param_a0,param_AR1,param_AR2,FDIM50_forecast\n"

    # Archivo conmpleto
    archivo_csv_completo = dicParam["FDIforecast_out_archivo_FDIM50_forecast_csv"].replace("FechA", "completo")
    archivo_completo = open(archivo_csv_completo, 'w')
    archivo_completo.write(encabezado_csv)

    # recorre tabla de acuerdo al orden de la lista de fechas
    tmpFecha = ''
    for fecha_orden in l_fechas:
        archivo_diario_creado = 0

        archivo_nuevo = 0
        if fecha_orden != tmpFecha:
            if tmpFecha != '':
                # Cierra el ultimo archivo creado
                if archivo_diario_creado == 1:
                    archivo_diario.close()
            tmpFecha = fecha_orden
            archivo_nuevo = 1

        if archivo_nuevo == 1:
            # Genera un archivo de salida por día
            archivo_csv_diario = dicParam["FDIforecast_out_archivo_FDIM50_forecast_csv"].replace("FechA", fecha_orden)
            if not os.path.isfile(archivo_csv_diario):
                archivo_diario = open(archivo_csv_diario, 'w')
                # encabezados
                archivo_diario.write(encabezado_csv)
                archivo_diario_creado = 1

        row_dia = d_acumulado[fecha_orden]
        # ordena claves de uso_region
        l_orden_datos = row_dia.keys()
        l_orden_datos.sort()
        # agrega registro a archivo CSV
        for estado_orden in l_orden_datos:
            registro = "%s,%s,%s,%s,%s,%s,%s,%s\n" % (fecha_orden, estado_orden, row_dia[estado_orden]['promedio_AR1'],
row_dia[estado_orden]['promedio_AR2'], row_dia[estado_orden]['a0'], row_dia[estado_orden]['AR1'], row_dia[estado_orden]['AR2'],
row_dia[estado_orden]['FDIM50_forecast'])
            if archivo_diario_creado == 1:
                archivo_diario.write(registro)
            archivo_completo.write(registro)

        # Cierra el ultimo archivo creado
        if archivo_diario_creado == 1:
            archivo_diario.close()

    # Cierra el ultimo completo
    archivo_completo.close()


# Lee contenido de archivo CSV de valores FDImean y lo pasa a un diccionario
def FDIforecast_archivos_CSV_FDImean_a_tabla(dicParam):
    print "FDIforecast_archivos_CSV_FDImean_a_tabla..."
    d_tabla = {}
    l_dias = []
    archivo_entrada = dicParam["FDImean_out_csv"]
    f = open(archivo_entrada, 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            fecha = l_linea[0]
            l_fecha = fecha.split("/")
            fecha = l_fecha[2] + "-" + l_fecha[1] + "-" + l_fecha[0]
            estado = int(l_linea[1])
            fdimean = float(l_linea[4])

            if not fecha in l_dias:
                l_dias.append(fecha)
                d_tabla[fecha] = {}
            if not estado in d_tabla[fecha]:
                d_tabla[fecha][estado] = {}
            d_tabla[fecha][estado]['FDImean'] = fdimean
        i += 1
    # Cierra archivo
    f.close()

    if 'l_dias' in locals():
        del l_dias
    if 'f' in locals():
        del f
    if 'lineas' in locals():
        del lineas

    return d_tabla


# Obtiene el promedio de 10 dias segun la fecha indicada (9 dias atras mas el actual) de la matriz FDImen
def FDIforecast_promedio10diasAtrasFDImean(d_FDImean):
    print "FDIforecast_promedio10diasAtrasFDImean ..."
    l_dias = []
    d_acumulado_10_dias = {}
```

```
    llaves = d_FDImean.keys()
    llaves.sort()
    for fecha in llaves:
        for estado in d_FDImean[fecha]:
            fecha_inicial = agrega_quita_dias_a_fecha(fecha, "RESTAR", 10-1)
            fecha_final = fecha

            acumulado = FDIforecast_acumula_por_rango_de_fechas(fecha_inicial, fecha_final, estado, d_FDImean, 'FDImean')
            if not fecha in l_dias:
                l_dias.append(fecha)
                d_acumulado_10_dias[fecha] = {}
            if not estado in d_acumulado_10_dias[fecha]:
                d_acumulado_10_dias[fecha][estado]={}

            d_acumulado_10_dias[fecha][estado] = acumulado

    return d_acumulado_10_dias


def FDIforecast_aplica_ecuaciones_NINC_PRED(dicParam, d_acumulado, d_FDImean_prom10dias):
    print "FDIforecast_aplica_ecuaciones_NINC_PRED..."
    # Carga archivo de parámetros a diccionario "d_parametros"
    f = open(dicParam["FDIforecast_in_archivo_parametros_NINC_PRED_csv"], 'r')
    lineas = f.readlines()
    i = 0
    d_parametros = {}
    d_acumulado_salida = {}
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            estado = int(l_linea[0])
            param_a = float(l_linea[1])
            param_b = float(l_linea[2])
            FDI95 = float(l_linea[3])
            d_parametros[estado] = {'param_a': param_a, 'param_b': param_b, 'FDI95': FDI95}
        i += 1
    # Aplica ecuacion a valores
    for fecha in d_acumulado:
        d_acumulado_fecha = {}
        for estado in d_acumulado[fecha]:
            FDImean = d_FDImean_prom10dias[fecha][estado]
            param_FDI95 = d_parametros[estado]['FDI95']
            if FDImean > param_FDI95:
                param_a = d_parametros[estado]['param_a']
                param_b = d_parametros[estado]['param_b']
                if d_acumulado[fecha][estado]['FDIM50_forecast'] > 0:
                    FDIM50_forecast = d_acumulado[fecha][estado]['FDIM50_forecast'] / 100
                NINC_PRED = round(param_a * FDIM50_forecast ** param_b, 2)
            else:
                param_a = 0
                param_b = 0
                NINC_PRED = 0

            d_acumulado_fecha[estado] = {'FDImean': FDImean, 'FDIM50_forecast': d_acumulado[fecha][estado]['FDIM50_forecast'],
'param_a': param_a, 'param_b': param_b, 'param_FDI95': param_FDI95, 'NINC_PRED': NINC_PRED}
        d_acumulado_salida[fecha] = d_acumulado_fecha

    return d_acumulado_salida


def FDIforecast_genera_CSV_salida_NINC_PRED(dicParam, d_acumulado):
    print "FDIforecast_genera_CSV_salida_NINC_PRED..."
    # ordena fechas
    l_fechas = d_acumulado.keys()
    l_fechas.sort()

    #encabezados
    encabezado_csv = "fecha,estado,FDImean,FDIM50_forecast,param_a,param_b,param_FDI95,NINC_PRED\n"

    # Archivo completo
    archivo_csv_completo = dicParam["FDIforecast_out_archivo_NINC_PRED_csv"].replace("FechA", "completo")
    archivo_completo = open(archivo_csv_completo, 'w')
    archivo_completo.write(encabezado_csv)

    # recorre tabla de acuerdo al orden de la lista de fechas
    tmpFecha = ''
    for fecha_orden in l_fechas:
        archivo_diario_creado = 0

        archivo_nuevo = 0
        if fecha_orden != tmpFecha:
            if tmpFecha != '':
                # Cierra el ultimo archivo creado
                if archivo_diario_creado == 1:
                    archivo_diario.close()
            tmpFecha = fecha_orden
            archivo_nuevo = 1

        if archivo_nuevo == 1:
            # Genera un archivo de salida por día
            archivo_csv_diario = dicParam["FDIforecast_out_archivo_NINC_PRED_csv"].replace("FechA", fecha_orden)
            if not os.path.isfile(archivo_csv_diario):
                archivo_diario = open(archivo_csv_diario, 'w')
                # encabezados
```

```
                archivo_diario.write(encabezado_csv)
                archivo_diario_creado = 1

        row_dia = d_acumulado[fecha_orden]
        # ordena claves de uso_region
        l_orden_datos = row_dia.keys()
        l_orden_datos.sort()
        # agrega registro a archivo CSV
        for estado_orden in l_orden_datos:
            registro = "%s,%s,%s,%s,%s,%s,%s,%s\n" % (fecha_orden, estado_orden, row_dia[estado_orden]['FDImean'],
row_dia[estado_orden]['FDIM50_forecast'], row_dia[estado_orden]['param_a'], row_dia[estado_orden]['param_b'],
row_dia[estado_orden]['param_FDI95'], row_dia[estado_orden]['NINC_PRED'])
            if archivo_diario_creado == 1:
                archivo_diario.write(registro)
            archivo_completo.write(registro)

        # Cierra el ultimo archivo creado
        if archivo_diario_creado == 1:
            archivo_diario.close()

    # Cierra el ultimo completo
    archivo_completo.close()


def FDIforecast(dicParam):
    d_tabla_parametros = FDIforecast_archivos_CSV_parametros_a_tabla(dicParam)
    d_acumulado50dias = FDIforecast_archivos_CSV_50_dias_a_tabla(dicParam)
    d_acumulado_10_dias = FDIforecast_acumula_10_dias(d_acumulado50dias)
    d_acumulado_con_ecuacion_forecast = FDIforecast_aplica_ecuaciones_forecast(dicParam, d_acumulado_10_dias)
    FDIforecast_genera_CSV_salida_forecast(dicParam, d_acumulado_con_ecuacion_forecast)

    d_FDImean = FDIforecast_archivos_CSV_FDImean_a_tabla(dicParam)
    d_FDImean_prom10dias = FDIforecast_promedio10diasAtrasFDImean(d_FDImean)
    d_acumulado_con_ecuacion_NINC_PRED = FDIforecast_aplica_ecuaciones_NINC_PRED(dicParam, d_acumulado_con_ecuacion_forecast,
d_FDImean_prom10dias)
    FDIforecast_genera_CSV_salida_NINC_PRED(dicParam, d_acumulado_con_ecuacion_NINC_PRED)


#########################################################################################################################
#  Termina Proceso FDI forecast 10 dias


#########################################################################################################################
#
# Proceso FDI acumulado 50 dias
#
def FDIacumulado_archivos_CSV_a_tabla(dicParam, d_tabla):
    print "FDIacumulado_archivos_CSV_a_tabla..."
    l_dias = []
    archivo_entrada = dicParam["FDImean_out_csv"]
    f = open(archivo_entrada, 'r')
    lineas = f.readlines()
    i = 0
    for linea in lineas:
        # Ignora la primer linea porque es de encabezados
        if i > 0:
            l_linea = linea.split(",")
            fecha = l_linea[0]
            l_fecha = fecha.split("/")
            fecha = l_fecha[2] + "-" + l_fecha[1] + "-" + l_fecha[0]
            estado = int(l_linea[1])

            #fdimean = int(round(float(l_linea[4]), 0))
            fdimean = float(l_linea[4])

            if not fecha in l_dias:
                l_dias.append(fecha)
                d_tabla[fecha] = {}
            d_tabla[fecha][estado] = fdimean
        i += 1
    # Cierra archivo
    f.close()

    if 'l_dias' in locals():
        del l_dias
    if 'f' in locals():
        del f
    if 'lineas' in locals():
        del lineas

    return d_tabla

#
# Acumula el valor para el estado indicado y del periodo de dias indicado
#
def FDIacumulado_acumula_por_rango_de_fechas(fecha_inicial, fecha_final, estado, d_tabla):
    suma = 0.0
    i = 0
    for fecha in d_tabla:
        if fecha >= fecha_inicial and fecha <= fecha_final:
            if fecha in d_tabla:
                if estado in d_tabla[fecha]:
                    suma += d_tabla[fecha][estado]
                    i += 1
```

```
    # se retorna el promedio
    return suma/i


def FDIacumulado_genera_CSV_salida(dicParam, d_acumulado):
    print "FDIacumulado_genera_CSV_salida..."
    # ordena fechas
    l_fechas = d_acumulado.keys()
    l_fechas.sort()

    # encabezados
    encabezado_csv = "fecha,estado,FDIm50_EDO\n"
    # Archivo completo
    archivo_csv_completo = dicParam["FDIacumulado_out_FDIm50_csv"].replace("FechA", "completo")
    if not os.path.isfile(archivo_csv_completo):
        archivo_completo = open(archivo_csv_completo, 'w')
        archivo_completo.write(encabezado_csv)
    else:
        archivo_completo = open(archivo_csv_completo, 'a')

    # recorre tabla de acuerdo al orden de la lista de fechas
    tmpFecha = ''
    for fecha_orden in l_fechas:
        archivo_diario_creado = 0

        archivo_nuevo = 0
        if fecha_orden != tmpFecha:
            if tmpFecha != '':
                # Cierra el ultimo archivo creado
                if archivo_diario_creado == 1:
                    archivo_diario.close()
            tmpFecha = fecha_orden
            archivo_nuevo = 1

        if archivo_nuevo == 1:
            # Genera un archivo de salida por dia
            archivo_csv_diario = dicParam["FDIacumulado_out_FDIm50_csv"].replace("FechA", fecha_orden)
            if not os.path.isfile(archivo_csv_diario):
                archivo_diario = open(archivo_csv_diario, 'w')
                # encabezados
                archivo_diario.write(encabezado_csv)
                archivo_diario_creado = 1

        row_dia = d_acumulado[fecha_orden]
        # ordena claves
        l_orden_datos = row_dia.keys()
        l_orden_datos.sort()
        # agrega registro a archivo CSV
        for estado_orden in l_orden_datos:
            registro = "%s,%s,%s\n" % (fecha_orden, estado_orden, row_dia[estado_orden])
            if archivo_diario_creado == 1:
                archivo_diario.write(registro)
            archivo_completo.write(registro)

        # Cierra el ultimo archivo creado
        if archivo_diario_creado == 1:
            archivo_diario.close()

    # Cierra el ultimo completo
    archivo_completo.close()

#
# Genera tabla con el aculumado de 50 dias
#
def FDIacumulado_acumulado_50_dias(d_tabla, d_acumulado):
    print "FDIacumulado_acumulado_50_dias..."
    l_dias = []
    # Determina la fecha del primer registro del archivo CSV acumulado de 50 dias
    l_fechas = d_tabla.keys()
    l_fechas.sort()

    for fecha in l_fechas:
        for estado in d_tabla[fecha]:
            #if fecha >= todayYMD:
            fecha_final = fecha
            fecha_inicial = agrega_quita_dias_a_fecha(fecha_final, "RESTAR", 50-1)
            valor = FDIacumulado_acumula_por_rango_de_fechas(fecha_inicial, fecha_final, estado, d_tabla)

            if not fecha in l_dias:
                l_dias.append(fecha)
                d_acumulado[fecha] = {}
            d_acumulado[fecha][estado] = valor

    if 'l_fechas' in locals():
        del l_fechas

    return d_acumulado


def FDIacumulado(dicParam):
    print "FDIacumulado..."
    d_tabla = {}
    d_acumulado = {}
    d_tabla = FDIacumulado_archivos_CSV_a_tabla(dicParam, d_tabla)
    d_acumulado = FDIacumulado_acumulado_50_dias(d_tabla, d_acumulado)
```

```
    FDIacumulado_genera_CSV_salida(dicParam, d_acumulado)

    if 'd_tabla' in locals():
        del d_tabla
    if 'd_acumulado' in locals():
        del d_acumulado


#############################################################################################################################
#  Termina Proceso FDI acumulado 50 dias



#############################################################################################################################
#
# Proceso FDImean
#
def FDImean_combina(dicParam, archivoTIFF, img_combine_salida):
    # Crea cadena de fecha a partir del nombre del archivo LM
    archivo = archivoTIFF.split("/")[-1]
    tmp = archivo.split("FDI_")
    tmp = tmp[1].split("_Day.tif")
    tmp = tmp[0]
    anio = tmp[0:4]
    fecha = tmp[6:8] + '/' + tmp[4:6] + '/' + anio

    arcpy.gp.Combine_sa(dicParam["inTIFFestados"] + ";" + archivoTIFF, img_combine_salida)

    # Lee tabla de datos de la imagen combine
    # Los primeros 10 caracteres del nombre del tiff son los que le pone a la columna
    ltmnombreArchivoTIFF = archivoTIFF.split('/')[-1]
    nombreColTIFF = ltmnombreArchivoTIFF[0:10]
    rows = arcpy.SearchCursor(img_combine_salida, "", "", "OID; COUNT,"+nombreColTIFF+",ESTADOS", "ESTADOS,"+nombreColTIFF)
    i = 0
    dicSumasUsoReg = {}
    dicSumaCount = {}
    listaLlaves = []
    for row in rows:
        # Elimina la clave -1 = no data
        if row.ESTADOS > 0:
            i += 1

            if row.ESTADOS in dicSumasUsoReg:
                dicSumasUsoReg[row.ESTADOS] += row.COUNT * eval("row."+nombreColTIFF)
                dicSumaCount[row.ESTADOS] += row.COUNT
            else:
                listaLlaves.append(row.ESTADOS)
                dicSumasUsoReg[row.ESTADOS] = row.COUNT * eval("row."+nombreColTIFF)
                dicSumaCount[row.ESTADOS] = row.COUNT

    #  Agrega datos de este tiff al archivo CSV
    FDI_mean_csv = open(dicParam["FDImean_out_csv"], 'a')

    # ordena los datos antes de mandarlos al archivo de salida
    for llave in listaLlaves:
        # uso_reg,Suma(count * LM10H_2012), contador de registros con el mismo uso_reg, promedio_ponderado
        ren = "%s,%s,%s,%s,%s\n" % (fecha, llave, dicSumasUsoReg[llave], dicSumaCount[llave],
dicSumasUsoReg[llave]/dicSumaCount[llave])
        FDI_mean_csv.write(ren)

    # Cierra archivo
    FDI_mean_csv.close()


def FDImean(dicParam):
    print "FDImean..."
    i = 0
    ruta_entrada_TIFF = dicParam["FDImean_in_tifFDI"]
    archivosTIFF = glob.glob(ruta_entrada_TIFF + "/*.tif")
    for archivoTIFF in archivosTIFF:
        # Acciones a realizar solo en la primera vuelta
        if i == 0:
            # asigna nombre al archivo de salida
            csv_salida_combine_suma_ponderada = dicParam["FDImean_out_csv"]
            # crea archivos CSV
            if not os.path.isfile(csv_salida_combine_suma_ponderada):
                FDI_mean_csv = open(csv_salida_combine_suma_ponderada, 'w')
                # pone encabezados
                ren = "fecha,estado,suma(count * FDI),suma(count),FDImean\n"
                FDI_mean_csv.write(ren)
                FDI_mean_csv.close()

        archivoTIFF = archivoTIFF.replace("\\", "/")

        # Si ya existe el archivo de salida se brinca este archivo y no genera nada
        ruta_combine_salida = dicParam["FDImean_out_tifFDI"]
        tmp = archivoTIFF.replace('.tif', '_edos.tif')
        ltmp = tmp.split('/')
        nombre_archivo_combine = ltmp[-1]
        img_combine_salida = ruta_combine_salida + '/' + nombre_archivo_combine
        if not arcpy.Exists(img_combine_salida):
            FDImean_combina(dicParam, archivoTIFF, img_combine_salida)
            dicParam["FDImean_procesa_archivo_nuevo"] = 1
        i += 1
#############################################################################################################################
```

```
# Termina proceso FDImean



#########################################################################################################################
#
# Prepara archivo tif para publicación WEB
#
def obtieneUltimoArchivoCSVdiarioDisponible(dicParam):
    rutaArchivoDiario = dicParam["FDIforecast_out_archivo_NINC_PRED_csv"]
    nombreArchivoDiario = rutaArchivoDiario.split("/")[-1]
    rutaArchivosNINC_PRED = rutaArchivoDiario.replace("/" + nombreArchivoDiario, "")
    archivosNINC_PRED = glob.glob(rutaArchivosNINC_PRED + "/NINC_PRED_20*.csv")
    archivosNINC_PRED.sort()
    ultimoArchivo = archivosNINC_PRED.pop()
    ultimoArchivo = ultimoArchivo.replace("\\", "/")
    return ultimoArchivo

# Carga archivo CSV de NINC_PRED a diccionario
def csvNINC_PREDaDic(archivoCSV):
    print "csvNINC_PREDaDic..."
    dicSalida = {}
    if os.path.isfile(archivoCSV):
        f = open(archivoCSV, 'r')
        lineas = f.readlines()
        i = 0
        for linea in lineas:
            if i > 0:
                lLinea = linea.split(',')
                dicSalida[int(lLinea[1])] = float(lLinea[7])
            i += 1
        # Cierra archivo CSV
        f.close()
        if 'f' in locals():
            del f
        if 'lineas' in locals():
            del lineas
    else:
        print "No existe el archivo CSV: %s" % archivoCSV

    return dicSalida


def agregaColumnasAtiff(archivoTIFF):
    print "agregaColumnasAtiff..."
    try:
        arcpy.DeleteField_management(archivoTIFF, "NINCpred")
    except:
        pass
    arcpy.AddField_management(archivoTIFF, "NINCpred", "SHORT", "", "", "", "", "NULLABLE", "NON_REQUIRED", "")


def actualizaDatosEnTIFF(archivoTIFF, dicCSV):
    cursor = arcpy.UpdateCursor(archivoTIFF, "", "", "VALUE; NINCPRED;", "")
    for row in cursor:
        try:
            row.NINCPRED = 0
            if row.VALUE in dicCSV:
                row.NINCPRED = round(dicCSV[row.VALUE])
            cursor.updateRow(row)
        except:
            print "\n__ERROR: no se pudo actualizar la tabla del mapa con estos valores:"
            print "Mapa:", archivoTIFF
            print "ESTADO: ", row.VALUE
            print "row.NINCPRED: ", row.NINCPRED
            exit(0)

    if 'row' in locals():
        del row
    if 'cursor' in locals():
        del cursor


# Pasa el valor de la columna NINCPRED a la columna VALUE en TIFF
def cambiaValorDeNINCPREDaVALUE(dicParam, archivoTIFF):
    print "cambiaValorDeNINCPREDaVALUE..."
    nombreArchivoTiff = archivoTIFF.split("/")[-1]
    nombre_tmp_shp  = nombreArchivoTiff.replace("_join.tif", ".shp")
    archivo_tmp_shp = dicParam["dirTMP"] + "/" + nombre_tmp_shp
    archivo_tif_final = archivoTIFF.replace("_join.tif", ".tif")
    borra_archivo(archivo_tmp_shp)
    arcpy.RasterToPolygon_conversion(archivoTIFF, archivo_tmp_shp, "NO_SIMPLIFY", "NINCPRED")

    # Pone el cell size del archivo de salida igual al archivo original
    archivoTIFF_cellsize = archivoTIFF
    borra_archivo(archivo_tif_final)
    arcpy.PolygonToRaster_conversion(archivo_tmp_shp, "GRIDCODE", archivo_tif_final, "CELL_CENTER", "GRIDCODE", archivoTIFF_cellsize)

    # Guarda nombre de archivo tif final
    dicParam["nombreArchivoTifFinalDiario"] = archivo_tif_final

# hace join de CSV y TIF de estados por clave de estado
def joinCSVmapa(dicParam, dicCSV, fecha):
    print "joinCSVmapa..."
```

```
    # Genera copia de Tif de estados
    archivoTIFF_origen = dicParam["inTIFFestados"]
    archivoTIFF_destino = dicParam["FDIforecast_out_archivo_NINC_PRED_tif"].replace("FechA", fecha + "_join")
    borra_archivo(archivoTIFF_destino)
    arcpy.CopyRaster_management(archivoTIFF_origen, archivoTIFF_destino)
    agregaColumnasAtiff(archivoTIFF_destino)
    # Actualiza los datos en nuevo TIFF
    actualizaDatosEnTIFF(archivoTIFF_destino, dicCSV)
    cambiaValorDeNINCPREDaVALUE(dicParam, archivoTIFF_destino)
    borra_archivo(archivoTIFF_destino)


def generaMapa(dicParam):
    print "generaMapa..."
    # Busca último archivo CSV diario disponible
    ultimoArchivoCSVdiarioDisponible = obtieneUltimoArchivoCSVdiarioDisponible(dicParam)
    nombreDeUltimoArchivoCSVdiarioDisponible = ultimoArchivoCSVdiarioDisponible.split("/")[-1]
    fechaUltimoArchivoCSVdiarioDisponible = nombreDeUltimoArchivoCSVdiarioDisponible.split("_")[2].split(".")[0]
    dicCSV = csvNINC_PREDaDic(ultimoArchivoCSVdiarioDisponible)
    joinCSVmapa(dicParam, dicCSV, fechaUltimoArchivoCSVdiarioDisponible)

def publicaMapaEnWeb(dicParam):
    print "publicaMapaEnWeb..."
    archivo_entrada = dicParam["nombreArchivoTifFinalDiario"]
    archivo_salida = dicParam["tifNINCpaginaWEB"]
    borra_archivo(archivo_salida)
    arcpy.CopyRaster_management(archivo_entrada, archivo_salida)

#########################################################################################################################
# Termina: Prepara archivo tif para publicación WEB


#
# Proceso principal
#

# Genera archivo csv NINC_PRED
FDImean(dicParam)
if dicParam["FDImean_procesa_archivo_nuevo"] == 1:
    FDIacumulado(dicParam)
    FDIforecast(dicParam)

# Prepara archivo tif para publicación WEB
generaMapa(dicParam)
publicaMapaEnWeb(dicParam)

print "\nTerminado: %s %s" % (todayYMD, time.strftime("%H:%M:%S") )
```

## Referencias.


Vega Nieva et al. 2018. Anexo 1. Módulo de análisis de peligro y riesgo de peligro y de incendios en México:. Reporte de la tercera etapa del proyecto CONACYT-CONAFOR 252620 "Desarrollo de un Sistema de predicción de Peligro de Incendios Forestales para México".

Vega Nieva et al. 2018. Anexo 1.1. Módulo de análisis de peligro y riesgo de peligro y de incendios en México: mapa de riesgo de ocurrencia de incendio por factores humanos. Reporte de la tercera etapa del proyecto CONACYT-CONAFOR 252620 "Desarrollo de un Sistema de predicción de Peligro de Incendios Forestales para México".

Vega Nieva et al. 2018. Anexo 1.2. Módulo de análisis de peligro y riesgo de peligro y de incendios en México: cálculo del peligro de incendio y del número de incendios esperados por estado. Reporte de la tercera etapa del proyecto CONACYT-CONAFOR "Desarrollo de un Sistema de predicción de Peligro de Incendios Forestales para México".

Vega Nieva et al. 2018. Anexo 2. Código abierto del Sistema de Peligro de Incendios para México. Reporte de la tercera etapa del proyecto CONACYT-CONAFOR 252620 "Desarrollo de un Sistema de Peligro de Incendios para México".

Vega-Nieva, et al. 2018. Developing Models to Predict the Number of Fire Hotspots from an Accumulated Fuel Dryness Index by Vegetation Type and Region in Mexico. Forests 2018, 9, 190 Disponible en: http://forestales.ujed.mx/incendios/incendios/pdf/forests-09-00190.pdf