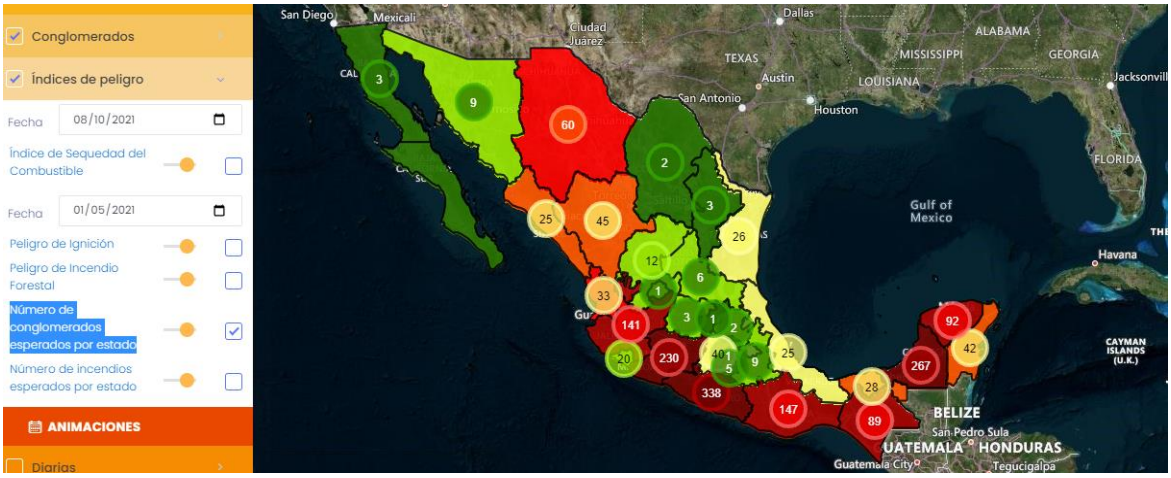
Anexo 1.1. Documentación y código del índice de pronóstico del número de conglomerados de puntos de calor esperados por estado.



Informe del Proyecto "Reforzamiento al Sistema Nacional de Predicción de Peligro de Incendios Forestales de México para el pronóstico de conglomerados y área quemada", con clave B-S-131553, convocatoria CONAFOR-2018-C02

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

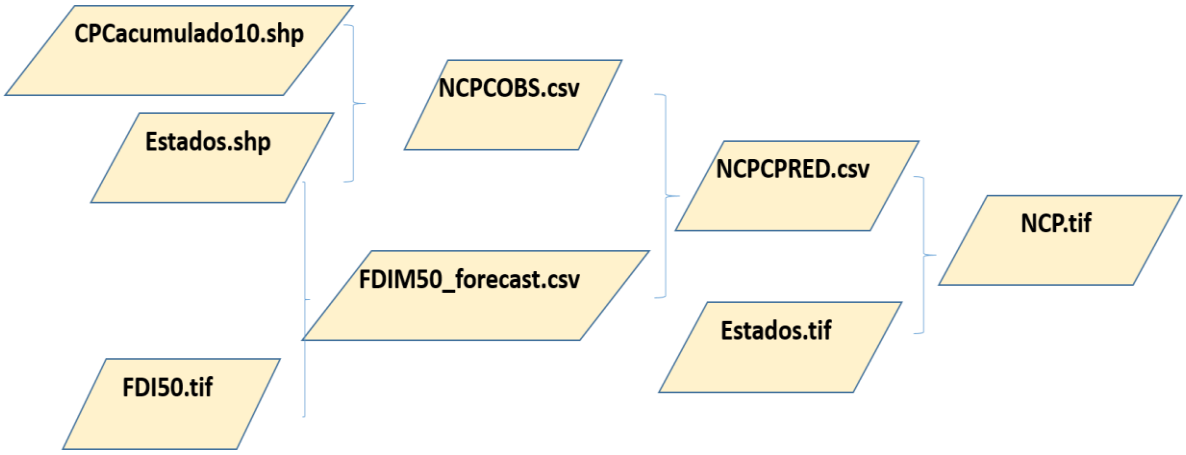
Autores: Dr. Daniel José Vega Nieva, Dr. Jaime Briseño Reyes (FCF-UJED).

RESUMEN.

En el presente documento se incluye y documenta el código del cálculo del número de conglomerados esperados por estado. En la primera sección se incluye un resumen de los subprocesos de este proceso de cálculo, describiendo los insumos y rutas de los mismos en el servidor del sistema de peligro. En la segunda sección se anexa en código para el cálculo del número de conglomerados esperados por estado.

1. **Proceso de cálculo del número de conglomerados esperados por estado.**

En el siguiente diagrama se resume el proceso de cálculo del número de conglomerados esperados por estado, señalando los archivos de entrada, resultados intermedios, y resultado final, el mapa de número de conglomerados esperados por estado, que se actualiza diariamente en el sistema de peligro.



Proceso de cálculo del número de conglomerados esperados por estado.

El cálculo comprende los siguientes pasos:

1- **Cálculo del número de conglomerados observados por estado (NCPCOBS.csv) en los últimos diez días.**
2- **Cálculo del número de conglomerados predichos por estado (NCPCPRED.csv)**
3- **Cálculo del mapa en formato .tif de número de conglomerados predichos por estado (NCPCPRED.tif)**

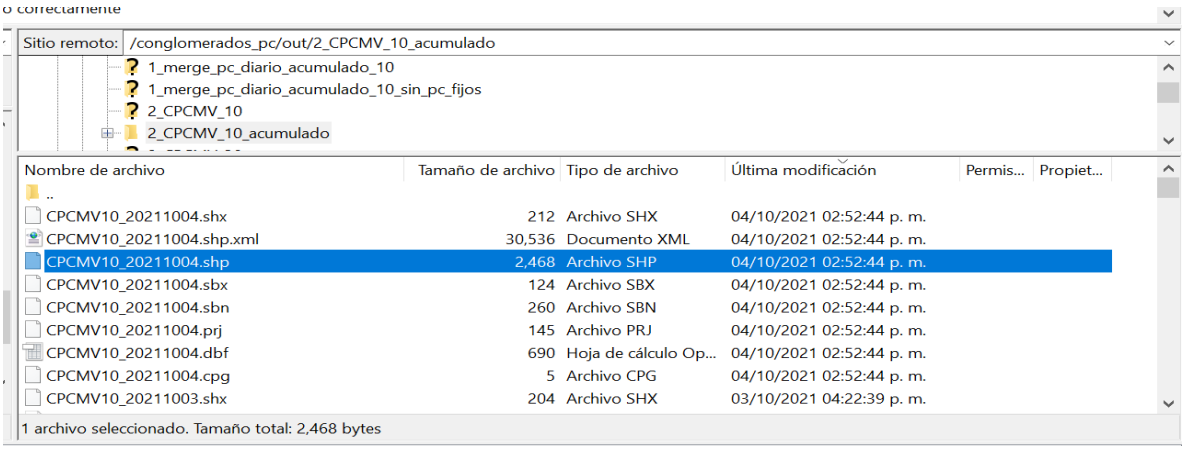
Se describen a continuación los procesos, incluyendo una descripción y rutas de los archivos de entrada y salida de cada subproceso.

1- **Cálculo del número de conglomerados observados por estado (NCPCOBS.csv) en los últimos diez días.**

Entradas:

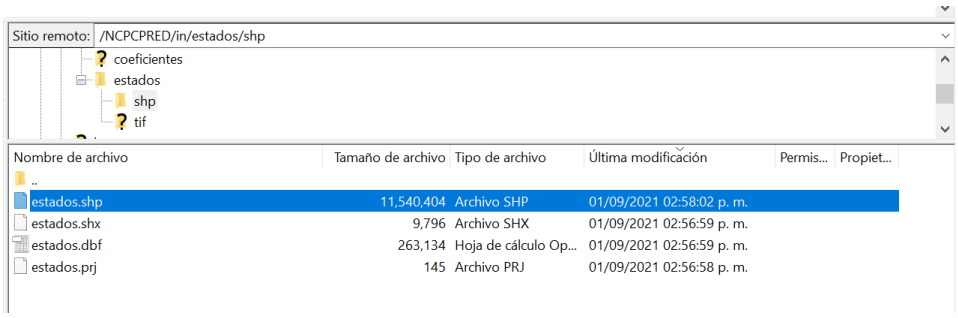
1.1. **CPCMV10_fecha.shp:** Archivo en formato .shp que contiene los perímetros de los conglomerados de puntos de calor MODIS y VIIRS observados en los últimos 10 días.

Se encuentra disponible en la ruta: **/conglomerados_pc/out/2_CPCMV_10_acumulado**



1.2. **Estados.shp:** Archivo en formato .shp que contiene el perímetro de los estados de la república. Se encuentra disponible en la ruta:

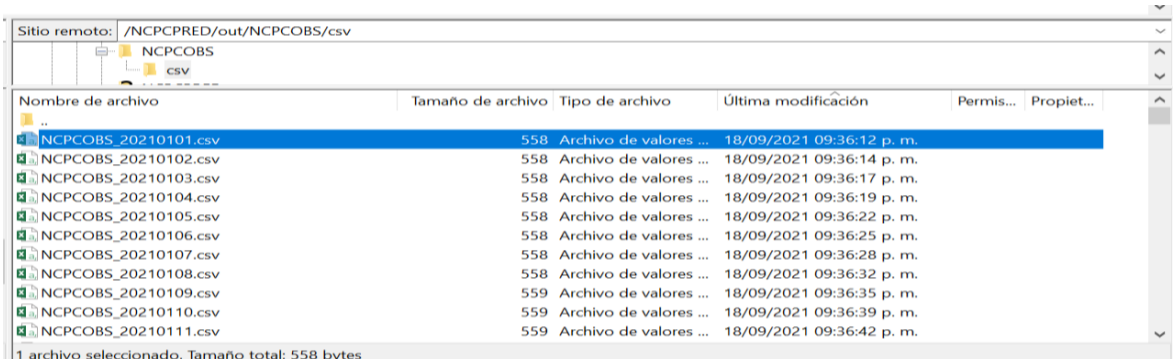
/NCPCPRED/in/estados/shp



Salidas:

NCPCOBS.csv: Archivo en formato .csv que contiene el número de conglomerados de puntos de calor observados en los últimos 10 días por estado. Se calcula realizando una intersección de la capa de conglomerados con la de estados, y un conteo del número de polígonos para cada ID de estado. El resultado se encuentra disponible en la ruta:

/NCPCPRED/out/NCPCOBS/csv



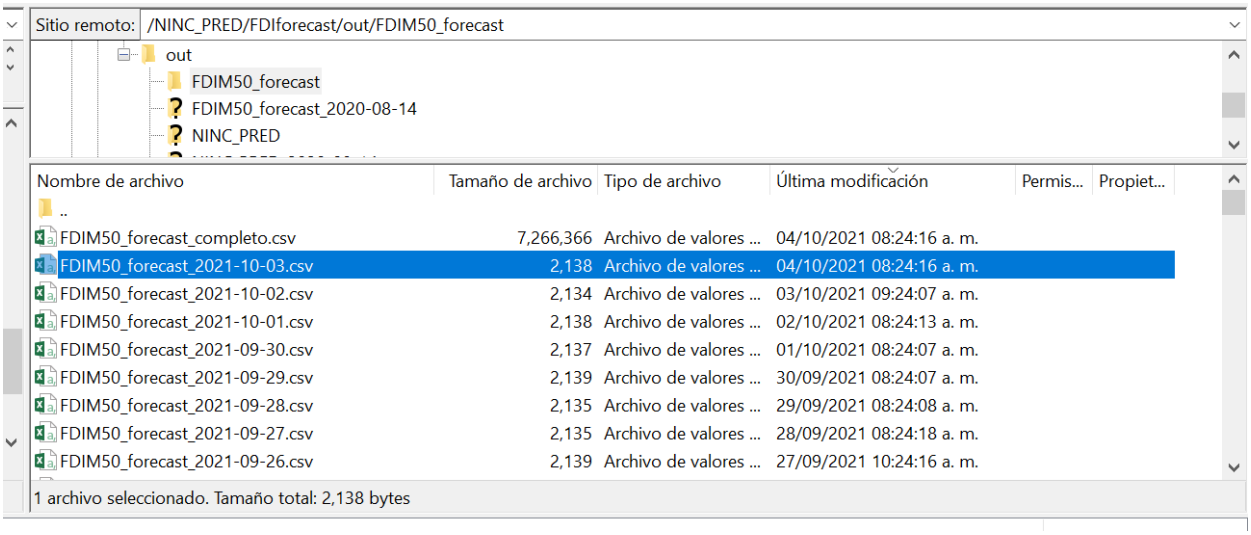
2- Cálculo del número de conglomerados predichos por estado (NCPCPRED.csv)

Entradas:

2.1. **NCPCOBS.csv.** Salida del proceso anterior.

2.2. **FDIM50_forecast.csv.** Archivo en formato .csv que contiene el pronóstico del FDI50 acumulado por estado. Se encuentra disponible en la ruta:

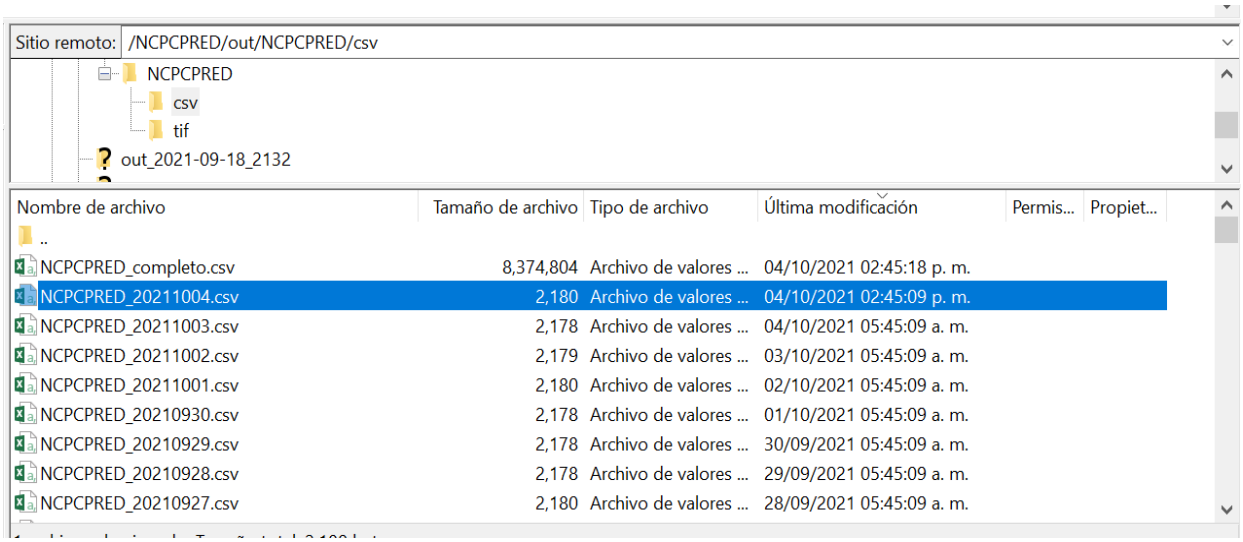
/NINC_PRED/FDIforecast/out/FDIM50_forecast



Salidas:

2.3. NCPCPRED.csv: Archivo en formato .csv que contiene el número de conglomerados de puntos de calor predicho por estado. Se calcula empleando los algoritmos descritos en el **Anexo 1-Informe técnico del desarrollo de un índice de pronóstico del número de conglomerados de puntos de calor esperados por estado.** Se encuentra disponible en la ruta:

/NCPCPRED/out/NCPCPRED/csv

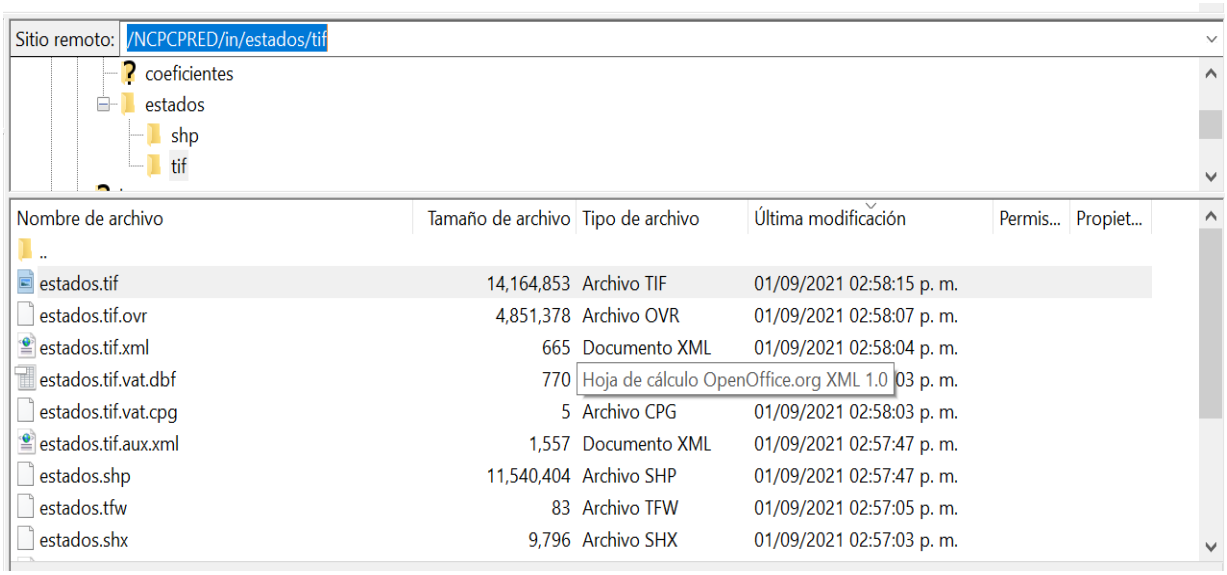


3- **Cálculo del mapa en formato .tif de número de conglomerados predichos por estado (NCPCPRED.tif)**

Entradas:

3.1. NCPCPRED.csv: Salida del proceso anterior.

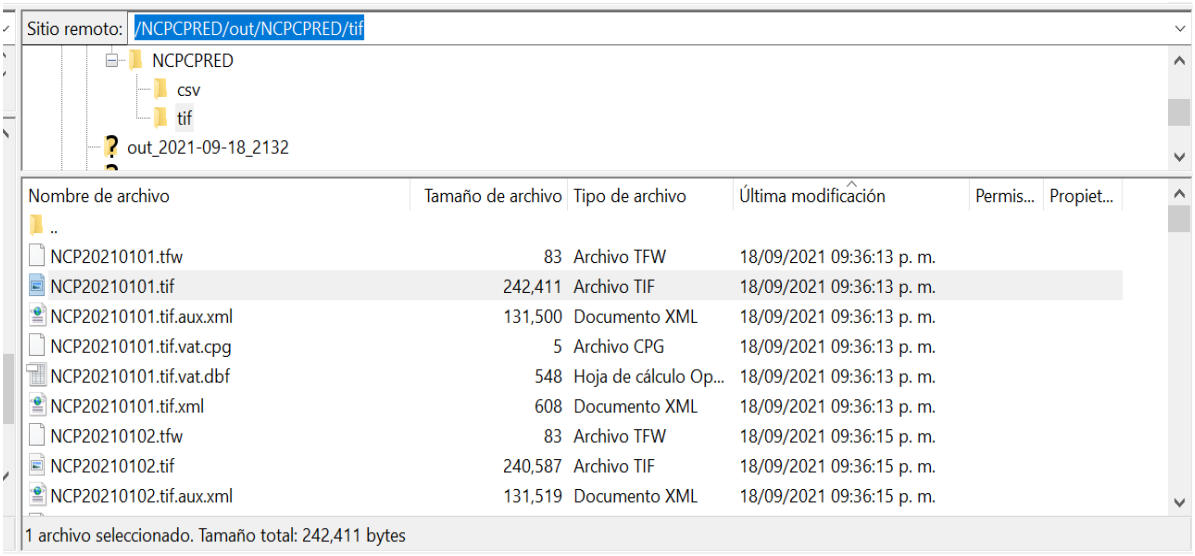
3.2. Estados.tif. Archivo en formato .tif que contiene los estados de la república. Se encuentra disponible en la ruta:**/NCPCPRED/in/estados/tif**



Salidas:

3.3. NCP.tif: Archivo en formato .tif que contiene el número de conglomerados de puntos de calor predicho por estado. Se encuentra disponible en la ruta:

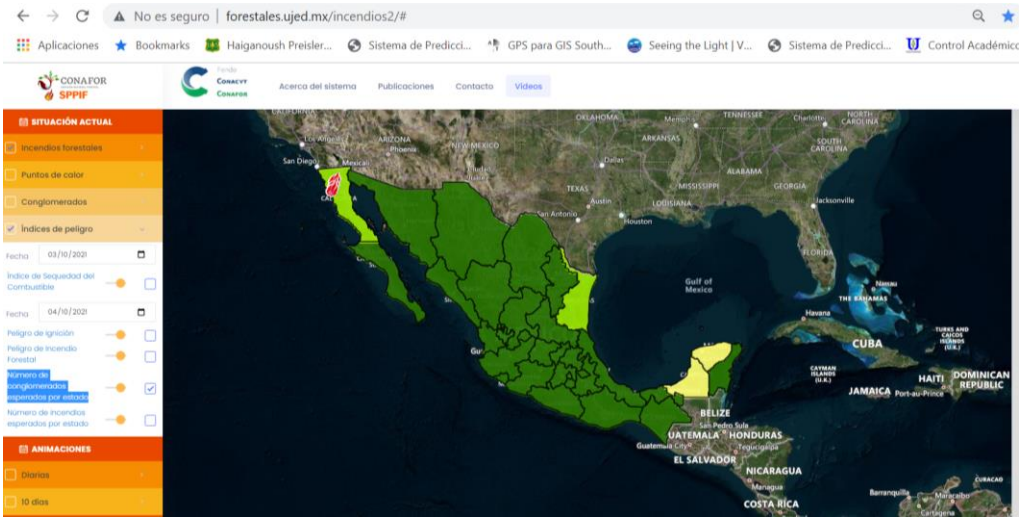
/NCPCPRED/out/NCPCPRED/tif



Cálculo y carga en el sistema de peligro.

El mapa de **NCP.tif** se calcula diariamente a las 8 am, a partir de la última observación disponible de conglomerados de puntos de calor MODIS y VIIRS, y en base al último pronóstico del índice de sequedad acumulada FDIM50forecast disponible.

Se muestra en el sistema de peligro en la sección **Situación Actual-> Índices de Peligro-> Número de conglomerados esperados por estado**



Número de conglomerados esperados por estado (04/10/2021).

2. Código del cálculo del Número de Conglomerados de Puntos de Calor esperados por estado

Lenguaje de programación: Python 2.7.14

```
# -*- coding: utf-8 -*-
#
# Sistema de Predicción de Peligro de Incendios Forestales de México (SPPIF)
#
# Cálculo de número de conglomerados de puntos de calor esperados por estado
#
# Autor: Universidad Juárez del Estado de Durango
#        Facultad de Ciencias Forestales
#        Jaime Briseño Reyes <jaime.briseno@gmail.com>
#
# Fecha: 24/ago/2021
#

import arcpy
from datetime import datetime
import os
import glob
import pandas as pd
import numpy as np

fechaHoraInicio = datetime.now()
print("Inicio: {}\n".format(fechaHoraInicio))

ruta_base = "D:/proyectos/incendios/proceso_diario/NCPCPRED"
ruta_CPCMV_10_acumulado =
"D:/proyectos/incendios/proceso_diario/conglomerados_pc/out/2_CPCMV_10_acumulad
o"
ruta_csv_NINC_PRED =
"D:/proyectos/incendios/proceso_diario/NINC_PRED/FDIforecast/out/NINC_PRED/csv"

dicParam = {
    # 1.1
    "ruta_CPCMV_10_acumulado": ruta_CPCMV_10_acumulado,
    "shp_estados": "{}/in/estados/shp/estados.shp".format(ruta_base),
    "csvs_NCPCOBS": "{}/out/NCPCOBS/csv".format(ruta_base),
    #1.2.1
    "ruta_csv_NINC_PRED": ruta_csv_NINC_PRED,
    "csv_coefs_NCP_EDO":
"{}/in/coeficientes/CoefsNCP_EDO160821.csv".format(ruta_base),
    "csvs_NCPCPRED": "{}/out/NCPCPRED/csv".format(ruta_base),
    "csv_NCPCPRED_completo":
"{}/out/NCPCPRED/csv/NCPCPRED_completo.csv".format(ruta_base),
    # 1.2.2 join tif estados y cvs de NCPCPRED
    "tif_estados": "{}/in/estados/tif/estados.tif".format(ruta_base),
    # 1.2.3 reclass de tif de estados por columna NCPCPRED
    "tifs_NCPCPRED": "{}/out/NCPCPRED/tif".format(ruta_base),

    "ruta_out": "{}/out".format(ruta_base),
    "ruta_tmp": "{}/tmp".format(ruta_base)
}

# Clase Principal
# ncoe = Número de Conglomerados Observados por Estado
class ncoe:
    dicParam = ""

    #
    # Método de inicialización
```

```
    #
    def __init__(self, dicParam):
        self.dicParam = dicParam
        self.ultimo_shp_CPCMV_10_acumulado =
self.obtieneUltimoShpCPCMV_10_acumulado()
        self.fecha_ultimo_shp_CPCMV_10_acumulado =
self.ultimo_shp_CPCMV_10_acumulado.split("/")[-1].split("_")[1].split(".")[0]
        self.shp_intersectCPCMV10estados = ""
        self.csv_NCPCOBS = ""
        self.csv_NCPCPRED = ""
        self.tif_NCPCPRED_salida = ""

        # Borra contenido de carpeta temporal
        self.borraArchivosDeCarpeta(self.dicParam['ruta_tmp'])

        # 1 NUMERO DE CONGLOMERADOS ESPERADOS POR ESTADO

        # 1.1 CALCULO DEL NUMERO DE CONGLOMERADOS OBSERVADOS POR ESTADO

        # 1.1.1 Intersect CPCMV10_acumulado y estados
        self.intersectCPCMV10estados()

        # 1.1.2 A la table dbf del intersect ICPCEDO210816.shp, calcularle por
el campo NUM_EDO el count (número) de polígonos
        self.calculaNumeroDeIncendiosPorEstado()

        # 1.2 CÁLCULO DE NUMERO DE CONGLOMERADOS PREDICHOS POR ESTADO (NCP):

        # 1.2.1 CALCULO NCPCPRED (Numero de conglomerados de puntos de calor
predichos)
        self.df_NCPCPRED = self.calculoNCPCPRED()

        # Re-construye archivo csv completo de NCPCPRED cada vez para evitar
que se corrompa o se repitan datos
        self.generaCsvNCPCPREDcompleto()

        # 1.2.2 Join a estados.tif (Unimos el value del tif de estados con el
número de estado de NCPCPRED.csv)
        # 1.2.3 Reclassify a tif de estados
        self.joinEstadosNCPCPRED()

    # Borra contenido de tmp
    def borraArchivosDeCarpeta(self, ruta):
        if os.path.isdir(ruta):
            archivos = glob.glob(ruta + "/*")
            for archivo in archivos:
                os.remove(archivo)

    def obtieneUltimoShpCPCMV_10_acumulado(self):
        shapes =
sorted(glob.glob("{}/*.shp".format(self.dicParam['ruta_CPCMV_10_acumulado'])))
        ultimo_shp = shapes[-1].replace("\\", "/")
        return ultimo_shp

    # 1 NUMERO DE CONGLOMERADOS ESPERADOS POR ESTADO


##############################################################################
######
    # 1.1 CALCULO DEL NUMERO DE CONGLOMERADOS OBSERVADOS POR ESTADO (NCPCOBS)

##############################################################################
######
```

```
    # 1.1.1 Intersect CPCMV10_acumulado y estados
    def intersectCPCMV10estados(self):
        print "Intersect de CSV con SHP de estados (intersectCPCMV10estados)
..."
        self.shp_intersectCPCMV10estados =
"{}/ICPCEDO{}.shp".format(self.dicParam['ruta_tmp'],
self.fecha_ultimo_shp_CPCMV_10_acumulado)
        arcpy.Intersect_analysis([self.ultimo_shp_CPCMV_10_acumulado,
self.dicParam['shp_estados']], self.shp_intersectCPCMV10estados, "ALL", "",
"INPUT")

    # 1.1.2 1.1.2 Calcula por el campo NUM_EDO el count (número) de polígonos a
la tabla dbf del intersect
    def calculaNumeroDeIncendiosPorEstado(self):
        print "calculaNumeroDeIncendiosPorEstado ..."
        dicIncendiosPorEstado = {}
        # Inicializa valores en el diccionario para todos los estados
        for i in range(1, 32+1):
            dicIncendiosPorEstado[i] = 0
        # Acumula incendios por estado
        with arcpy.da.SearchCursor(self.shp_intersectCPCMV10estados, ['FID',
'NUM_EDO']) as cursor:
            for row in cursor:
                dicIncendiosPorEstado[int(row[1])] =
dicIncendiosPorEstado[int(row[1])] + 1

        # Genera archivo CSV de NCPCOBS
        print " - Generando archivo CVS de NCPCOBS..."
        self.csv_NCPCOBS =
"{}/NCPCOBS_{}.csv".format(self.dicParam['csvs_NCPCOBS'],
self.fecha_ultimo_shp_CPCMV_10_acumulado)
        fecha_csv = "{}-{}-
{}".format(self.fecha_ultimo_shp_CPCMV_10_acumulado[0:4],
self.fecha_ultimo_shp_CPCMV_10_acumulado[4:6],
self.fecha_ultimo_shp_CPCMV_10_acumulado[6:8])
        f = open(self.csv_NCPCOBS, 'w')
        f.write('fecha,cve_edo,NCPCOBS\n')
        for i in range(1, 32+1):
            f.write("{},{},{}\n".format(fecha_csv, i,
dicIncendiosPorEstado[i]))
        f.close()



##########################################################################
######
    # 1.2 CÁLCULO DE NUMERO DE CONGLOMERADOS PREDICHOS POR ESTADO (NCP):

##########################################################################
######

    #
    # 1.2.1 CALCULO NCPCPRED (Numero de conglomerados de puntos de calor
predichos)
    #
    def calculoNCPCPRED(self):
        print "Calculo  de NCPCPRED (calculoNCPCPRED)..."
        # Obtiene datos de ultimo csv de NINC_PRED
        df_NINC_PRED = self.obtiene_ultimo_NINC_PRED()
        # Carga CSV de NCPCOBS (Generado en paso 1)
        df_NCPCOBS = pd.read_csv(self.csv_NCPCOBS)
        # Carga CSV de coeficientes
```

```
        df_coefs = pd.read_csv(self.dicParam['csv_coefs_NCP_EDO'])

        # Une dataframes requeridos para calculo de NCPCPRED
        # tmp = NINC_PRED + NCPCOBS
        tmp = df_NINC_PRED.merge(df_NCPCOBS, left_on=['fecha', 'estado'],
right_on=['fecha', 'cve_edo'])
        # df_NCPCPRED = tmp + coefs
        df_NCPCPRED = tmp.merge(df_coefs, on="estado")

        # Calcula NCPCPRED
        #
        # Inicializa en cero la columna NCPCPRED
        df_NCPCPRED['NCPCPRED'] = 0
        # Cambia el valor de 'NCPCPRED' únicamente en los renglones donde
'FDImean' > 'FDI95'
        df_NCPCPRED.loc[df_NCPCPRED['FDImean'] > df_NCPCPRED['FDI95'],
['NCPCPRED']] = \
            np.round(df_NCPCPRED['a'] * np.exp(df_NCPCPRED['b'] *
df_NCPCPRED['FDIM50_forecast']) + (df_NCPCPRED['c'] * df_NCPCPRED['NCPCOBS']),
0)

        # Genera archivo CSV de NCPCPRED
        print " - Generando archivo CVS de NCPCPRED..."
        self.csv_NCPCPRED =
"{}/NCPCPRED_{}.csv".format(self.dicParam['csvs_NCPCPRED'],
self.fecha_ultimo_shp_CPCMV_10_acumulado)
        # Elimina columnas no requeridas del dataframe generado con los merge
        df_NCPCPRED = df_NCPCPRED.drop(['param_a', 'param_b', 'param_FDI95',
'NINC_PRED', 'cve_edo'], axis='columns')
        # Genera CSV a partir de dataframe de NCPCPRED
        df_NCPCPRED.to_csv(self.csv_NCPCPRED, index=False)

        return df_NCPCPRED

    def obtiene_ultimo_NINC_PRED(self):
        print " - Obtiene datos del último NINC_PRED (obtiene_ultimo_NINC_PRED)
..."
        csvs =
sorted(glob.glob("{}/NINC_PRED_20*.csv".format(self.dicParam['ruta_csv_NINC_PRE
D'])))
        ultimo_csv = csvs[-1]
        df_ultimo_csv = pd.read_csv(ultimo_csv)
        # Al último csv le pone la fecha del último shape en
shp_CPCMV_10_acumulado para poder hacer el merge
        df_ultimo_csv['fecha'] = "{}-{}-
{}".format(self.fecha_ultimo_shp_CPCMV_10_acumulado[0:4],
self.fecha_ultimo_shp_CPCMV_10_acumulado[4:6],
self.fecha_ultimo_shp_CPCMV_10_acumulado[6:8])
        return df_ultimo_csv


    # Re-construye archivo csv completo de NCPCPRED cada vez para evitar que se
corrompa o se repitan datos
    def generaCsvNCPCPREDcompleto(self):
        print "Genera CSV historico de NCPCPRED (generaCsvNCPCPREDcompleto)...
"
        csv_files =
sorted(glob.glob("{}/NCPCPRED_20*.csv".format(self.dicParam['csvs_NCPCPRED'])))
        i = 0
        for csv_file in csv_files:
            csv_file = csv_file.replace("\\", "/")
            df = pd.read_csv(csv_file)
            if i == 0:
```

```
                df.to_csv(self.dicParam['csv_NCPCPRED_completo'], index=False)
            else:
                df.to_csv(self.dicParam['csv_NCPCPRED_completo'], index=False,
mode='a', header=False)
            i = i + 1

    #
    # 1.2.2 Join a estados.tif (Unimos el value del tif de estados con el
número de estado de NCPCPRED.csv)
    # 1.2.3 Reclassify a tif de estados
    #
    def joinEstadosNCPCPRED(self):
        print "Join y Reclasify a tif de estados (joinEstadosNCPCPRED) ..."
        dic_NCPCPRED = self.df_NCPCPRED.to_dict()
        # Crea copia de tif de estados en espacio temporal
        tif_tmp = "{}/estados_tmp.tif".format(self.dicParam['ruta_tmp'])
        if arcpy.Exists(self.dicParam['tif_estados']):
            if arcpy.Exists(tif_tmp):
                arcpy.Delete_management(tif_tmp)
            arcpy.CopyRaster_management(self.dicParam['tif_estados'], tif_tmp)

            # Crea columna en archivo Raster de estados
            arcpy.AddField_management(tif_tmp, "NCPCPRED", "SHORT", "4", "",
"", "", "NULLABLE", "NON_REQUIRED", "")

            # Actualiza columna "NCPCPRED" del tif de estados
            cursor = arcpy.UpdateCursor(tif_tmp, "", "", "VALUE; NCPCPRED;",
"VALUE;")
            for row in cursor:
                row.NCPCPRED = int(dic_NCPCPRED['NCPCPRED'][row.VALUE - 1])
                cursor.updateRow(row)

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

            # Reclassify por columna "NCPCPRED"
            self.tif_NCPCPRED_salida =
"{}/NCP{}.tif".format(self.dicParam['tifs_NCPCPRED'],
self.fecha_ultimo_shp_CPCMV_10_acumulado)
            if arcpy.Exists(self.tif_NCPCPRED_salida):
                arcpy.Delete_management(self.tif_NCPCPRED_salida)
            arcpy.gp.Reclassify_sa(tif_tmp, "NCPCPRED", "0 0;NODATA 9999",
self.tif_NCPCPRED_salida, "DATA")

################################################################################
#############

# Instancia objeto
obj = ncoe(dicParam)


# Fecha/Hora actual
from datetime import datetime
fechaHoraTermino = datetime.now()
print("\nTermino: {}".format(fechaHoraTermino))
print("Tiempo: {}".format(fechaHoraTermino - fechaHoraInicio))
```