INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

DEPARTAMENTO DE ELECTRÓNICA, SISTEMAS E INFORMÁTICA
MAESTRÍA EN DISEÑO ELECTRÓNICO

MSC2526A – PROGRAMACIÓN PARA ANÁLISIS DE DATOS
PRIMAVERA 2021

ITESO, Universidad Jesuita de Guadalajara

REQUERIMIENTO DE AGUA EN CULTIVO TOMATE ROJO

PROYECTO FINAL
Presenta:

Ángel Adán Baltazar Ortiz
se710985, adan.baltazar.ortiz@gmail.com

Tlaquepaque, Jalisco. Mayo de 2021.

RESUMEN

El presente trabajo tiene como objetivo principal diseñar una herramienta para efecto de consulta del cálculo del requerimiento de agua en el cultivo de tomate rojo, buscando impactar en la planificación y optimización de riegos para pequeños y medianos productores ubicados en la zona sur de Jalisco.

De forma particular, se fusiona una base de datos existente proveniente de la estación climatológica Ciudad Guzman afiliada a la red Conagua (temperatura del aire, humedad relativa del aire, presión atmosférica, velocidad del viento, lluvia acumulada, etc.) con variables no censadas directamente en campo, como es el caso de la radiación solar, a partir de parámetros existentes como horas sol diarias.

Se implementa el lenguaje de programación Python para dos propósitos principalmente: manipulación y preparación de los datos climatológicos y la construcción del modelo FAO Penman-Monteith para determinar el requerimiento de agua de un cultivo de referencia. Bajo estas dos premisas se construye un modelo para calcular la evapotranspiración del cultivo, que a su vez permite realizar análisis de consumos de uso de agua agrícola para las diferentes etapas fenológicas de la planta bajo diferentes circunstancias climatológicas a lo largo del año.

Finalmente, se presentan los resultados estimados del requerimiento de agua para el cultivo de tomate rojo contemplando escenarios entre los años 2019 y 2020.

TABLA DE CONTENIDOS

1. OBJETIVO DE INVESTIGACIÓN

Resumen: En esta sección se presenta una introducción al trabajo producido por el análisis de datos meteorológicos para llevar a cabo la estimación de agua requerida en el cultivo de tomate rojo. Se mencionan como antecedentes algunos casos de estudio derivados de la aplicación del modelo FAO (Organización de Alimento y Agricultura, por sus siglas en inglés) Penman-Monteith [1] en el cálculo de la evapotranspiración de un cultivo. Dicho objeto de estudio tiene como finalidad exponer una herramienta digital destinada a los productores de pequeña y mediana escala de tomate rojo ubicados en la zona sur de Jalisco, la cual les permita planificar y optimizar el uso del agua agrícola, buscando impactos a mediano-largo plazo de mejoras en los rendimientos del cultivo.

1.1. Introducción

El siguiente documento presenta la recolección y un análisis de bases de datos climatológicas de la Comisión Nacional del Agua (Conagua) [2], para llevar a cabo la estimación del requerimiento de agua necesario a lo largo del cultivo de tomate rojo (Solanum Lycopersicum) representativo a la zona Sur de Jalisco. El cálculo se llevará a cabo a través del método FAO Penman-Monteith el cual es reconocido como el único modelo estándar para el cálculo computacional de la evapotranspiración proveniente de datos meteorológicos.

1.2. Antecedentes

Múltiples estudios derivados del método de FAO Penman-Monteith para cálculo de evapotranspiración han dado lugar como objeto de estudio en el cultivo de tomate rojo, abordando temáticas tales como:

1.2.1 Cosecha de fruta de tomate y evapotranspiración en el sur de Serbia

El agua es un recurso limitado, y este estudio se encuentra relacionado al uso racional del agua con tecnologías de crecimiento intensivo de tomate. Esta es una investigación de 2 años la cuál es producto de un procedimiento biológico a través de un campo experimental con las condiciones de riego de tomate hibrido Amati F1, en un tipo de suelo aluvial en el valle del sur de Morava cerca de Nish. Los experimentos fueron ajustados en un cuadro completamente aleatorio con cuatro réplicas, dónde 3 variantes de riegos fueron involucradas: potencial de agua del suelo (SWP, por sus siglas en inglés) 20 kPa, 30 kPa, y 40 kPa junto con él control de no riegos. Observando ambas investigaciones a través de los años, el mayor cultivo de fruta fue logrado en la variante de SWP 30 kPa, mientras que en la variante con mayor (SWP 20 kPa) y menor (SWP 40 kPa) humedad del suelo el cultivo de la fruta disminuyo. La mayor cosecha de tomate se observó cuando el promedio del consumo de agua para potencial de evapotranspiración (ETP, por sus siglas en inglés) acumulo 584 mm, de manera que este valor puede ser considerado como la demanda del tomate para agua en el sur de Serbia. El valor más alto de eficiencia del uso de agua (WUE, por sus siglas en inglés) y eficiencia del uso de agua en riego (IWUE, por sus siglas en inglés) para el tomate se alcanzaron en la variante de SWP 30 kPa, el cual significa el consumo racional del agua fue habilitado en este valor SWP. Los resultados de este estudio muestran que, usando tensiómetros, el régimen de los riegos de tomate puede ser exitosamente mantenido en SWP 30 kPa, en un tipo de suelo aluvial en el sur de Serbia. [3]

1.2.2 Determinación de evapotranspiración de tomate y calabaza usando lisímetros en el centro de Arabia Saudita.

Nueve lisímetros fueron utilizados para cultivar alfalfa como cultivo de referencia, y tomate calabacita como cultivo experimental para medir su evapotranspiración. Los coeficientes de cultivo fueron estimados para el tomate y la calabaza en las diferentes etapas fenológicas, basados en la medición de evapotranspiración de lisímetro y con referencia a la evapotranspiración de alfalfa. Los valores estimados del coeficiente de cultivo para tomate en la cuarta etapa (inicial, desarrollo de cultivo, reproducción y maduración) fueron 0.28, 0.8, 0.96 y 0.75 y los valores correspondientes el coeficiente del cultivo de calabaza fueron 0.56, 0.72, 0.96 y 0.63. Ambos valores de coeficiente del cultivo fueron mucho menores que los sugeridos por la FAO en áreas áridas. Cinco métodos climatológicos fueron seleccionados para la

estimación de evapotranspiración (FAO-Penman, Blaney-Criddle, Jensen-Haise, Evapotranspiración en Bandeja y Thornthwaite). Un intento realizado fue desarrollar la relación entre la evapotranspiración medida por los lisímetros y los métodos de estimación climatológica. El rendimiento de los métodos de estimación climatológica como de medición de evapotranspiración fueron evaluados utilizando la desviación entre los valores, la raíz media del error cuadrático y el coeficiente de determinación. La comparación entre varios métodos de estimación indirecta de y medición de evapotranspiración revelaron que el método evaporación en bandeja para tomate y el método FAO –Penman para calabaza producen las mejores estimaciones como implementación de uso diario [4].

1.2.3 Modelado de evapotranspiración del tomate y respuesta de la cosecha a salinidad usando diferentes reducciones de funciones macroscópicas

La respuesta de las plantas a la salinidad puede ser utilizado para administrar los riegos con agua salobre. En este estudio se evaluaron los efectos de riego con agua salobre en tomate, proponiendo un nuevo modelo para describir la evaporación relativa de la planta y el cultivo relativo como una función de la conductividad eléctrica de un suelo saturado, y comparando las estimaciones de este nuevo modelo con otros cuatro modelos tales como: modelo lineal de Mass y Hoffman, modelo no lineal Genuchten, Dirksen, Homaee. Un invernadero experimental fue tratado con 5 tipos de agua de la llave de conductividad eléctrica 0.6, agua salobre de conductividad eléctrica de 2 y 3, agua con osmosis revertida de conductividad eléctrica de 4 y 6 dSm^{-1}. Los resultados mostraron que el nuevo modelo fue más simple y superior en la estimación de la respuesta de la planta a la conductividad eléctrica. El tomate es un cultivo sensible a la salinidad, y los riegos con aguas salobres adversamente tienen influencia en el crecimiento y evapotranspiración [5].

1.3. Justificación

El agua es un recurso limitado en la agricultura, por ello el impacto ambiental de este trabajo ha sido ligado al uso racional del agua que pudiese estarse incurriendo de forma implícita en los riegos del cultivo de tomate rojo, buscando así ralentizar la inyección de dióxido de carbono en la atmosfera. Uno de los grandes pilares para poder determinar de manera confiable una proyección de producción de un cultivo, es mediante la optimización del uso de agua como requerimiento de la planta. Aunado a los dos puntos anteriores, una de las razones económicas principales por las que se encuentra en constante lucha el productor de tomate rojo, es poder lograr una estabilización en el modelo de producción-venta Vs gastos de operación a lo largo del ciclo del cultivo.

1.4. Problema

Una de las grandes problemáticas que enfrenta el pequeño y mediano productor de tomate rojo a cielo abierto y/o de macro túnel en la región sur de Jalisco, es la planificación del riego basado en el cálculo del consumo de agua requerida por la planta de acuerdo con su etapa fenológica en las diferentes condiciones climáticas a lo largo del año.

1.5. Hipótesis

El tomate rojo que se cultiva en pequeña y mediana escala es un producto con enfoque de venta a intermediarios, en el cual los precios pueden estar variando en el mercado con una frecuencia semanal,

denotados por una gran amplitud entre los precios picos máximos y mínimos haciendo un proceso de difícil estimación de ventas. Por ello lograr una constancia en los niveles de producción en cada ciclo a lo largo del año, permite al productor anticipar los puntos bajos de producción-venta para así poder minimizar de forma gradual sus costos de operación e impactar en menor medida al medio ambiente.

1.6. Objetivos

1.6.1 Objetivo General

Diseñar una herramienta para efecto de consulta o calculo en el requerimiento de agua de la planta de tomate rojo enfocado al diseño y optimización de riegos en la zona sur de Jalisco.

1.6.2 Objetivos Específicos

A continuación, se definen los objetivos específicos para el cultivo de tomate rojo:

- Fusionar bases de datos existentes provenientes de las estaciones climatológicas del sur de Jalisco para obtención de las variables: radiación solar, temperatura del aire, humedad del aire y velocidad del viento.
- Creación de herramienta de SW para análisis y procesamiento de datos en el cálculo de requerimiento de agua basado en el modelo FAO Penman-Monteith.
- Análisis de consumos en el uso de agua agrícola.

1.7. Novedad científica, tecnológica o aportación

Soluciones tecnológicas de punta para análisis del consumo de agua en cultivos agrícolas en la actualidad son implementaciones con altos costos de inversión inicial difíciles de acceder para aquellos productores de pequeña y mediana escala con proyección de venta local. La aportación de dicho trabajo es presentar una herramienta tecnológica de libre acceso destinado a los productores locales de la zona sur de Jalisco para consultar o calcular el requerimiento de agua en el cultivo de tomate rojo.

2. OBTENCIÓN DE LOS DATOS

Resumen: En esta sección se presenta la fuente de datos climatológicos provistos por Conagua, así como una descripción de las principales variables a utilizar como requerimiento para el cálculo de evapotranspiración del tomate rojo por el método de FAO Penman-Monteith. Se hace mención sobre la estación del clima que provee dichos parámetros, la cual tendrá como representación a la zona sur de Jalisco y como se encuentra estructurada dicha base de datos.

2.1. Proceso de obtención de los datos

Los datos climatológicos representativos a la zona sur de Jalisco se recabaron a través de la unidad del observatorio meteorológico de Ciudad Guzmán (19.59216667, -103.59075) ubicado en el municipio de Zapotlán el Grande, el cual forma parte de una cadena de estaciones climatológicas denominada Organismo de Cuenca Lerma Santiago Pacifico pertenecientes al sistema meteorológico de Conagua. La base de datos con la que cuenta dicho plantel se presenta en dos grandes categorías: físico y digital. El archivo físico comprende un registro del año 2018 y previos a este, mientras que el archivo digital comprende un registro del año 2019 a la fecha actual (febrero 2021), siendo este último bloque con el cual se procederá a trabajar.

El archivo en formato Excel provisto por la estación meteorológica de Ciudad Guzman Conagua, es un compendio de los datos climatológicos de la zona registrados a lo largo de todo un año, el cual delimita cada uno de los meses a través de los tabs dentro del mismo archivo con su respectivo nombre del mes. Al interior, se puede observar que cuenta con un mayor número de registros/variables extras a las fundamentales (descritas anteriormente), las cuales son parte del monitoreo diario del centro meteorológico, por ejemplo: Niebla, Roció, Lluvia, Helada, Calima, Humo, Halo, Arco Iris, Chubascos, Truenos, Relámpagos, Temperatura Eléctrica, Eficiencia, Dirección.

A continuación, se describe de forma general los parámetros climatológicos requeridos para el cálculo de evapotranspiración por el modelo FAO Penman-Monteith:

2.1.1 Radiación Solar

El proceso de evapotranspiración está determinado por la cantidad de energía disponible para vaporizar agua. La radiación solar es la mayor fuente de energía y es capaz de cambiar grandes cantidades de agua líquida en vapor de agua. Debido a las diferencias de la posición del Sol, el potencial de radiación difiere en las diferentes latitudes y en las diferentes estaciones del año. La actual radiación solar que alcanza la evaporación de la superficie depende de la turbulencia de la atmósfera y la presencia de nubes las cuales reflejan y absorben la mayor parte de la radiación. Cuando se evalúa el efecto de la radiación solar en la evapotranspiración, se debería tener en mente que no toda la energía disponible es utilizada para vaporizar agua, si no que parte de la energía solar es utilizada para calentar la atmósfera y el perfil del suelo también [6].

2.1.2 Temperatura del Aire

La radiación solar absorbida por la atmósfera y el calor emitido por la tierra incrementa la temperatura del aire. La sensibilidad del calor que rodea las transferencias de energía de aire a el cultivo ejerce una influencia en el control de la velocidad de evapotranspiración en climas soleados y cálidos, mientras que la pérdida de agua por evapotranspiración es mucho mayor en climas nubosos y fríos [6].

2.1.3 Humedad del Aire

La fuente de energía del sol y el aire que rodea son la principal fuerza de vaporización del agua, la diferencia entre la presión del vapor de agua en la superficie de evapotranspiración y el aire que lo envuelve es el factor que determina el vapor que se remueve. Los campos bien suministrados de agua con riegos en áreas secas y calientes consumen grandes cantidades de agua debido a la abundancia de energía y el poder de desecación de la atmósfera. En regiones tropicales húmedas a pesar de la alta energía de

entrada, la alta humedad del aire reducirá la demanda de evapotranspiración. En tales ambientes el aire se encuentra saturado, de manera que la menor adición de agua puede ser guardada y de aquí que la velocidad de evapotranspiración es menor que en las regiones áridas [6].

2.1.4 Velocidad del Viento

El proceso de remover vapor depende de grandes cantidades de viento y turbulencia de aire las cuales transfieren grandes cantidades de aire sobre la superficie a evaporar. Cuando el agua vaporiza, el aire por encima de la superficie que se evapora se convierte gradualmente en vapor de agua saturada. Si dicho aire no es continuamente reemplazado por aire seco, la fuerza resultante para remover el vapor de agua y la velocidad de transpiración se decrementan [6].

2.2. Conclusiones

La base de datos provista por la estación meteorológica de Ciudad Guzmán Jalisco se encuentra en formato digital y de forma estructurada, lo cual facilitara el análisis de estos en los siguientes pasos. Es necesario continuar con la labor de búsqueda de las estaciones meteorológicas aledañas para extrapolar datos de diversas fuentes y así lograr definir un referente a la zona Sur de Jalisco [6].

3. PREPARACIÓN DE LOS DATOS

Resumen: En esta sección se presenta el proceso de transformación de datos a partir del registro anualizado que ofrece la base meteorológica de Ciudad Guzman afiliada a Conagua, el cual se presenta en formato digital. El proceso de preparación de datos es un módulo de Python el cual contempla las funcionalidades: cargado de archivo anual Conagua, manipulación de columnas, estimación de radiación solar (método de temperaturas o método horas sol al día), y exportación de archivo en formato CSV (valores separados por comas, según sus siglas en inglés) para cada uno de los meses seleccionados.

3.1. Proceso de preparación de los datos

El módulo de Python “data_preparation” [7] propuesto se compone de las siguientes tres funciones:

- Función para preparar datos Conagua: data_req_for_et0
- Función para estimación de radiación solar por horas sol: rad_solar_from_sun_hours
- Función para estimación de radiación solar por temperaturas: rad_solar_from_temp

Es importante mencionar que las últimas dos funciones mencionadas anteriormente dependen de la librería $PyET_O$ [7], que si bien es capaz de calcular ET_O (evaporación de referencia) pero que no será implementado como parte de nuestro modelos de análisis, provee numerosas funciones para la estimación de datos meteorológicos faltantes del cual nos apoyaremos para obtener la radiación solar no provista por la base meteorológica.

3.1.1 data_req_for_et0 ()

Esta función es la encargada de preparar los datos requeridos para cálculo de ET_O. A continuación, se describen los parámetros involucrados:

```
#############################################################################################################
# Funcion para generar los archivos correspondientes a los meses especificados como requerimiento para calculo ET0
def data_req_for_et0(conagua_year_data, month_list, year, latitude, longitude, altitude, coastal, export_file):
    """
    Parametros
    ----------
    Entrada
        conagua_year_data : str
            Ruta del archivo anualizado Conagua
        month_list,  : list
            Lista de los meses a preparar
        year : int
            Año de referencia
        latitude : float
            Latitud de base meteorologica (grados decimales)
        longitude : float
            Longitud de base meteorologica (grados decimales)
        altitude : list
            Altitud sobre el nivel del mar (meotrs)
        coastal : boolean
            True  -> zona costera
            False -> zona fuera de la costa
        export_file : boolean
            True  -> guardar archivo en ../output/
            False -> se despliega en consola los datos preparados
    ----------
    Salida
        return : int
            0
    """
#############################################################################################################
```

3.1.1.1 Cargado de archivo anualizado Conagua

Seleccionamos el archivo Conagua anualizado al contexto de pandas mediante un dataframe, posteriormente se realiza una conversión a minúsculas para dichos títulos y hacemos un renombre de las columnas que vienen actualmente en acrónimos para un mejor entendimiento de estas.

```
#########################################################################################################################
#display("data_req_for_et0 %s-%s" %(month,year))
# Asinamos la ruta del archivo con datos anualizados Conagua a una variable
xls = pd.ExcelFile(conagua_year_data)

# Ciclo for para iterar sobre la lista de meses
for month in month_list:
# Creación del dataframe a partir del archivo excel cargado usando el tab del mes
conagua_df = pd.read_excel(xls, month)

# Renombrado de encabezado de columnas de mayusculas a minusculas
conagua_df.columns = [col.lower() for col in conagua_df]

# Renombrado de encabezados de columnas principales para eliminar sus acronimos
conagua_df.rename(columns={
                           't.s': 'termometro seco',
                           't.h': 'termometro humedo',
                           't.v': 'tension de vapor',
                           'max': 'temperatura maxima',
                           'min': 'temperatura minima',
                           'p.r': 'punto de rocio',
                           'h.r': 'humedad relativa',
                           'max.1': 'humedad maxima',
                           'min.1': 'humedad minima',
                           'psen':'presion de la estacion',
                           'max.2': 'presion de la estacion maxima',
                           'min.2': 'presion de la estacion minima',
                           'prnm':'presion residual al nivel del mar',
                           'max.3': 'presion residual al nivel del mar maxima',
                           'min.3': 'presion residual al nivel del mar minima',
                           'max.4': 'evaporacion maxima',
                           'min.4': 'evaporacion minima',
                           'direcc': 'direccion dominante',
                           'vel' : 'velocidad dominante',
                           'direcc.1' : 'direcciom maxima',
                           'vel.1' : 'velocidad maxima',
                           'media' : 'velocidad media',
                           'p.1': 'pluviometro'
}, inplace=True)
#########################################################################################################################
```

3.1.1.2 Eliminación de columnas no requeridas

Se hace una comparativa entre el total de los nombres de las columnas del dataframe original y la lista que contiene los nombres de las columnas que deseamos preservar. De esta forma retiramos las variables que no tienen objeto de análisis en futuros pasos y procedemos a sobre escribir nuestro dataframe original. Se llevará a cabo un renombramiento de columnas de acuerdo con la nomenclatura sugerida por la librería de Python ET_O [8], utilizada en la construcción de nuestro modelo.

```
#################################################################################################################
# Definicion de lista que contiene las columnas dentro de conagua_df
columns_conagua_df = list(conagua_df.columns)

#Definimos la lista que contiene el nombre de las columnas preservar dentro de conagua_df
columns_remaining = [
                            'dia',
                            'tension de vapor',
                            'temperatura maxima',
                            'temperatura minima',
                            'humedad relativa',
                            'humedad minima',
                            'humedad maxima',
                            'inten',
                            'velocidad dominante',
                            'presion de la estacion',
                            'pluviometro'
]

# A traves de la funcion set realizamos la substraccion
# Cada elemento de columns_conagua_df se buscara dentro de columns_remaining
# Si no se encuentra coincidencia alguna esta se guarda como elemento de columns_to_drop
columns_to_drop_set = set(columns_conagua_df)-set(columns_remaining)
columns_to_drop = sorted(columns_to_drop_set, key = lambda k : columns_conagua_df.index(k))

# Removemos columns_to_drop de conagua_df y guardamos en conagua_df
conagua_df.drop(columns=columns_to_drop,inplace=True)

# https://eto.readthedocs.io/en/latest/package_references.html
# Renombramiento de columnas necesarias para calculacion de ETo
conagua_df.rename(columns={
                            'tension de vapor': 'e_a',
                            'temperatura maxima': 'T_max',
                            'temperatura minima': 'T_min',
                            'humedad relativa': 'RH_mean',
                            'humedad minima': 'RH_min',
                            'humedad maxima': 'RH_max',
                            'inten': 'n_sun',
                            'velocidad dominante': 'U_z',
                            'presion de la estacion': 'P',
                            'pluviometro': 'rain_mm'
}, inplace=True)
#################################################################################################################
```

3.1.1.3 Conversión de cero a NaN e imputación

Puesto que la columna de indexado del dataframe original no corresponde uno a uno a los días de captura, crearemos una copia de la columna día para luego definirla como el conjunto de valores índice del dataframe. A través de la librería calendar de Python obtenemos el número de días correspondientes al mes a evaluar y mediante este valor procedemos a recortar nuestro dataframe original. Esto nos permite descartar aquellos renglones que contienen datos no representativos. Cualquier valor en cero se convierte a un valor NaN el cual será remplazado con el dato más próximo del día anterior o posterior y garantizar que todos los días del mes completo cuentan con un valor apropiado.

```
#########################################################################################################
# Generamos una nueva columna la cual tiene como proposito hacer una copia directa
# de la columna dia a dia_index para ser usada como indice del dataframe
conagua_df['dia_index'] = ''
conagua_df['dia_index'] = conagua_df['dia']

# Definimos el index en conagua_df para la columna dia_index
conagua_df.set_index('dia_index', inplace = True)

# Calcular dias disponibles en el mes para acotar los contenidos extras a la tabla de referencia Conagua
# Obtencion de numeros de dias en el mes a evaluar
monthrange_result = monthrange(year,month_list.index(month)+1)
weekday_of_first_day_of_month = monthrange_result[0]
number_of_days_in_month = monthrange_result[1]

# Recortamos nuestro daframe al contexto de dias dentro del mes a evaluar
conagua_df = conagua_df.iloc[1:number_of_days_in_month+1]

# Remplazamor cualquier 0 tipo int/string por Nan
conagua_df[conagua_df.columns] = conagua_df[conagua_df.columns].replace({'0':np.nan, 0:np.nan})

# Obtenemos informacion del total de valores nulos presentes en conagua_df
conagua_isnull = conagua_df.isnull().sum()

# Lista de columnas con valores NaN presentes
conagua_columns_isnull = list(conagua_isnull.index.values.tolist())

# Definicion de lista vacia con informacion de las columnas a remover que tengan un valor NaN
conagua_columns_nan = []

# Checamos que columnas cuentan con valores nulos y lo agregamos a conagua_columns_dropna
for i in range(0,len(conagua_isnull)):
    if conagua_isnull[conagua_columns_isnull[i]] > 0:
        conagua_columns_nan.append(conagua_columns_isnull[i])

# Realizamos la imputacion de aquellos valores NaN en la columna tomando el valor anterior
for each in conagua_columns_nan:
    # Para el caso de la columna lluvia, buscamos dejar en 0 aquellos campos vacios
    if(each == 'rain_mm'):
        conagua_df[each].fillna(0, inplace = True)
    else:
        conagua_df[each].fillna(method = 'ffill', inplace = True)
        conagua_df[each].fillna(method = 'bfill', inplace = True)
#########################################################################################################
```

3.1.1.4 Creación de columna fecha y formato de tiempo

Se crea la columna fecha la cual se construye a través de la concatenación de las variables año y mes que recibe como parámetro la función data_req_for_et0. Una vez depositados los valores de fecha tipo "string" pasamos a formato serie de tiempo dicha columna.

```
#########################################################################################################
# Creacion de la columna fecha en conagua_df
conagua_df['date'] = ''

# Iteramos cada uno de los valores de la columna dia para construir el valor correspondiente a su fecha
# partir de concatenar año/mes/dia en la columna previamente insertada en conagua_df
for i in range(conagua_df['dia'][1],len(conagua_df['dia'])+1):
    conagua_df['date'][i] = str(year)+'-'+str(month_list.index(month)+1)+'-'+str(conagua_df['dia'][i])

#Conversión de la columna "date" al formato de serie de tiempo
conagua_df['date'] = pd.to_datetime(conagua_df['date'])
#########################################################################################################
```

3.1.1.5 Inspección de Temperaturas

Para garantizar que el cálculo de radiación solar (estimación basado en temperaturas) sea correcto y no se tengan errores por parte del módulo "math" de Python, checaremos para cada uno de los días

registrados que el valor de la temperatura máxima sea mayor al valor de la temperatura mínima. En caso contrario se procede a realizar el intercambio de valores.

```
#############################################################################################################
# Hacemos una inspeccion para la seccion de temperaturas diarias
# garantizando que la T_max sea mayor que la T_min
# En caso de no cumplirse se invertiran los valores
# y evitar problemas en el apartado del calculo de radiacion solar.
for day in conagua_df['dia']:
    #print(day,conagua_df['T_min'][day],conagua_df['T_max'][day])
    if (conagua_df['T_max'][day] < conagua_df['T_min'][day]):
        temp_t_max = conagua_df['T_max'][day]
        temp_t_min = conagua_df['T_min'][day]
        conagua_df['T_max'][day] = temp_t_min
        conagua_df['T_min'][day] = temp_t_max
#############################################################################################################
```

3.1.1.6 Columna y cálculo de radiación solar

Insertamos una nueva columna a nuestro dataframe para depositar el valor calculado de la radiación solar mediante la función rad_solar_from_temp/rad_solar_from_sun_hours el cual se explicará en el siguiente subtema. Las columnas referentes a tensión de vapor y presión de la estación de la base de datos de Conagua se encuentran denotados con el prefijo de kilo por lo que es necesario llevarlos a su base unitaria para cumplir el requerimiento de librería Python ET_O.

```
#############################################################################################################
# Anadimos para cada registro de la tabla su radiacion solar a partir de temperaturas
for day in conagua_df['dia']:
    conagua_df['R_s'][day] = rad_solar_from_sun_hours(year,month_list.index(month)+1,day,latitude,longitud
e,altitude,conagua_df['n_sun'][day])
    #conagua_df['R_s'][day] = rad_solar_from_temp(year,month_list.index(month)+1,day,latitude,longitude,al
titude,coastal,conagua_df['T_min'][day],conagua_df['T_max'][day])

# Arreglamos cualquier error producido por datos tipo flotante en columna de radiacion solar
conagua_df['R_s'] = conagua_df['R_s'].astype(float, errors = 'raise')

# Conversion de Hecto Pascales a Kilo Pascales en tension de vapor
#1 hectopascal [hPa] = 0.1 kilopascal [kPa]
conagua_df['e_a'] = conagua_df['e_a']*0.1

# Conversion de Hecto Pascales a Kilo Pascales en presion de la estacion
conagua_df['P'] = conagua_df['P']*0.1
#############################################################################################################
```

3.1.1.7 Guardado/Visualización datos preparados

Previo a guardar o visualizar en consola nuestro dataframe preparado ET_O (según el valor asignado a la variable export_file), eliminamos la columna de día y reasignamos como índice del dataframe a la columna fecha. Para estandarizar todos los valores tipo flotantes al interior del dataframe realizamos un redondeo a dos decimales.

```
#########################################################################################################################
# Eliminacion de la columna dia
conagua_df.drop(columns=['dia'],inplace=True)

# Seteando el index para la columna dia
conagua_df.set_index('date', inplace = True)

# Redondeamos todos los valores de los parametros a 2 decimales
conagua_df[conagua_df.columns] = conagua_df[conagua_df.columns].round(2)

# Guardado del Archivo en formato CSV
if(export_file == True):
    conagua_df.to_csv('../output/preparation/conagua_' + str(year) + '_' + month.lower() + '_req_for_eto.c
sv')
# Si no se requiere guardado, se visualiza el reordenamiento de datos para ETo
elif(export_file == False):
    display("Datos Conagua %s-%s preparados para ETo" %(month,year))
    #display("Paso >> Guardado/Visualización datos preparados ETo >> conagua_df")
    display(conagua_df)
#########################################################################################################################
```

3.1.2 rad_solar_from_temp ()

La base matemática de esta función radica en la fórmula de Hargreaves y Samani [9] como se muestra a continuación:

$$R_S = K_{RS}\sqrt{(Tmax - Tmin)}R_a$$

Donde:

- Ra, radiación extraterrestrial [$MJm^{-2}d^{-1}$]
- Tmax, temperatura máxima del aire [°C]
- Tmin, temperatura mínima del aire [°C]
- k_{RS}, coeficiente de ajuste (0.16… 0.19)

Dicha función recibe los argumentos de entrada tales como: fecha para analizar, posición geográfica, referencia costera, y temperaturas. Se obtiene como salida el valor estimado de radiación solar. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################
def rad_solar_from_sun_hours(year, month, day, latitude, longitude, altitude, sun_hours):
    """
    Parametros
    ----------
    Entrada
        year : int
            Año a evaluar
        month,  : int
            Mes a evaluar
        day : int
            Dia a evaluar
        latitude : float
            Latitud de base meteorologica (grados decimales)
        longitude : float
            Longitud de base meteorologica (grados decimales)
        altitude : list
            Altitud sobre el nivel del mar (metros)
        coastal : boolean
            True  -> zona costera
            False -> zona fuera de la costa
        temp_min : float
            Temperatura minima (grados centigrados)
        temp_max : float
            Temperatura maxima (grados centigrados)
    ----------
    Salida
        rad_sol_from_t : float
            Radiacion Solar (MJ m-2 day-1)
    """
#########################################################################################################
```

3.1.3 rad_solar_from_sun_hours ()

El cálculo de la radiación solar de entrada R_S (radiación que impacta en el plano horizontal disperso en la atmosfera) a partir de la duración relativa de horas sol. Si los datos de radiación no se encuentran disponibles, este método es el preferido para calcular la radiación solar.

Dicha función recibe los argumentos de entrada tales como: fecha para analizar, posición geográfica, referencia costera, y horas sol. Se obtiene como salida el valor estimado de radiación solar. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################################
def rad_solar_from_sun_hours(year, month, day, latitude, longitude, altitude, sun_hours):
    """
    Parametros
    ----------
    Entrada
        year : int
            Año a evaluar
        month,  : int
            Mes a evaluar
        day : int
            Dia a evaluar
        latitude : float
            Latitud de base meteorologica (grados decimales)
        longitude : float
            Longitud de base meteorologica (grados decimales)
        altitude : list
            Altitud sobre el nivel del mar (metros)
        coastal : boolean
            True  -> zona costera
            False -> zona fuera de la costa
        sun_hours : float
            Horas sul durante el dia (horas)
    ----------
    Salida
        rad_sol_from_sun_hours : float
            Radiacion Solar (MJ m-2 day-1)
    """
#########################################################################################################################
```

3.2. Conclusiones

Uno de los puntos clave en la preparación de los datos fue hacer una verificación de los valores de temperatura registrados por Conagua, ya que, si el valor máximo de temperatura no correspondía realmente a un valor mayor al registrado en el campo de la temperatura mínima, el cálculo de radiación se vería afectado de forma matemática por la ecuación de Hargreaves y Samani [9].

Se presento un caso extraordinario en el cual uno de los días registrados en el formato de Conagua, involucro el uso del carácter alfanumérico conocido como guion medio en cada una de las columnas. Al ser utilizado el dato por la función “data_req_for_et0” dichos valores se tomaron como tipo “string” lo cual rompe con la estructura de análisis del código implementado generando errores del tipo de comparaciones no permitidas. Por ahora la solución temporal consistió en hacer una inspección rápida del archivo si se observan este tipo de errores y procederá a remplazar manualmente con campos completamente vacíos o valores en cero.

Para efecto práctico, el método de cálculo de radiación solar por horas sol tiende a ser más confiable que el método de cálculo de radiación solar por temperaturas tal cual también lo indica la bibliografía. Esto se logró comprobar al realizar la comparación de los datos obtenidos por el visor interactivo de NASA predicción de recursos y energías del mundo [10] (POWER, por sus siglas en inglés) consultada para el mismo punto geográfico de la base meteorológica.

4. EXPLORACIÓN DE LOS DATOS

Resumen: En esta sección se presenta el proceso de exploración de los datos Conagua. Se proponen tres perspectivas de visualización en el plano del tiempo: extendido, duración del cultivo, y por rango de fechas. Dicho modulo genera y exporta los gráficos en formato de imagen, así como un archivo que será consumido por el módulo de Python “eto_base”.

4.1. Proceso de exploración de los datos

Previo a consumir los datos preparados Conagua en nuestro modelo para cálculo de ET_C (evapotranspiración o requerimiento de agua del cultivo), nos hemos dado a la tarea de hacer exploraciones graficas con tres perspectivas de tiempo:

1. A lo largo del tiempo contenido en el dataframe Conagua.
2. En la ventana de tiempo del periodo del cultivo tomando como referencia el día del trasplante del cultivo. Utilizaremos como referencia los requerimientos del cultivo tomate estipulados por la FAO [10].
3. En un rango de fechas establecido por día inicial y día final.

Para un entendimiento más profundo de los datos se han propuesto las siguientes funciones en el módulo de Python denominado "data_exploration" [13] las cuales se describen a continuación.

4.1.1 concat_conagua_month ()

Esta función permite la concatenación de los archivos mensuales Conagua preparados por el módulo de Python "data_preparation" el cual está diseñado para almacenar en un dataframe todos los datos climatológicos a lo largo de los años especificados y que será consumido por el resto de las funciones para la visualización de las variables climatológicas más relevantes en el cálculo de ET_O (evapotranspiración de referencia). Se tiene la opción de exportar el dataframe con todos los datos mensuales Conagua a un archivo CSV con la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#############################################################################################################
def concat_conagua_month_(base_station, start_year, end_year, export_file = True):
    """
    Parametros
    ----------
    Entrada
        base_station : string
            Nombre de la estacion climatologica
        start_year,  : int
            Año inicial
        end year : int
            Año Final
        export_file : boolean
            True  -> guardar archivo en ../output/exploration/
            False -> se despliega en consola los datos preparados
    ----------
    Salida
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
    """
#############################################################################################################
```

4.1.2 conagua_average_month_graph ()

Esta función permite visualizar el promedio mensual de temperatura, humedad, irradiación solar, velocidad viento, lluvia a lo largo del tiempo contenido en Conagua dataframe. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################################
def conagua_average_month_graph(conagua_df, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#########################################################################################################################
```

4.1.3 crop_season_humidity_graph ()

Esta función permite visualizar el comportamiento de la humedad relativa del aire (%) durante el periodo de tiempo en el que tiene lugar el cultivo a partir del día de trasplante que se indique (líneas puenteadas verticales de color gris). Se pueden ajustar los limites superiores e inferiores para identificar en que fechas se presentan valores de humedad fuera de lo esperado. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################################
def crop_season_humidity_graph(conagua_df, trans_day, crop_days_stage, upper_limit = 90, lower_limit = 40, export_im-
age = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        trans_day,  : date
            Dia de transplante
        crop_days_stage : dict
            Diccionario que describe los dias para cada etapa del cultivo
        upper_limit,  : int
            limite superior [default = 90]
        lower_limit : int
            limite inferior [default = 40]
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#########################################################################################################################
```

4.1.4 crop_season_temperature_graph ()

Esta función permite visualizar el comportamiento de la temperatura del aire (°C) durante el periodo de tiempo en el que tiene lugar el cultivo a partir del día de trasplante que se indique (líneas puenteadas verticales de color gris). Se pueden ajustar los limites superiores e inferiores para identificar en que fechas se presentan valores de temperatura fuera de lo esperado. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################
def crop_season_temperature_graph(conagua_df, trans_day, crop_days_stage, upper_limit = 25, lower_limit = 8, export_im-
age = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        trans_day,  : date
            Dia de transplante
        crop_days_stage : dict
            Diccionario que describe los dias para cada etapa del cultivo
        upper_limit,  : int
            limite superior [default = 25]
        lower_limit : int
            limite inferior [default = 8]
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#########################################################################################################
```

4.1.5 crop_season_solar_graph ()

Esta función permite visualizar el comportamiento de la irradiación solar ($MJm^{-2}day^{-1}$) durante el periodo de tiempo en el que tiene lugar el cultivo a partir del día de trasplante que se indique (líneas puenteadas verticales de color gris). Se presentan los limites superior e inferiores para identificar en qué fecha se presentan valores máximos y mínimos de radiación solar/horas sol. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################
def crop_season_solar_graph(conagua_df, trans_day, crop_days_stage, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        trans_day,  : date
            Dia de transplante
        crop_days_stage : dict
            Diccionario que describe los dias para cada etapa del cultivo
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#########################################################################################################
```

4.1.6 crop_season_windspeed_graph ()

Esta función permite visualizar el comportamiento de la velocidad del viento (m/s) durante el periodo de tiempo en el que tiene lugar el cultivo a partir del día de trasplante que se indique (líneas puenteadas verticales de color gris). Se presentan los limites superior e inferiores para identificar en qué fecha se presentan los valores máximos y mínimos de radiación solar/horas sol. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################################
def crop_season_windspeed_graph(conagua_df, trans_day, crop_days_stage, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        trans_day,  : date
            Dia de transplante
        crop_days_stage : dict
            Diccionario que describe los dias para cada etapa del cultivo
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#########################################################################################################################
```

4.1.7 crop_season_rain_graph ()

Esta función permite visualizar el comportamiento de la cantidad de lluvia acumulada en mm/día durante el periodo de tiempo en el que tiene lugar el cultivo a partir del día de trasplante que se indique (líneas puenteadas verticales de color gris). Se presentan los limites superior e inferiores para identificar en qué fecha se presentan los valores máximos y mínimos de radiación solar/horas sol. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################################
def crop_season_rain_graph(conagua_df, trans_day, crop_days_stage, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        trans_day,  : date
            Dia de transplante
        crop_days_stage : dict
            Diccionario que describe los dias para cada etapa del cultivo
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#########################################################################################################################
```

4.1.8 timerange_humidity_graph ()

Esta función permite visualizar el comportamiento de la humedad relativa del aire (%) durante un periodo de tiempo comprendido por una fecha inicial y una fecha final. Se pueden ajustar los limites superiores e inferiores para identificar en que fechas se presentan valores de humedad fuera de lo esperado. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################################
def timerange_humidity_graph(conagua_df, start_date, end_date, upper_limit = 90, lower_limit = 40, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        start_date,  : date
            Dia inicial del rango de tiempo a graficar
        end_date,  : date
            Dia final del rango de tiempo a graficar
        upper_limit,  : int
            limite superior [default = 90]
        lower_limit : int
            limite inferior [default = 40]
        crop_days_stage : dict
            Diccionario que describe los dias para cada etapa del cultivo
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#########################################################################################################################
```

4.1.9 timerange_temperature_graph ()

Esta función permite visualizar el comportamiento de la temperatura del aire (°C) durante un periodo de tiempo comprendido por una fecha inicial y una fecha final. Se pueden ajustar los limites superiores e inferiores para identificar en que fechas se presentan valores de temperatura fuera de lo esperado. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#########################################################################################################################
def timerange_temperature_graph(conagua_df, start_date, end_date, upper_limit = 25, lower_limit = 8, export_im-
age = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        start_date,  : date
            Dia inicial del rango de tiempo a graficar
        end_date,  : date
            Dia final del rango de tiempo a graficar
        upper_limit,  : int
            limite superior [default = 25]
        lower_limit : int
            limite inferior [default = 8]
        crop_days_stage : dict
            Diccionario que describe los dias para cada etapa del cultivo
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#########################################################################################################################
```

4.1.10 timerange_solar_graph ()

Esta función permite visualizar el comportamiento de la radiación solar ($MJm^{-2}day^{-1}$) durante un periodo de tiempo comprendido por una fecha inicial y una fecha final. Se presentan los limites superior e inferiores para identificar en qué fecha se presentan valores máximos y mínimos de radiación solar/horas sol. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#################################################################################################################
def timerange_solar_graph(conagua_df, start_date, end_date, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        start_date,  : date
            Dia inicial del rango de tiempo a graficar
        end_date,  : date
            Dia final del rango de tiempo a graficar
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#################################################################################################################
```

4.1.11 timerange_windspeed_graph ()

Esta función permite visualizar el comportamiento de la velocidad del viento (m/s) durante un periodo de tiempo comprendido por una fecha inicial y una fecha final. Se presentan los limites superior e inferiores para identificar en qué fecha se presentan valores máximos y mínimos de velocidad del viento. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#################################################################################################################
def timerange_windspeed_graph(conagua_df, start_date, end_date, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        start_date,  : date
            Dia inicial del rango de tiempo a graficar
        end_date,  : date
            Dia final del rango de tiempo a graficar
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#################################################################################################################
```

4.1.12 timerange_rain_graph ()

Esta función permite visualizar el comportamiento de la cantidad de lluvia recolectada en un dia (mm/dia) durante un periodo de tiempo comprendido por una fecha inicial y una fecha final. Se presentan los limites superior e inferiores para identificar en qué fecha se presentan valores máximos y mínimos de lluvia. Ofrece la opción de exportar el grafico como archivo imagen en la ruta por defecto. A continuación, se describen los parámetros involucrados:

```
#################################################################################################################
def timerange_rain_graph(conagua_df, start_date, end_date, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
        start_date,  : date
            Dia inicial del rango de tiempo a graficar
        end_date,  : date
            Dia final del rango de tiempo a graficar
        export_image : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#################################################################################################################
```

4.2. Aplicación exploración de datos

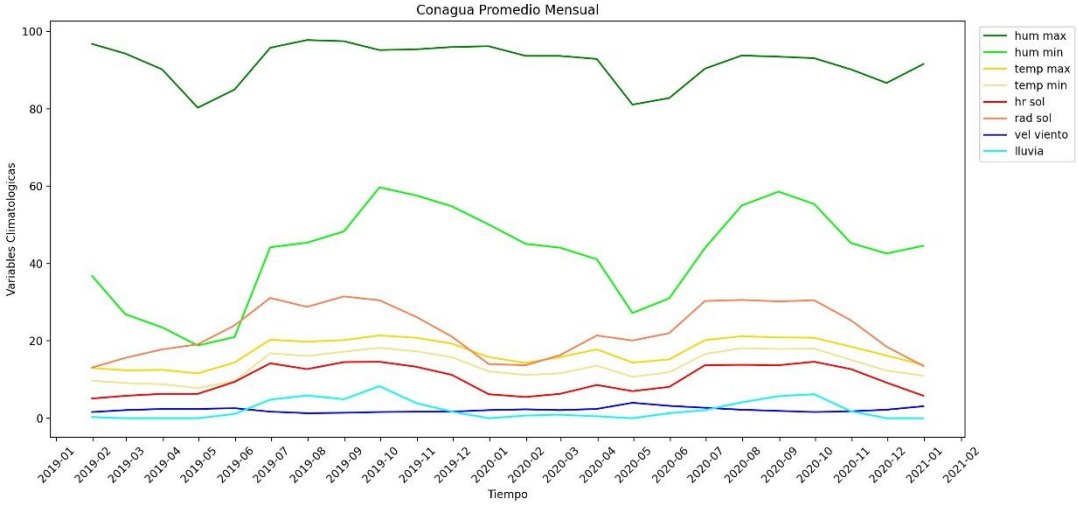
A continuación, se presenta la exploración de los datos utilizando las funciones descritas anteriormente. Para ello utilizaremos las siguientes variables para inyectar a cada uno de los argumentos de entrada:

- Nombre de la estación climatológica: Cd. Guzman
- Año inicial y final de archivos digitales Conagua a utilizar: 2019-2020
- Fecha inicial y final para visualización de datos climatológicos: 17/Junio/2019-17/Julio/2019
- Fecha de trasplante del cultivo del tomate: 4/Mayo/2019
- Número de días para cada una de las etapas de acuerdo con el cultivo del tomate: inicial (30 días), desarrollo (40 días), media (45 días), tardía (30 días)

```
#########################################################################################################
# Definir nombre estacion climatologica
base_station = 'cdguzman'

# Definir el año de inicio y final de nuestros archivos digitales Conagua a explorar
start_year = 2019
end_year = 2020

# Definir el rango de tiempo para visualizar datos climatologicos
# Fecha inicial para visualizacion rango de tiempo
start_timerange_year = 2020
start_timerange_month = 6
start_timerange_day = 17

# Fecha final para visualizacion rango de tiempo
end_timerange_year = 2020
end_timerange_month = 7
end_timerange_day = 17

# Generacion de variable tipo datetime para fecha inicial/final
start_date = datetime(start_timerange_year, start_timerange_month, start_timerange_day)
end_date = datetime(end_timerange_year, end_timerange_month, end_timerange_day)

# Definir fecha de trasplante
year = 2019
month = 12
day = 1
hour = 10
minute = 00

# Convertimos fecha de transplante a una variable de tipo datetime
transplant_day = datetime(year, month, day, hour, minute)
# print(transplant_day)w.fao.org/land-water/databases-and-software/crop-information/tomato/en/
# Definimos las etapas del cultivo de tomate
tomato_days_stage= {
                    "initial":      30,
                    "development":  40,
                    "mid_season":   45,
                    "late_season":  30
}
#########################################################################################################
```

4.2.1 Promedio mensual datos Conagua

A continuación, se muestra la representación gráfica de las variables del clima de forma mensual, a lo largo del tiempo registrado:

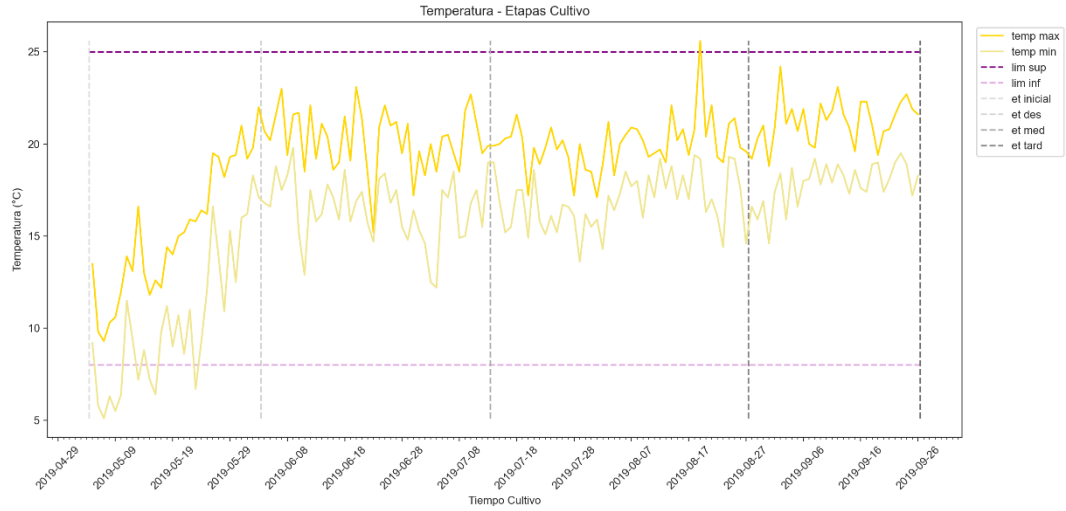


4.2.2 Humedad relativa temporada cultivo

A continuación, se muestra la representación gráfica de la variable humedad relativa, a lo largo del tiempo del cultivo:

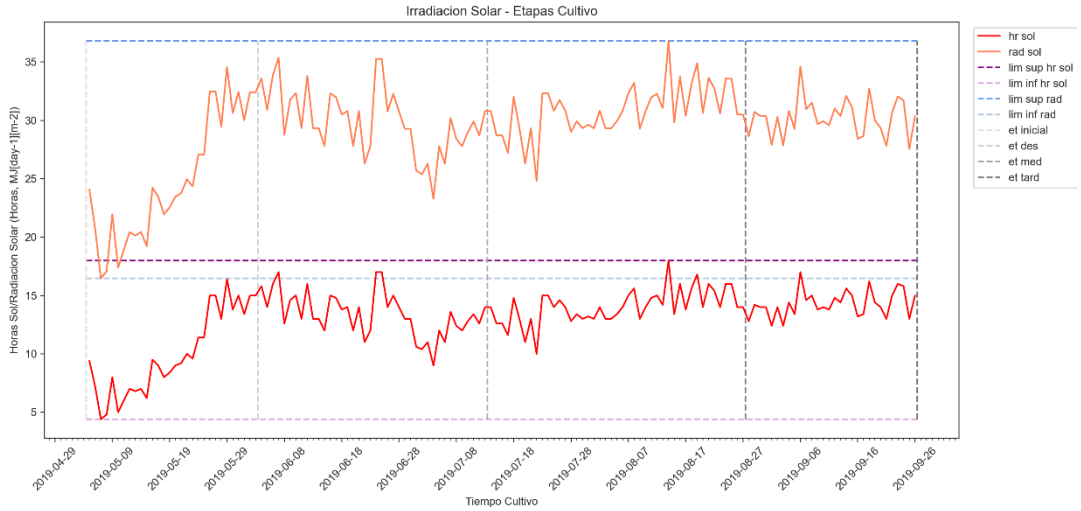
4.2.3 Temperatura temporada cultivo

A continuación, se muestra la representación gráfica de la variable temperatura del aire, a lo largo del tiempo del cultivo:



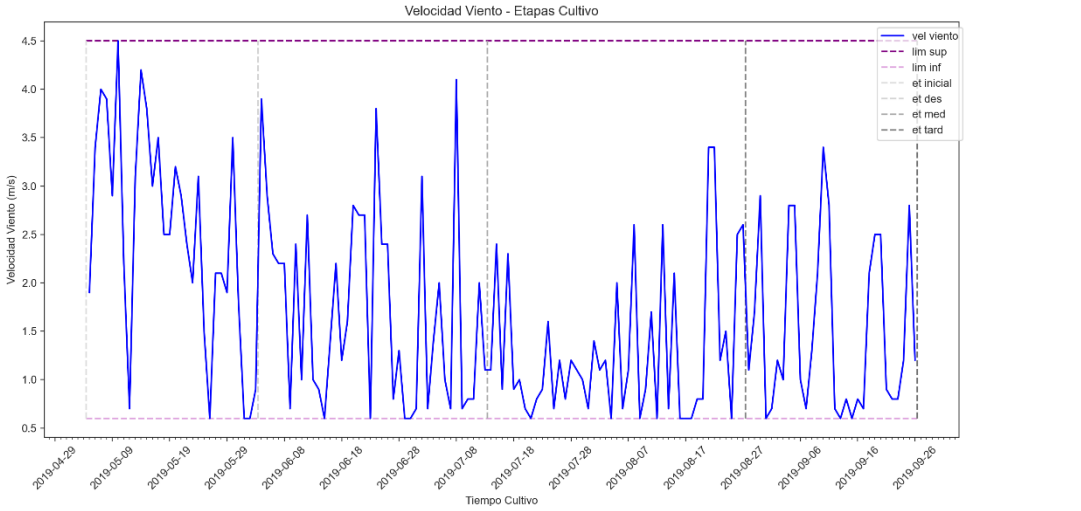
4.2.4 Radiación solar temporada cultivo

A continuación, se muestra la representación gráfica de la variable radiación solar, a lo largo del tiempo del cultivo:



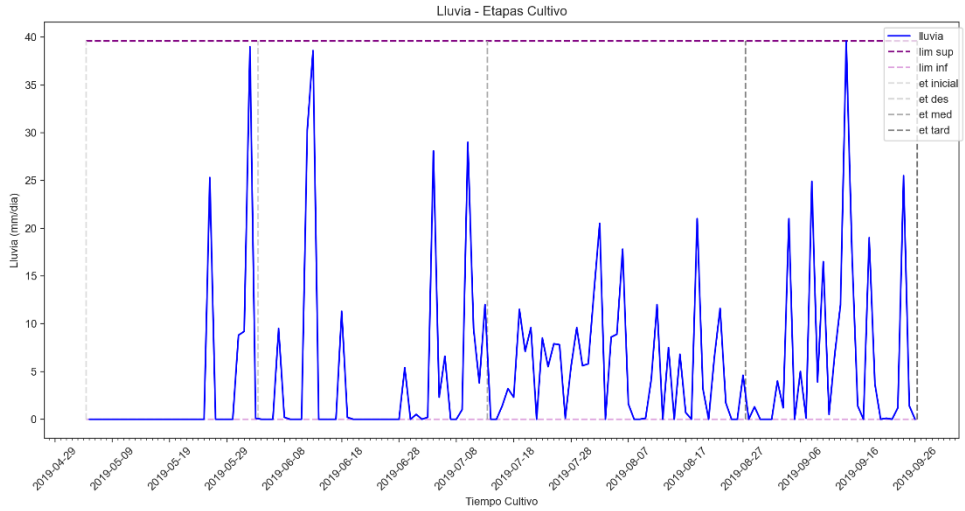
4.2.5 Velocidad viento temporada cultivo

A continuación, se muestra la representación gráfica de la variable velocidad del viento, a lo largo del tiempo del cultivo:



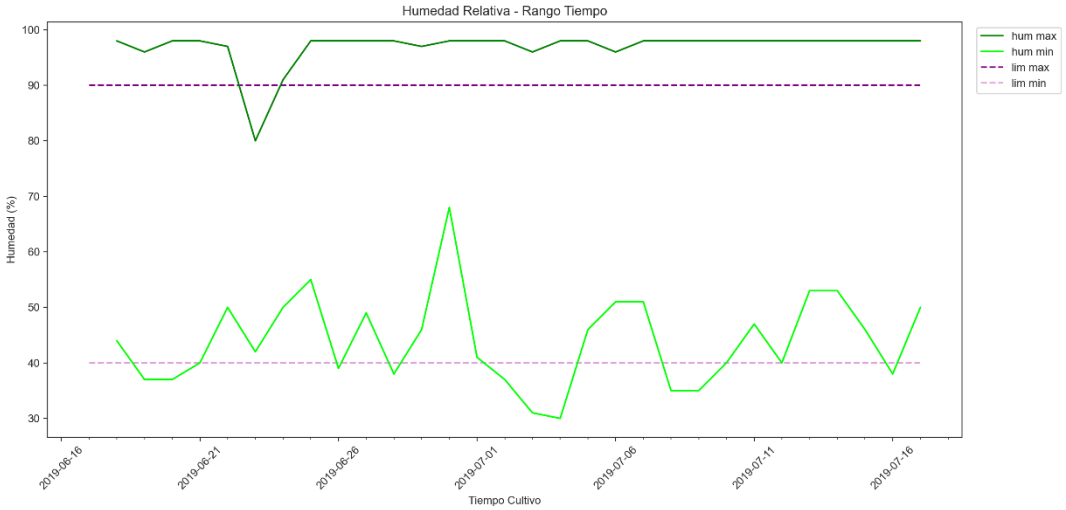
4.2.6 Lluvia temporada cultivo

A continuación, se muestra la representación gráfica de la variable lluvia acumulada, a lo largo del tiempo del cultivo:



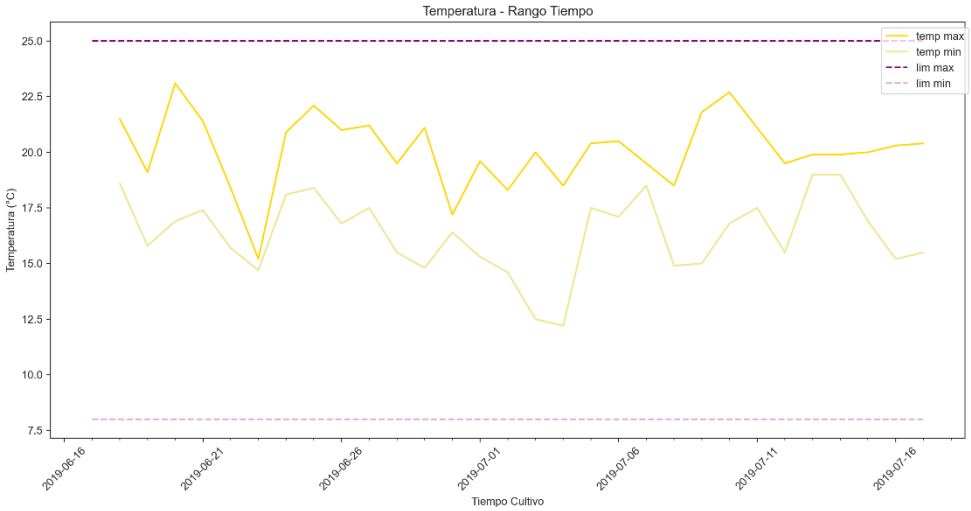
4.2.7 Humedad rango fechas

A continuación, se muestra la representación gráfica de la variable humedad del aire en el rango de tiempo comprendido entre 17/Junio/2019 - 17/Julio/2019:



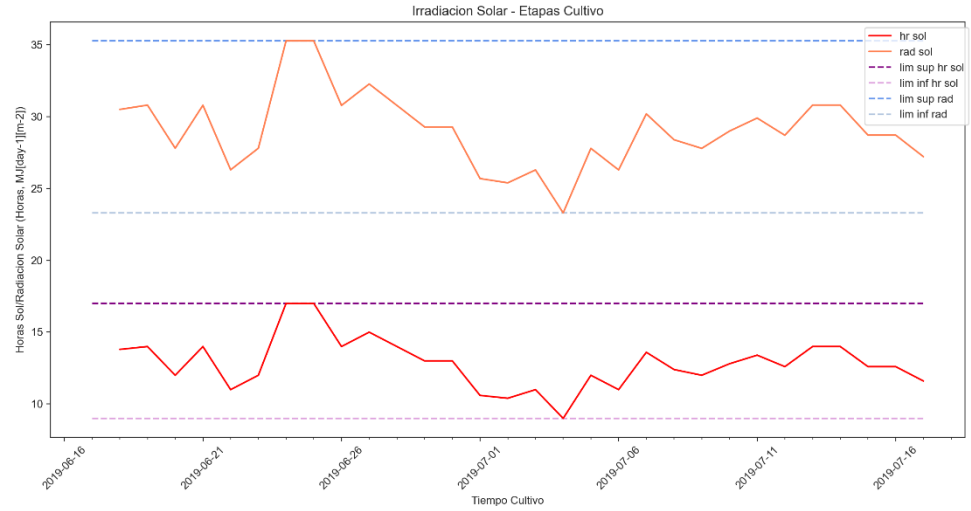
4.2.8 Temperatura rango fechas

A continuación, se muestra la representación gráfica de la variable temperatura del aire en el rango de tiempo comprendido entre 17/Junio/2019 - 17/Julio/2019:



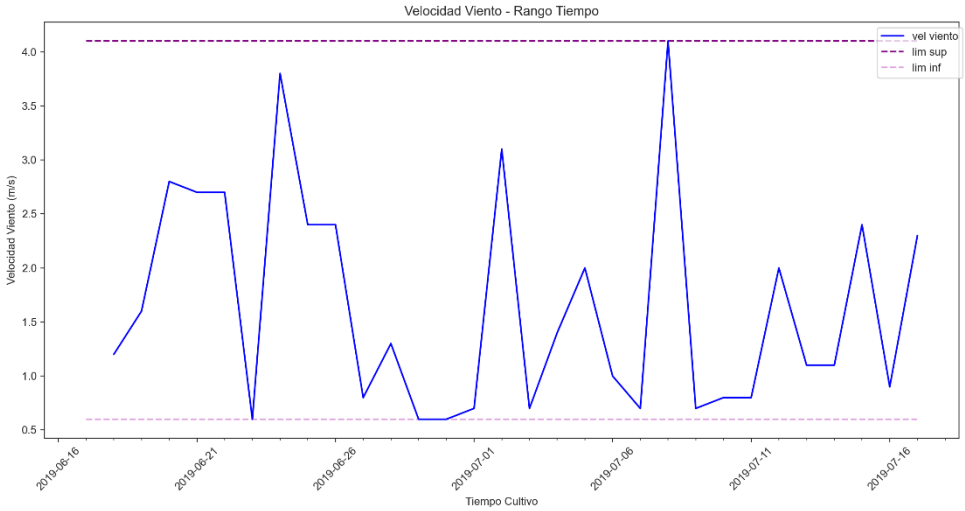
4.2.9 Radiación solar rango fechas

A continuación, se muestra la representación gráfica de la variable radiación solar en el rango de tiempo comprendido entre 17/Junio/2019 - 17/Julio/2019:



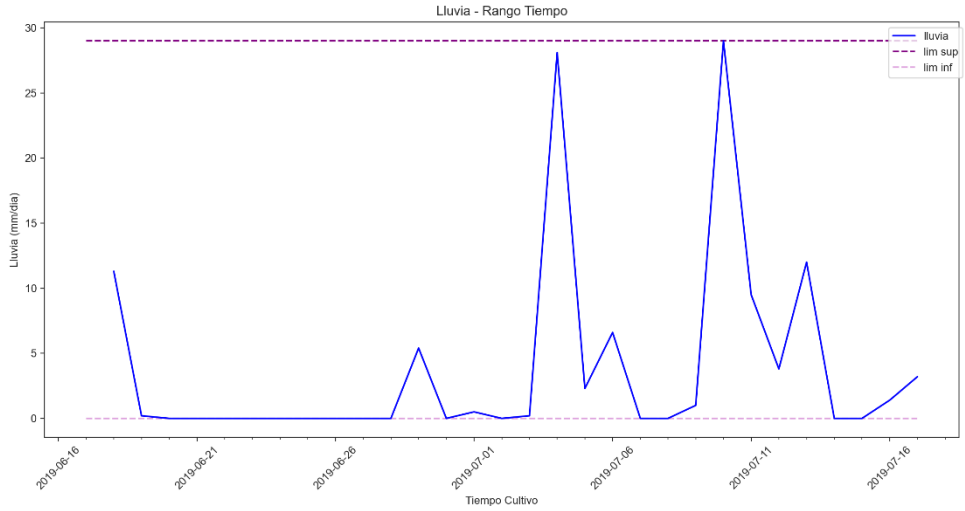
4.2.10 Velocidad viento rango fechas

A continuación, se muestra la representación gráfica de la variable velocidad del viento en el rango de tiempo comprendido entre 17/Junio/2019 - 17/Julio/2019:



4.2.11 Lluvia rango fechas

A continuación, se muestra la representación gráfica de la variable acumulación de lluvia en el rango de tiempo comprendido entre 17/Junio/2019 - 17/Julio/2019:



4.3. Conclusiones

La exploración de datos climatológicos nos ha permitido obtener una visualización con mayor detalle gracias al formato “datetime” dentro de la columna fecha.

En la práctica, nos dimos cuenta de que la combinación de las herramientas en tiempo ampliado y por rango de fechas nos permitió fácilmente encontrar posibles datos que no se encuentren dentro de los rangos de valores esperados (ya sea por un error humano en el proceso de captura de este) los cuales al no tener un valor coherente (como en el caso de las temperaturas) sus gráficas los representaban con puntos muy

por fuera a la tendencia de la curva de la variable. Para este punto se optó por el momento, realizar una corrección manual dentro del archivo mensual preparado Conagua.

El tiempo de referencia para cada una de las etapas del cultivo, agrega simplicidad para monitorear los posibles efectos del clima en relación con la demanda de agua necesaria por la planta en sus diferentes etapas fenológicas.

5. CONSTRUCCIÓN DE MODELOS

Resumen: En esta sección se presenta la construcción del modelo fundamentado en el cálculo de evapotranspiración de referencia a través de la ecuación de Penman-Monteith, la cual es analizada con la finalidad de ejemplificar las variables del clima involucradas. Se describen las principales funciones de los módulos desarrollados: 1) Cálculo ET_O, 2) Cálculo ET_C y 3) Exploración ET_C. Para el primer bloque la parte matemática quedara a cargo de la librería de Python ET_O. El segundo bloque y tercer bloque están destinados al cálculo y visualización de ET_C.

5.1. Proceso de construcción de modelos

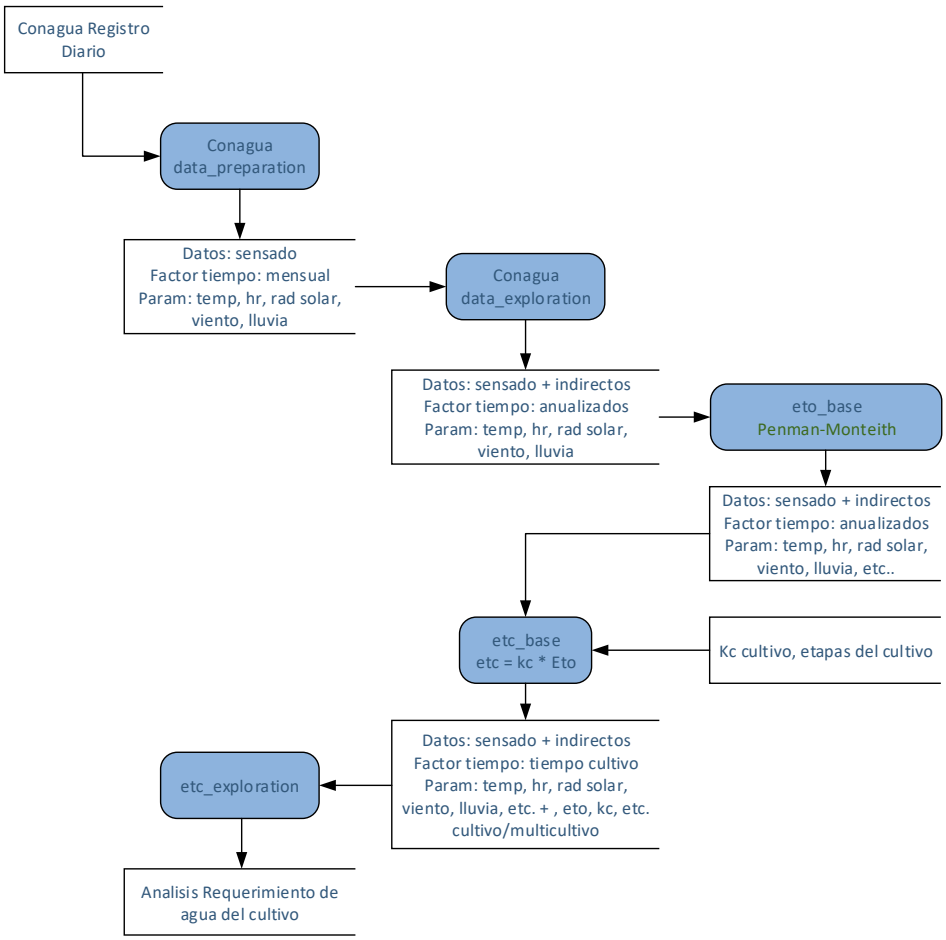
La evapotranspiración (ET) es la combinación del proceso de evaporación de la superficie y la transpiración del tejido de las plantas principalmente por la actividad de sus estomas. La medición directa de la ET real es un proceso muy difícil y por lo tanto se utilizan estimaciones indirectas como consecuencia de ello. Debido a que la ET es parte dominante en el ciclo del agua (~65%) y la agricultura optimizada requiere estimaciones precisas para poder determinar las necesidades de riego, se han realizado extensos trabajos para poder determinar con gran precisión a través del uso de metodologías prácticas la estimación de la ET [8].

La mayoría de este desarrollo proviene en conjunto de la Organización de las Naciones Unidas de Alimentación y Agricultura (UN-FAO por sus siglas en ingles), en la cual se ha trabajado con investigadores para una estandarización internacional. Como lo más practico era solo la medición de parámetros climatológicos (temperatura, humedad relativa, por mencionar algunos), el termino ET de referencia (ET_O) fue adjudicado para definir una superficie especifica de vegetación por la cual la estimación de ET se pudiera representar. Diferentes coeficientes de cultivo pudieron ser aplicados para convertir la referencia del cultivo a otros tipos de cultivos o superficies vegetativas [8].

El método utiliza la ecuación de ET Penman-Monteith y los lineamientos provee algunos métodos para estimar parámetros meteorológicos faltantes. El método y los lineamientos pueden manejar desde un nivel bajo de datos como mínimo y máximo de temperatura, hasta un conjunto de parámetros meteorológicos muy robusto. Con dicha estandarización, los investigadores y administradores de agua pueden de forma precisa estimar ET_O y de ultima forma ser capaces de comparar los resultados a través de otras regiones [8].

La derivación de ET_O se ha desarrollado a través de muchos años con una gran cantidad de ecuaciones. Las más recientes y posiblemente las ultimas variantes son las derivadas de la ecuación Penman-Montieth. Una documentación extensiva al respecto puede ser encontrada en el artículo UN-FAO 56 [9].

El modelo propuesto a nivel programación para contemplar el cálculo del requerimiento de agua de un cultivo se puede simplificar a través de la siguiente imagen:



5.2. Modulo ET_O

A continuación, se muestra una descripción de la ecuación original Penman-Monteith:

$$ET_o = \frac{0.408\Delta(R_n - G) + \gamma \frac{900}{T+273} u_2 (e_s - e_a)}{\Delta + (1 + 0.34u_2)}$$

donde:

- ET_O evapotranspiración referencia [mm day^{-1}]
- Rn radiación en superficie del cultivo [MJ m^{-2} day^{-1}]
- G densidad de flujo del suelo [MJ m^{-2} day^{-1}]
- T temperatura del aire a 2mt altura [°C]
- u2 velocidad del viento a 2mt altura [m s^{-1}]
- es presión de saturación vapor [kPa]
- ea presión de vapor real [kPa]
- es - ea déficit de saturación de vapor [kPa]
- D pendiente a curva presión vapor [kPa $°C^{-1}$]
- g constante psicométrica [kPa $°C^{-1}$]

El proceso para llevar a cabo el cálculo de la ET_O, contempla un mayor o menor número de variables según el escalón de tiempos a utilizar: 10 días o mensual, diario, y hora. Cada uno de estos procesos, así como la variante de la ecuación de Penman-Monteith a utilizarse puede ser consultada en el capítulo 4 de los lineamientos de evapotranspiración del cultivo de la FAO [12]. Para cubrir nuestro objetivo y por el formato del cual nos proporciona registro Conagua para las diferentes variables del clima, se llevará a cabo el proceso con frecuencia de día a día.

El módulo de Python "eto_base" [16] tiene dependencia con la librería de Python ETO [6] y está diseñado para asistir en la estimación eficiente de la evapotranspiración de referencia para una serie de datos meteorológicos, en la cual la instrumentación y los parámetros cambian a través del tiempo debido a cambios por prioridades o por presupuesto. En seguida se describen las funciones principales de dicho módulo.

5.2.1 loading_conagua_prep_data ()

Esta función tiene como objetivo tomar el archivo CSV con los datos preparados anualizados producto del módulo de Python "data_exploration" y exponerlo en un dataframe para su manipulación. A continuación, se describen los parámetros involucrados:

```
#################################################################################################################
def loading_conagua_prep_data(base_station, start_year, end_year):
    """
    Parametros
    ----------
    Entrada
        base_station : string
            Nombre de la estacion climatologica
        start_year,  : int
            Año inicial de datos conagua preparados
        end year : int
            Año Final de datos conagua preparados
    ----------
    Salida
        conagua_df : dataframe
            Dataframe que contempla todo el conjunto de datos conagua
    """
#################################################################################################################
```

5.2.2 eto_base ()

La función "eto_base" es donde reside el principal motor para llevar a cabo la estimación de la ET_O, la cual recibe el conjunto de datos Conagua, así como las características del lugar donde se desea

analizar. A continuación, se describen los parámetros a utilizar en dicha función:

```
#################################################################################################################
def eto_base(eto, conagua_df, freq, z_msl, lat, lon, TZ_lon):
    """
    Parametros
    ----------
    Entrada
        eto : object
            Objeto de clase ETo
        freq  : string
            Frecuencia de datos contemplada
            en base de datos conagua: hora, dia, mes
        z_msl : int
            Altitud del punto a evaluar (m)
        lat : float
            Longitud de la estacion (dec deg)
        lon : float
            Longitud de la estacion (dec deg)
        TZ_lon : Longitud central de la zona de horario (dec deg)
    ----------
    Salida
        0 : int
    """
#################################################################################################################
```

El objeto que hereda de la clase ET_O toma como referencia el nombre de las columnas del dataframe de Conagua para determinar si es necesario realizar la estimación de algunos parámetros faltantes. En nuestro proceso de preparación se manipularon las columnas para garantizar contemplar los siguientes parámetros base: temperaturas, humedad, vientos, radiación solar, y lluvia los cuales son el mínimo requerido para un cálculo con una frecuencia de datos por día.

Se realiza un barrido para cada uno de los registros al interior del dataframe de Conagua y se calcula su ET_O donde el valor obtenido será depositado en una nueva columna referente a ET_O.

5.2.3 export_file ()

La función para exportar permite guardar dos archivos, el dataframe el cual consiste en datos Conagua más la columna ET_O y un segundo archivo el cual contiene los datos Conagua más las variables faltantes contempladas por la función que hereda de la clase ET_O denominada “ts_param”. A continuación, se describen los parámetros a utilizar en la presente función:

```
#################################################################################################################
def export_file(conagua_df, eto):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Dataframe Conagua + columna ETo
        eto  : object
            Objeto de la clase ETo
    ----------
    Salida
        0 : int
    """
    # Se guardan dos archivos, el base el cual consiste de datos conagua + eto
    # y el autogenerado por eto_ts_param el cual involucra inferencia de mas variables
    conagua_df.to_csv('../output/eto/eto_'+base_station+'.csv')
    eto.ts_param.to_csv('../output/eto/eto_ts_param_'+base_station+'.csv')

    return 0
#################################################################################################################
```

5.3. Modulo ET_C

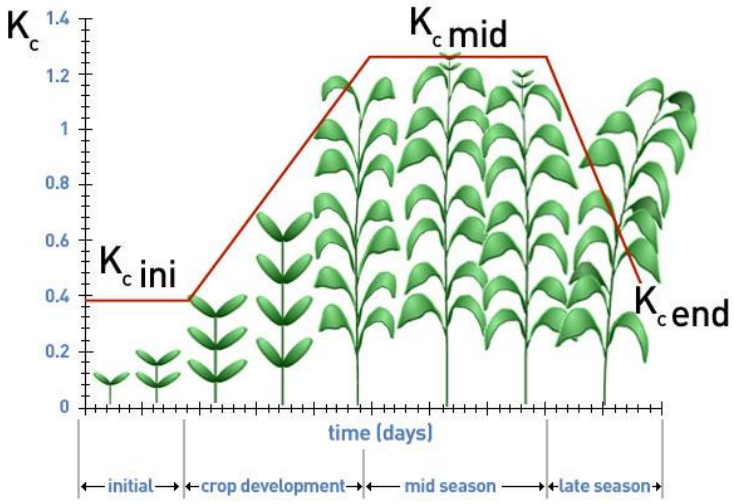
El enfoque de evapotranspiración por coeficiente del cultivo es calculado por la multiplicación directa de la referencia de evapotranspiración por un coeficiente del cultivo, en otras palabras:

$$ET_c = K_c ET_O$$

donde:

- ET_C evapotranspiración del cultivo [mm d^{-1}]
- K_C coeficiente del cultivo [adimensional]
- ET_O referencia de evapotranspiración [mm d^{-1}]

K_C se le conoce como coeficiente sencillo y el propósito de dicha calculación va enfocado a la planificación y desarrollo de riegos. En el caso de nuestro cultivo seleccionado para este proyecto es el tomate y para ello tomaremos la siguiente referencia propuesta por la FAO para el factor K_C en las diferentes etapas fenológicas de la planta [10]:




```
#############################################################################################################
# Definimos las etapas del cultivo de tomate
tomato_days_stage= {
                    "initial":      30,
                    "development":  40,
                    "mid_season":   45,
                    "late_season":  30
}

# Definimos Kc del cultivo de tomate
tomato_kc = {
                    "initial":      0.6,
                    "mid_season":   1.15,
                    "late_season":  0.8
}
#############################################################################################################
```

El módulo “etc_base” [17] toma como referencia el archivo que contiene la base de datos Conagua más la columna de ET_O. Como parte del análisis se presentan dos funcionalidades: una sencilla y una múltiple.

El análisis sencillo nos arroja como resultado el cálculo total requerido en un cultivo denotado por una fecha de trasplante, el cual es exportado en formato CSV para su posterior exploración de resultados.

El análisis múltiple permite hacer una comparativa de gasto de agua requerido en un rango de tiempos en el cual se van realizando de forma recurrente los cálculos de agua total requerida por el cultivo utilizando los diferentes días de trasplante entre las fechas especificadas. El objetivo es poder hacer una comparativa de los gastos de agua en caso de tener movimientos en el día del trasplante.

5.3.1 slope ()

Esta función permite calcular la pendiente que presenta la curva de K_C para aquellos valores en transición en la etapa desarrollo y tardía de forma independiente a los días involucrados para cada una de las etapas fenológicas del cultivo. A continuación, se describen los parámetros a utilizar en la presente función:

```
#############################################################################################################
def slope(x1, y1, x2, y2):
    """
    Parametros
    ----------
    Entrada
        x1 : int
            Punto 1 en el eje x
        y1  : int
            Punto 1 en el eje y
        x2 : int
            Punto 2 en el eje x
        y2 : int
            Punto 2 en el eje y
    ----------
    Salida
        slope : int
            Pendiente de la recta producto de P1 y P2
    """
#############################################################################################################
```

5.3.2 kc_init_stage ()

Esta función permite anexar a la lista de valores K_C los correspondientes a la etapa inicial. Por ser una etapa lineal solo se asignan los valores de K_C de acuerdo con los días de duración de esta. A continuación, se describen los parámetros a utilizar en la presente función:

```
#############################################################################################################
def kc_init_stage(kc_crop, crop_days_stage, crop_kc_steps):
    """
    Parametros
    ----------
    Entrada
        kc_crop : list
            Lista que contiene los valores de Kc
            durante todo el cultivo
        crop_day_stage  : dictionary
            Diccionario con los dias de cada etapa
            del cultivo
        crop_kc_steps : dictionary
            Diccionario con los valores de Kc en
            las diferentes etapas/escalones del cultivo
    ----------
    Salida
        kc_crop : list
            Los valores de kc durante la etapa inicial
            se anexan a la lista kc_crop
    """
#############################################################################################################
```

5.3.3 kc_dev_stage ()

Esta función permite anexar la lista de valores K_C los correspondientes a la etapa de desarrollo. Al ser una etapa donde presenta un cambio ascendente para K_C conforme evolución la planta, se requiere hacer uso del valor de la recta pendiente producto de la función “slope” para guardar el valor correspondiente a cada uno de sus días. A continuación, se describen los parámetros a utilizar en la presente función:

```
#########################################################################################################
def kc_dev_stage(kc_crop, crop_days_stage, crop_kc_steps):
    """
    Parametros
    ----------
    Entrada
        kc_crop : list
            Lista que contiene los valores de Kc
            durante todo el cultivo
        crop_day_stage  : dictionary
            Diccionario con los dias de cada etapa
            del cultivo
        crop_kc_steps : dictionary
            Diccionario con los valores de Kc en
            las diferentes etapas/escalones del cultivo
    ----------
    Salida
        kc_crop : list
            Los valores de kc durante la etapa de desarrollo
            se anexan a la lista kc_crop
    """
#########################################################################################################
```

5.3.4 kc_mid_stage ()

Esta función permite anexar a la lista de valores K_C los correspondientes a la etapa media. Por ser una etapa lineal solo se asignan los valores de K_C de acuerdo con los días de duración de esta. A continuación, se describen los parámetros a utilizar en la presente función:

```
#########################################################################################################
def kc_mid_stage(kc_crop, crop_days_stage, crop_kc_steps):
     """
    Parametros
    ----------
    Entrada
        kc_crop : list
            Lista que contiene los valores de Kc
            durante todo el cultivo
        crop_day_stage  : dictionary
            Diccionario con los dias de cada etapa
            del cultivo
        crop_kc_steps : dictionary
            Diccionario con los valores de Kc en
            las diferentes etapas/escalones del cultivo
    ----------
    Salida
        kc_crop : list
            Los valores de kc durante la etapa mediana
            se anexan a la lista kc_crop
    """
lista kc_crop
    """
#########################################################################################################
```

5.3.5 kc_late_stage ()

Esta función permite anexar la lista de valores K_C los correspondientes a la etapa tardía. Al ser una etapa donde presenta un cambio descendente para K_C conforme evolución la planta, se requiere hacer uso del valor de la recta pendiente producto de la función “slope” para guardar el valor correspondiente a cada uno de sus días. A continuación, se describen los parámetros a utilizar en la presente función:

```
#########################################################################################################
def kc_late_stage(kc_crop, crop_days_stage, crop_kc_steps):
     """
    Parametros
    ----------
    Entrada
        kc_crop : list
            Lista que contiene los valores de Kc
            durante todo el cultivo
        crop_day_stage  : dictionary
            Diccionario con los dias de cada etapa
            del cultivo
        crop_kc_steps : dictionary
            Diccionario con los valores de Kc en
            las diferentes etapas/escalones del cultivo
    ----------
    Salida
        kc_crop : list
            Los valores de kc durante la etapa tardia
            se anexan a la lista kc_crop
    """
#########################################################################################################
```

5.3.6 etc_timerange ()

Esta función nos brinda un dataframe reducido proveniente del contexto completo que alberga Conagua para así realizar un análisis sencillo de requerimiento de agua en el cultivo de una sola temporada. A continuación, se describen los parámetros a utilizar en la presente función:

```
#########################################################################################################
def etc_timerange(conagua_df, kc, trans_day):
    """
    Parametros
    ----------
    Entrada
        conagua_df : dataframe
            Base de datos climatologicos Conagua + ETo
        kc  : list
            Lista con todos los factores Kc durante el
            cultivo
        trans_day : datetime
            Dia de trasplante del cultivo
    ----------
    Salida
        crop_df : dataframe
            dataframe con datos climatologicos + ETo + Kc + ETc,
            referentes al periodo del cultivo
    """
#########################################################################################################
```

5.3.7 export_file ()

Esta función nos permite exportar en archivo CSV los datos de requerimiento de agua del cultivo para su posterior exploración grafica. A continuación, se describen los parámetros a utilizar en la presente función:

```
#########################################################################################################
def export_file(etc_df, file_ver=''):
    """
    Parametros
    ----------
    Entrada
        etc_df : dataframe
            dataframe con datos climatologicos + ETo + Kc + ETc,
            referentes al periodo del cultivo
        file_ver  : string
            Nombre de la version que se concatenara al nombre
            base del archivo de salida
    ----------
    Salida
        0 : int
    """
#########################################################################################################
```

5.3.8 daterange ()

Esta función devuelve un contador tipo “int” para un rango de fechas provenientes en formato tiempo. A continuación, se describen los parámetros a utilizar en la presente función:

```
#################################################################################################################
def daterange(start_date, end_date):
    """
    Parametros
    ----------
    Entrada
        start_date : datetime
            Fecha de inicio
        end_date  : datetime
            Fecha final
    ----------
    Salida
        0 : int
    """
#################################################################################################################
```

5.3.9 crop_water_req_season ()

Esta función permite entregar un dictamen del requerimiento de agua del cultivo ETC para una temporada descrita por su fecha de trasplante. Se arrojan estimados del agua acumulada por lluvias durante el ciclo, así como la diferencia de agua requerida por un sistema de riego en caso de ser necesaria. A continuación, se describen los parámetros a utilizar en la presente función:

```
#################################################################################################################
def crop_water_req_season(etc_crop_df, crop_days_stage):
    """
    Parametros
    ----------
    Entrada
        etc_crop_df : dataframe
            dataframe con datos climatologicos + ETo + Kc + ETc,
            referentes al periodo del cultivo
        crop_days_stage  : dictionary
            Diccionario con los dias referentes a cada etapa
            del cultivo
    ----------
    Salida
        0 : int
    """
#################################################################################################################
```

5.3.10 crop_water_req_season_multiple ()

Esta función nos permite calcular y realizar una comparativa del requerimiento de agua para múltiples temporadas del mismo cultivo tomando como referencia los días entre las fechas especificadas como posibles días de trasplante. A continuación, se describen los parámetros a utilizar en la presente función:

```
#################################################################################################################
def crop_water_req_season_multi-
ple(start_trans_year, start_trans_month, start_trans_day, end_trans_year, end_trans_month, end_trans_day, cona-
gua_df, kc):
    """
    Parametros
    ----------
    Entrada
        start_trans_year : int
            Año de inicio de trasplantes
        start_trans_month  : int
            Mes de inicio de trasplantes
        start_trans_day  : int
            Dia de inicio de trasplantes
        end_trans_year : int
            Año de inicio de trasplantes
        end_trans_month  : int
            Mes final de trasplantes
        end_trans_day  : int
            Dia final de trasplantes
        conagua_df : dataframe
            dataframe con datos climatologicos + ETo + Kc + ETc,
            referentes al periodo del cultivo
        kc : float
            Los valores de kc durante las etapas del cultivo
    ----------
    Salida
        0 : int
    """
#################################################################################################################
```

5.4. Modulo exploración ET_C

El módulo de Python “etc_exploration” [18] está diseñado propiamente para realizar un análisis grafico de nuestros resultados obtenidos a lo largo del proceso de cálculo de requerimiento de agua del cultivo de tomate. Se alimenta propiamente de un archivo CSV el cual contiene los datos climatológicos Conagua de la zona, evapotranspiración de referencia, factor K_C, y evapotranspiración del cultivo en un formato de tiempo en el que incurre la temporada del cultivo. A continuación, se describen las funcionalidades más representativas de dicho módulo.

5.4.1 loading_etc ()

Esta función nos permite cargar en un dataframe todos los parámetros climatológicos, evapotranspiración de referencia, factor K_C, y evapotranspiración registrados durante el periodo del cultivo, el cual será utilizado a lo largo del módulo para efecto de visualización. A continuación, se describen los parámetros a utilizar en la presente función:

```
#################################################################################################################
def loading_etc(base_station, year, month, day):
    """
    Parametros
    ----------
    Entrada
        base_station : string
            Nombre de la estacion base del clima
        year  : int
            Año del trasplante
        month  : int
            Mes del trasplante
        dia  : int
            Dia del trasplante
    ----------
    Salida
        conagua_etc_df : dataframe
            Dataframe con registros Conagua + ETo, Kc, ETc
    """
#################################################################################################################
```

5.4.2 evapotranspiration_graph ()

Esta función nos muestra de forma gráfica la evolución a lo largo del cultivo de las variables de ET_O, factor K_C, ET_C. Se establecen con una línea horizontal tanto los puntos máximos como mínimo de ET_C que lograron incurrir en dicho rango de tiempo. Se denotan con líneas en vertical y en color gris (intensidad de tono en forma ascendente gradual) las fechas en las que da fin e inicia una nueva etapa fenológica del cultivo. A continuación, se describen los parámetros a utilizar en la presente función:

```
#############################################################################################################
def evapotranspiration_graph(conagua_etc_df, crop_days_stage, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_etc_df : dataframe
            Base de datos conagua que contiene variables clima,
            ETo, Kc, ETc,
        crop_days_stage  : dictionary
            Dictionario con los dias para cada etapa fenologica
            de la planta
        export_image  : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#############################################################################################################
```

5.4.3 rain_graph ()

Esta función nos muestra de forma gráfica la evolución de la lluvia acumulada a lo largo del cultivo. Se establecen con una línea horizontal tanto los puntos máximos como mínimo de acumulación de lluvia que lograron incurrir en dicho rango de tiempo. Se denotan con líneas en vertical y en color gris (intensidad de tono en forma ascendente gradual) las fechas en las que da fin e inicia una nueva etapa fenológica del cultivo. A continuación, se describen los parámetros a utilizar en la presente función:

```
#############################################################################################################
def rain_graph(conagua_etc_df, crop_days_stage, export_image = True):
    """
    Parametros
    ----------
    Entrada
        conagua_etc_df : dataframe
            Base de datos conagua que contiene variables clima,
            ETo, Kc, ETc,
        crop_days_stage  : dictionary
            Dictionario con los dias para cada etapa fenologica
            de la planta
        export_image  : boolean
            True  -> guardar archivo en ../output/exploration/ [default]
            False -> se despliega grafico de los datos promediados mensualmente
    ----------
    Salida
        0 : int
    """
#############################################################################################################
```

El módulo de Python “etc_exploration” está diseñado propiamente para realizar un análisis grafico de nuestros resultados obtenidos a lo largo del proceso de cálculo de requerimiento de agua del cultivo de tomate. Se alimenta propiamente de un archivo CSV el cual contiene los datos climatológicos Conagua de la zona, evapotranspiración de referencia, factor K_C, y evapotranspiración del cultivo en un formato de tiempo en el que incurre la temporada del cultivo. A continuación, se describen las funcionalidades más representativas de dicho módulo.

5.5. Conclusiones

Producto del módulo de la exploración de datos, se reutilizo el enfoque de tiempos bajo el periodo en el cual tiene lugar el cultivo, así como los valores K_C de manera escalonada como parámetros de entrada. Fue necesario involucrar el cálculo de la pendiente de una recta para asociar los valores intermedios de K_C a las etapas de desarrollo y la etapa tardía, de acuerdo con el número de días que se contemplan desde su fecha de inicio (x_1, y_1) hasta su fecha final (x_2, y_2). De esta forma es posible reutilizar los módulos ET_O/ET_C para el análisis de diversos cultivos.

La base de datos provista por Conagua presenta una frecuencia de registro diario, por lo que por ahora no se ha realizado la implementación de la librería de Python ET_O en el esquema máximo (mensual) o mínimo (horas). La librería Python ET_O simplificó en gran medida la construcción en código de la parte matemática y permitió rápidamente llevar a cabo la implementación del modelo propuesto.

6. PRESENTACIÓN DE RESULTADOS

Resumen: En esta sección se presentan las variables a poblar en cada uno de los diferentes módulos para poder obtener un resultado de ET_C satisfactorio. Además de suministrar el registro climatológico previamente preparado, se hace referencia a las coordenadas del lugar, altura sobre el nivel del mar, características fenológicas del cultivo, y fecha de trasplante principalmente. Se realiza una estimación de requerimiento de agua de tomate rojo para la ubicación de Ciudad Guzman y se presentan algunos gráficos para contrastar la información obtenida.

6.1. Presentación de resultados

Para la presentación de resultados podríamos resumirla a través de la implementación únicamente de los módulos de Python: “eto_base”, “etc_base” y “etc_exploration” descritos en el capítulo anterior, los cuales tienen como objetivo calcular y evaluar los consumos de agua requeridos por la planta de tomate en el periodo de tiempo definido tomando como referencia el día de trasplante que se halla establecido.

A manera de resumen, se describen los ajustes principales provistos a través de los módulos de Python implementados en nuestro modelo de cálculo de requerimiento de agua por ET_C:

eto_base:

- Año inicial y final de archivos digitales Conagua a utilizar: 2019-2020
- Nombre de la estación climatológica: Cd. Guzman
- Altitud: 1408.99
- Latitud: 19.59216667
- Longitud: -103.59075
- Longitud central de la zona horaria: -103
- Frecuencia de datos suministrados: Dia

etc_base:

- Nombre de la estación climatológica: Cd. Guzman
- Número de días para cada una de las etapas de acuerdo con el cultivo del tomate: inicial (30 días), desarrollo (40 días), media (45 días), tardía (30 días)
- Factor KC para cada uno de los escalones en las diferentes etapas del cultivo: inicial (0.6), media (1.15), tardía (0.8)
- Fecha de trasplante del cultivo del tomate: 1/Junio/2019
- Fecha inicial y final para evaluó múltiple de $ET_{C:}$

etc_exploration:

- Nombre de la estación climatológica: Cd. Guzman
- Número de días para cada una de las etapas de acuerdo con el cultivo del tomate: inicial (30 días), desarrollo (40 días), media (45 días), tardía (30 días)
- Fecha de trasplante del cultivo del tomate: 1/Junio/2019

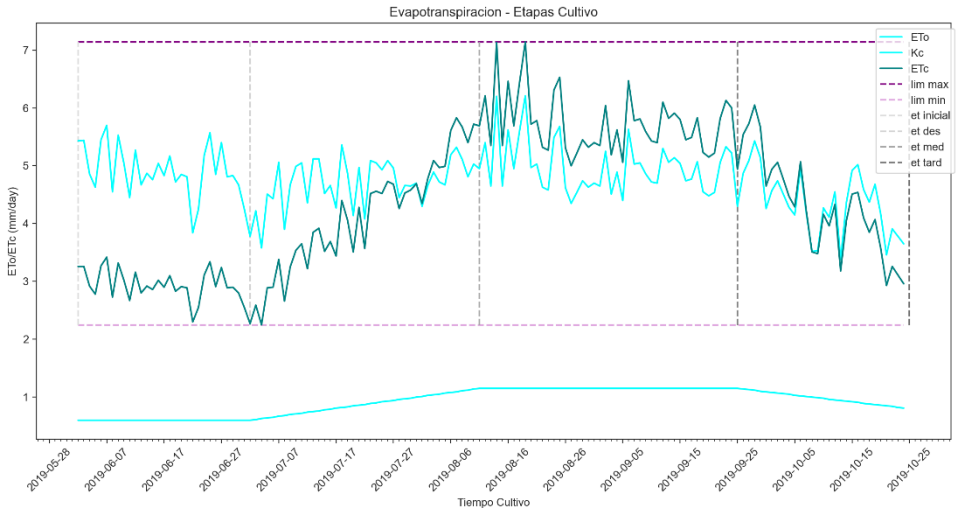
Así bien, con los valores previamente ajustados obtenemos los siguientes resultados:

```
#########################################################################################################
#####################################################################
Cultivo:                            Tomate Rojo
Fecha de Trasplante:                2019-06-01 00:00:00
Agua requerida durante el cultivo:  639.70 mm (~6.4x10^6L)
Agua de lluvia durante el cultivo:  820.8 mm
Deficit agua durante el cultivo:    -181.09 mm
Temp Max durante el cultivo:        25.6 °C
Temp Min durante el cultivo:        12.2 °C
Humedad Max durante el cultivo:     98.0 %
Humedad Min durante el cultivo:     27.0 %
Vel Viento Max durante el cultivo:  4.1 m/s
Vel Viento Min durante el cultivo:  0.6 m/s
Rad Solar Max durante el cultivo:   36.8 MJ[day-1][m-2]
Rad Solar Min durante el cultivo:   23.06 MJ[day-1][m-2]
#####################################################################
```

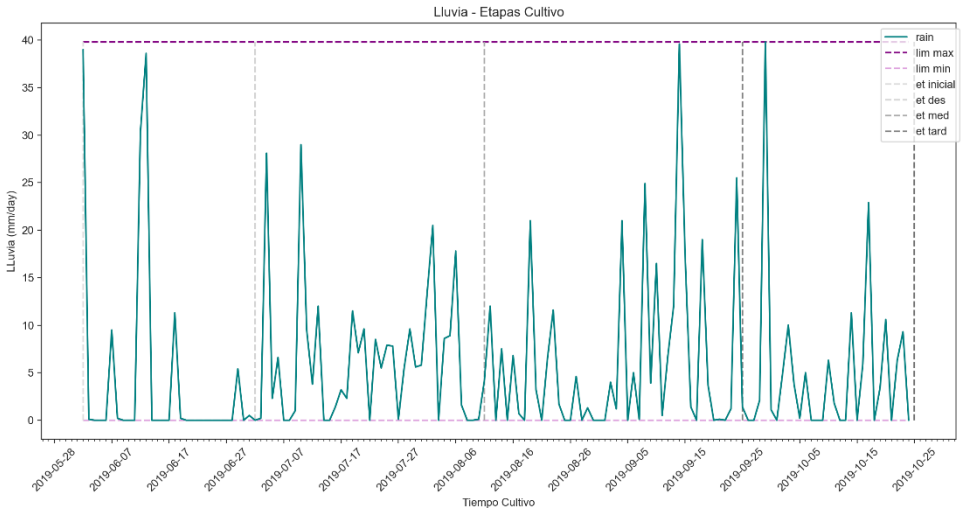
	transp_day_season	etc_crop_season	rain_crop_season	irrig_water_crop_season
0	2019-05-27	643.730042	812.6	-168.869958
1	2019-05-28	643.625212	823.2	-179.574788
2	2019-05-29	642.322771	823.2	-180.877229
3	2019-05-30	641.909063	829.5	-187.590937
4	2019-05-31	640.849346	830.0	-189.150654
5	2019-06-01	639.770587	820.8	-181.029413
6	2019-06-02	638.398858	782.4	-144.001142
7	2019-06-03	636.829037	782.3	-145.470963
8	2019-06-04	635.054604	782.3	-147.245396
9	2019-06-05	633.519283	782.3	-148.780717
10	2019-06-06	631.927321	782.3	-150.372679

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

A continuación, se muestra la representación gráfica del cultivo tomate rojo con una fecha de trasplante del 1/Junio/2019:



A continuación, se muestra la representación gráfica de la evolución de lluvia acumulada a lo largo del cultivo tomate rojo con una fecha de trasplante del 1/Junio/2019:



6.2. Conclusiones

A manera de interpretación de los gráficos producidos, podemos concluir que nuestra agua requerida para un cultivo de tomate rojo que tiene como fecha de trasplante el día 1/Junio/2019 es de un acumulado de 639.7mm, en el cual se tiene un acumulado de lluvia de 820.8mm y será necesario un abastecimiento de agua por riego de -181mm para el periodo de 145 días del cultivo.

El signo negativo de la columna “agua de riego” como parte del resultado obtenido en el módulo de Python “etc_base”, nos indica que para el temporal de las lluvias evaluado se ha presentado un excedente de agua al requerido en el cultivo bajo la estimación de ET_C. Para los productores a campo abierto esto puede representar un factor clave para identificar posibles enfermedades por exceso de agua y decidir/anticipar como manejar tal situación.

Así mismo fue posible hacer un análisis en el consumo de gasto de ET_C, lluvia acumulada, y agua de riego entre los 5 días previos y posteriores a la fecha de trasplante.

7. CONCLUSIONES

Resumen: En esta sección se presentan las principales conclusiones en contraste a los objetivos planteados en el trabajo cálculo de requerimiento de agua de un cultivo, haciendo uso de técnicas aprendidas en el curso programación y análisis de datos a través del lenguaje de programación Python.

7.1. Conclusiones

A pesar de que la estación meteorológica de Ciudad Guzman no cuenta con un sensor dedicado para la medición de la variable de radiación solar, nos fue de gran utilidad utilizar las horas de sol contenidas en la misma base de datos para hacer su estimación a través de la ecuación 34 descrita en el capítulo 3 de la documentación para el cálculo de ET_C de la FAO [6]. Dicha estimación se validó utilizando otras fuentes terceras, como la NASA Power en su apartado de agricultura, para así determinar que los valores no estuviesen por fuera de los valores indicados por una correlación satelital. No fue requerido por lo tanto hacer una fusión de otras bases de datos existentes, como parte de los objetivos específicos mencionados.

El cálculo de ET_C a través de la ecuación Penman-Monteith como parte de los objetivos específicos estipulados, implementa un coeficiente de K_C simple, en el cual, según la bibliografía de la FAO, es una estimación que tiene como finalidad generar programación y diseño de riegos básicos.

La herramienta de SW "Cálculo de Requerimiento de Agua en el Cultivo Tomate Rojo" descrito en el presente documento, se encuentra disponible a manera de consulta externa como repositorio de github [19], cubriendo así el objetivo general del trabajo.

La funcionalidad de "cálculo múltiple para requerimiento de agua" definida en el módulo de Python "etc_base" en conjunto con "etc_exploration", permite realizar un análisis de consumos de agua a lo largo de un tiempo determinado para un cultivo deseado. Por el momento la herramienta, basado en el análisis de datos históricos climatológicos, puede lograr obtener estimaciones altamente confiables de la cantidad de agua que se busca consumir de acuerdo con la planificación de los cultivos.

El desarrollo del capítulo preparación de los datos, se le dio un mayor énfasis a nivel de programación en el uso de las herramientas de Pandas [13] , Numpy [14] para poder llevar a cabo la manipulación/filtración de los datos, como objeto del aprendizaje obtenido en la materia: Programación y Análisis de Datos.

7.2. Trabajo a Futuro

Este trabajo servirá como punto de referencia para evaluar consumos de agua de un cultivo en tiempo real, proveniente de las variables del clima reportados por un sistema de monitoreo basado en comunicación LoRaWAN, el cual se está trabajando actualmente por un servidor.

Se continuará trabajando en los módulos de Python propuestos para agregar al modelo de cálculo de ET_C, una frecuencia por hora y manejo del doble coeficiente de K_C. De esta forma se buscará garantizar una programación de riego en tiempo real con inyecciones de alta frecuencia. Así mismo, se podrá utilizar como referencia de objeto de futuras investigaciones en el tema de estudios de suelo, balances hidrológicos, etc.

La base de datos Conagua es una red meteorológica muy amplia y que guarda relación en la estructura de sus bases de datos mensuales/anuales (por las pláticas que se tuvieron con el representante de la estación de Ciudad Guzman). Un trabajo sin lugar a duda a futuro será buscar como esfuerzo en conjunto la obtención de registros de otras zonas para así evaluar las características de requerimiento de agua para diversos cultivos en diferentes condiciones de clima. El contar con un mayor número de archivos digitales

provistos por Conagua de la zona Sur de Jalisco, permitirá hacer cálculos del tipo estadísticos/predictivos más complejos en cuanto a las características en las cuales se desarrollan los cultivos.

8. REFERENCES

[1] «Chapter 2 - FAO Penman-Monteith equation,» 1998. [En línea]. Available: http://www.fao.org/3/x0490e/x0490e06.htm.

[2] «Conagua,» [En línea]. Available: https://www.gob.mx/conagua.

[3] M. AKSIC, . S. GUDZIC, D. N., G. N. y S. S., «agrojournal,» 2011. [En línea]. Available: https://www.agrojournal.org/17/02-03-11.pdf.

[4] A. Alomran, F. S. Mohammad, H. M. Al-Ghobari y P. Alazba, «researchgate.net,» Enero 2004. [En línea]. Available: https://www.researchgate.net/publication/236607843_Determination_of_evapotranspiration_of_tomato_and_squash_using_lysimeters_in_central_Saudi_Arabia.

[5] Y. H, D. T, M. X y S. MK, «online library wiley,» 21 Mayo 1010. [En línea]. Available: https://acsess.onlinelibrary.wiley.com/doi/pdf/10.1002/vzj2.20074.

[6] A. R. G., P. L. S., R. D. y S. M, «Chapter 3 - Meteorological data,» 1998. [En línea]. Available: http://www.fao.org/3/x0490e/x0490e07.htm#meteorological%20factors%20determining%20et.

[7] A. Baltazar, «data_preparation_eto.ipynb,» 2021. [En línea]. Available: https://github.com/polk96/crop_water_requirement/blob/main/source_code/data_preparation_eto.ipynb.

[8] M. Richards, «pyeto,» [En línea]. Available: https://pyeto.readthedocs.io/en/latest/#.

[9] M. Kittridge, «eto,» 2018. [En línea]. Available: https://eto.readthedocs.io/en/latest/usage.html.

[10] A. R. G., P. L. S., R. D. y S. M, «Solar Radiation data derived from air temperature differences,» 1998. [En línea]. Available: http://www.fao.org/3/X0490E/x0490e07.htm#estimating%20missing%20radiation%20data.

[11] «NASA POWER,» [En línea]. Available: https://power.larc.nasa.gov/data-access-viewer/.

[12] FAO, «Crop Information - Tomato,» 2021. [En línea]. Available: http://www.fao.org/land-water/databases-and-software/crop-information/tomato/en/.

[13] A. Baltazar, «data_exploration_eto.ipynb,» 2021. [En línea]. Available: https://github.com/polk96/crop_water_requirement/blob/main/source_code/data_exploration_eto.ipynb.

[14] A. R. G., P. L. S., R. D. y S. M, «Crop evapotranspiration-Guidelines for computing crop water requirements-FAO Irrigation and drainage paper 56,» FAO, 1998. [En línea]. Available: http://www.fao.org/3/x0490e/x0490e00.htm.

[15] A. R. G., P. L. S., R. D. y S. M, «Chapter 4 - Determination of ETo,» 1998. [En línea]. Available: http://www.fao.org/3/X0490E/x0490e08.htm#chapter%204%20%20%20determination%20of%20eto.

[16] A. Baltazar, «eto_base.ipynb,» 2021. [En línea]. Available: https://github.com/polk96/crop_water_requirement/blob/main/source_code/eto_base.ipynb.

[17] A. Baltazar, «etc_base.ipynb,» 2021. [En línea]. Available: https://github.com/polk96/crop_water_requirement/blob/main/source_code/etc_base.ipynb.

[18] A. Baltazar, «etc_exploration.ipynb,» 2021. [En línea]. Available: https://github.com/polk96/crop_water_requirement/blob/main/source_code/etc_exploration.ipynb.

[19] A. Baltazar, «crop_water_requirement,» 2021. [En línea]. Available: https://github.com/polk96/crop_water_requirement.

[20] NumFOCUS, «Pandas,» [En línea]. Available: https://pandas.pydata.org/.

[21] «Numpy,» 2021. [En línea]. Available: https://numpy.org/.