# 气象数据

## 数据获取

将NASA数据下载到本地，不使用pcse中的气象数据获取方法。

### 获取研究区范围

使用2020年天门市行政区划图。先利用渔网工具建立5000m\*5000m的格网，然后要素转点并计算点的XY坐标，使用十进制的大地2000（4490）坐标系。随后将属性表导出为excel文件。

NASA Power database中的气象数据分辨率为0.5°。所以使用以下函数对XY坐标进行标准化。

def st\_loc(num):

"""

选取气象数据的时候标准化输入为0.5的分辨率

"""

import math

if num-math.floor(num) <=0.25 or num-math.floor(num)>=0.75:

result = round(num)

elif 0.25 < num-math.floor(num) < 0.75:

result = math.floor(num) + 0.5

return result

### 下载数据

利用代码从数据库中获取数据。

# 获取数据的基础函数

import pandas as pd

import numpy as np

import requests

import datetime as dt

from math import exp

MJ\_to\_KJ = lambda x: x \* 1e3

mm\_to\_cm = lambda x: x / 10.

tdew\_to\_kpa = lambda x: ea\_from\_tdew(x) / 10 \* 10.

to\_date = lambda d: d.date()

def getnasadata(latitude, longitude, start\_date, end\_date):

server = "https://power.larc.nasa.gov/api/temporal/daily/point"

# Variable names in POWER data

power\_variables = ["TOA\_SW\_DWN", "ALLSKY\_SFC\_SW\_DWN", "T2M", "T2M\_MIN",

"T2M\_MAX", "T2MDEW", "WS2M", "PRECTOTCORR"]

payload = {"request": "execute",

"parameters": ",".join(power\_variables),

"latitude": latitude,

"longitude": longitude,

"start": start\_date.strftime("%Y%m%d"),

"end": end\_date.strftime("%Y%m%d"),

"community": "AG",

"format": "JSON",

"user": "anonymous"

}

req = requests.get(server, params=payload)

return req.json()

def \_process\_POWER\_records(powerdata):

"""Process the meteorological records returned by NASA POWER

"""

fill\_value = float(powerdata["header"]["fill\_value"])

power\_variables = ["TOA\_SW\_DWN", "ALLSKY\_SFC\_SW\_DWN", "T2M", "T2M\_MIN",

"T2M\_MAX", "T2MDEW", "WS2M", "PRECTOTCORR"]

df\_power = {}

for varname in power\_variables:

s = pd.Series(powerdata["properties"]["parameter"][varname])

s[s == fill\_value] = np.NaN

df\_power[varname] = s

df\_power = pd.DataFrame(df\_power)

df\_power["DAY"] = pd.to\_datetime(df\_power.index, format="%Y%m%d")

# find all rows with one or more missing values (NaN)

ix = df\_power.isnull().any(axis=1)

# Get all rows without missing values

df\_power = df\_power[~ix]

return df\_power

def \_estimate\_AngstAB(df\_power):

"""Determine Angstrom A/B parameters from Top-of-Atmosphere (ALLSKY\_TOA\_SW\_DWN) and

top-of-Canopy (ALLSKY\_SFC\_SW\_DWN) radiation values.

:param df\_power: dataframe with POWER data

:return: tuple of Angstrom A/B values

The Angstrom A/B parameters are determined by dividing swv\_dwn by toa\_dwn

and taking the 0.05 percentile for Angstrom A and the 0.98 percentile for

Angstrom A+B: toa\_dwn\*(A+B) approaches the upper envelope while

toa\_dwn\*A approaches the lower envelope of the records of swv\_dwn

values.

"""

# check if sufficient data is available to make a reasonable estimate:

# As a rule of thumb we want to have at least 200 days available

angstA = 0.29

angstB = 0.49

if len(df\_power) < 200:

msg = ("Less then 200 days of data available. Reverting to " +

"default Angstrom A/B coefficients (%f, %f)")

print(msg)

return angstA, angstB

# calculate relative radiation (swv\_dwn/toa\_dwn) and percentiles

relative\_radiation = df\_power.ALLSKY\_SFC\_SW\_DWN / df\_power.TOA\_SW\_DWN

ix = relative\_radiation.notnull()

angstrom\_a = float(np.percentile(relative\_radiation[ix].values, 5))

angstrom\_ab = float(np.percentile(relative\_radiation[ix].values, 98))

angstrom\_b = angstrom\_ab - angstrom\_a

MIN\_A = 0.1

MAX\_A = 0.4

MIN\_B = 0.3

MAX\_B = 0.7

MIN\_SUM\_AB = 0.6

MAX\_SUM\_AB = 0.9

A = abs(angstrom\_a)

B = abs(angstrom\_b)

SUM\_AB = A + B

if A < MIN\_A or A > MAX\_A or B < MIN\_B or B > MAX\_B or SUM\_AB < MIN\_SUM\_AB or SUM\_AB > MAX\_SUM\_AB:

msg = ("Angstrom A/B values (%f, %f) outside valid range " +

"Reverting to default values.")

msg = msg % (angstrom\_a, angstrom\_b)

print(msg)

return angstA, angstB

return angstrom\_a, angstrom\_b

def \_POWER\_to\_PCSE(df\_power):

# Convert POWER data to a dataframe with PCSE compatible inputs

df\_pcse = pd.DataFrame({"DAY": df\_power.DAY.apply(to\_date),

"IRRAD": df\_power.ALLSKY\_SFC\_SW\_DWN.apply(MJ\_to\_KJ),

"TMIN": df\_power.T2M\_MIN,

"TMAX": df\_power.T2M\_MAX,

"VAP": df\_power.T2MDEW.apply(tdew\_to\_kpa),

"WIND": df\_power.WS2M,

"RAIN": df\_power.PRECTOTCORR})

return df\_pcse

def ea\_from\_tdew(tdew):

# Raise exception:

if (tdew < -95.0 or tdew > 65.0):

# Are these reasonable bounds?

msg = 'tdew=%g is not in range -95 to +60 deg C' % tdew

raise ValueError(msg)

tmp = (17.27 \* tdew) / (tdew + 237.3)

ea = 0.6108 \* exp(tmp)

return ea

# 单点获取

def main(lat,lon):

start\_date = dt.date(2019, 1, 1)

end\_date = dt.date(2021, 12, 31)

latitude = lat

longitude = lon

powerdata = getnasadata(latitude, longitude, start\_date, end\_date)

if not powerdata:

msg = "Failure retrieving POWER data from server. This can be a connection problem with the NASA POWER server, retry again later."

print(msg)

description = [powerdata["header"]["title"]]

elevation = float(powerdata["geometry"]["coordinates"][2])

df\_power = \_process\_POWER\_records(powerdata)

# Determine Angstrom A/B parameters

angstA, angstB = \_estimate\_AngstAB(df\_power)

# Convert power records to PCSE compatible structure

df\_pcse = \_POWER\_to\_PCSE(df\_power)

# 去除空值，填充空值，加上降雪

SNOWDEPTH = [-999 for i in range((end\_date - start\_date).days + 1)]

std\_datatime = pd.DataFrame(pd.date\_range(start=start\_date, end=end\_date, freq='D'), columns=['DAY'])

data = df\_pcse.dropna()

data['DAY'] = pd.to\_datetime(data['DAY'])

result = pd.merge(std\_datatime, data, how='left', on='DAY')

result['SNOWDEPTH'] = SNOWDEPTH

# 找出缺失值索引

nan\_indexset = set(np.where(np.isnan(result.iloc[:, 1:]))[0])

nan\_index = list(nan\_indexset)

miss\_value = len(nan\_index)

nan\_index.sort()

# 替换空值

for i in nan\_index:

result.loc[i, 'IRRAD'] = result.iloc[i - 5:i + 6, 1].mean()

result.loc[i, 'TMIN'] = result.iloc[i - 5:i + 6, 2].mean()

result.loc[i, 'TMAX'] = result.iloc[i - 5:i + 6, 3].mean()

result.loc[i, 'VAP'] = result.iloc[i - 5:i + 6, 4].mean()

result.loc[i, 'WIND'] = result.iloc[i - 5:i + 6, 5].mean()

result.loc[i, 'RAIN'] = result.iloc[i - 5:i + 6, 6].mean()

# 写文件表头

excelhead = pd.DataFrame({'DAY': ['Site Characteristics', 'Country', 'Station', 'Description', 'Source', 'Contact', 'Missing values', 'Longitude', longitude, 'Observed data', 'DAY', 'date'],

'IRRAD': [np.NaN, 'China', 'miss\_value={}days'.format(miss\_value), description, 'Meteorology and Air Quality Group, Wageningen University', 'Peter Uithol', -999, 'Latitude', latitude, np.NaN, 'IRRAD', 'kJ/m2/day or hours'],

'TMIN': [np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, 'Elevation', elevation, np.NaN, 'TMIN', 'Celsius'],

'TMAX': [np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, 'AngstromA', angstA, np.NaN, 'TMAX', 'Celsius'],

'VAP': [np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, 'AngstromB', angstB, np.NaN, 'VAP', 'kPa'],

'WIND': [np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, 'HasSunshine', False, np.NaN, 'WIND', 'm/sec'],

'RAIN': [np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, 'RAIN', 'mm'],

'SNOWDEPTH': [np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, np.NaN, 'SNOWDEPTH', 'cm']})

dataexcel = pd.concat([excelhead, result], axis=0, ignore\_index=True)

dataexcel.to\_excel(r'D:\Desktop\WOFOST\_AP\Meteorological\_parameter\NASA天气文件lat={},lon={}.xlsx'.format(latitude, longitude),

index=False, header=None)

print('getNASA天气文件lat={},lon={}.xlsx successful'.format(latitude, longitude))

def st\_loc(num):

"""

选取气象数据的时候标准化输入为0.5的分辨率

"""

import math

if num-math.floor(num) <=0.25 or num-math.floor(num)>=0.75:

result = round(num)

elif 0.25 < num-math.floor(num) < 0.75:

result = math.floor(num) + 0.5

return result

if \_\_name\_\_ == '\_\_main\_\_':

gewang = pd.read\_excel(r'E:\2022.11.21毕业论文\3实验过程\气象数据\格点.xls')

columns = gewang.columns

res = []

for c in columns:

d = gewang[c].tolist()

res.append(d)

for i in range(len(res[0])):

main(st\_loc(res[1][i]),st\_loc(res[0][i]))

最终的得到以excel表格形式保存的天气数据。如：NASA天气文件lat=30.5,lon=112.5.xlsx。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Site Characteristics |  |  |  |  |  |  |  |
| Country | China |  |  |  |  |  |  |
| Station | miss\_value=0days | |  |  |  |  |  |
| Description | ['NASA/POWER CERES/MERRA2 Native Resolution Daily Data'] | | | | | | |
| Source | Meteorology and Air Quality Group, Wageningen University | | | | | | |
| Contact | Peter Uithol | |  |  |  |  |  |
| Missing values | -999 |  |  |  |  |  |  |
| Longitude | Latitude | Elevation | AngstromA | AngstromB | HasSunshine | |  |
| 112.5 | 30.5 | 43.15 | 0.102888 | 0.607144 | FALSE |  |  |
| Observed data |  |  |  |  |  |  |  |
| DAY | IRRAD | TMIN | TMAX | VAP | WIND | RAIN | SNOWDEPTH |
| date | kJ/m2/day or hours | Celsius | Celsius | kPa | m/sec | mm | cm |
| 2019-01-01 00:00:00 | 5730 | -5.17 | 4.3 | 0.389351 | 1.57 | 0.17 | -999 |
| 2019-01-02 00:00:00 | 4770 | -3.57 | 5.68 | 0.434473 | 1.01 | 0 | -999 |
| 2019-01-03 00:00:00 | 3680 | -2.22 | 5.87 | 0.566503 | 1.03 | 0.28 | -999 |
| 2019-01-04 00:00:00 | 2580 | -0.63 | 7.15 | 0.669134 | 2.08 | 1.65 | -999 |
| 2019-01-05 00:00:00 | 2470 | -2.13 | 7.02 | 0.525072 | 1.75 | 0.01 | -999 |
| 2019-01-06 00:00:00 | 5240 | -3.21 | 7.22 | 0.55619 | 1.35 | 0 | -999 |
| 2019-01-07 00:00:00 | 2100 | -2.07 | 8.08 | 0.561119 | 2.45 | 0 | -999 |
| 2019-01-08 00:00:00 | 3370 | -2.73 | 5.51 | 0.510099 | 2.29 | 6.07 | -999 |
| 2019-01-09 00:00:00 | 2910 | -0.78 | 2.98 | 0.589359 | 2.38 | 6.49 | -999 |
| 2019-01-10 00:00:00 | 3070 | -2.67 | 3.11 | 0.53725 | 2.16 | 0.46 | -999 |
| 2019-01-11 00:00:00 | 2490 | -2.71 | 6.79 | 0.552518 | 1.16 | 1.51 | -999 |
| 2019-01-12 00:00:00 | 4240 | -2.52 | 7.91 | 0.618848 | 0.77 | 0.4 | -999 |
| 2019-01-13 00:00:00 | 5220 | -1.06 | 10.88 | 0.677857 | 1.32 | 0.07 | -999 |
| 2019-01-14 00:00:00 | 2150 | 1.21 | 8.69 | 0.696104 | 2.28 | 1.54 | -999 |
| 2019-01-15 00:00:00 | 1900 | -3.17 | 8.43 | 0.467073 | 3.82 | 0.01 | -999 |
| 2019-01-16 00:00:00 | 10870 | -4.85 | 6.5 | 0.320303 | 1.68 | 0 | -999 |
| 2019-01-17 00:00:00 | 13140 | -2.11 | 8.58 | 0.462883 | 1.32 | 0.01 | -999 |
| 2019-01-18 00:00:00 | 4670 | -1.4 | 8.55 | 0.677857 | 1.84 | 0.13 | -999 |
| 2019-01-19 00:00:00 | 2790 | 1.15 | 8.65 | 0.70817 | 1.78 | 0.2 | -999 |
| 2019-01-20 00:00:00 | 5570 | -2.31 | 10.77 | 0.513902 | 3.05 | 0 | -999 |
| 2019-01-21 00:00:00 | 10820 | -1.36 | 11.43 | 0.472003 | 1.72 | 0 | -999 |
| 2019-01-22 00:00:00 | 15910 | -1.31 | 14.68 | 0.518114 | 1.65 | 0 | -999 |
| 2019-01-23 00:00:00 | 14610 | -1.19 | 15.54 | 0.668171 | 1.96 | 0 | -999 |

# 土壤数据

利用经验公式转换

## 下载官方土壤数据

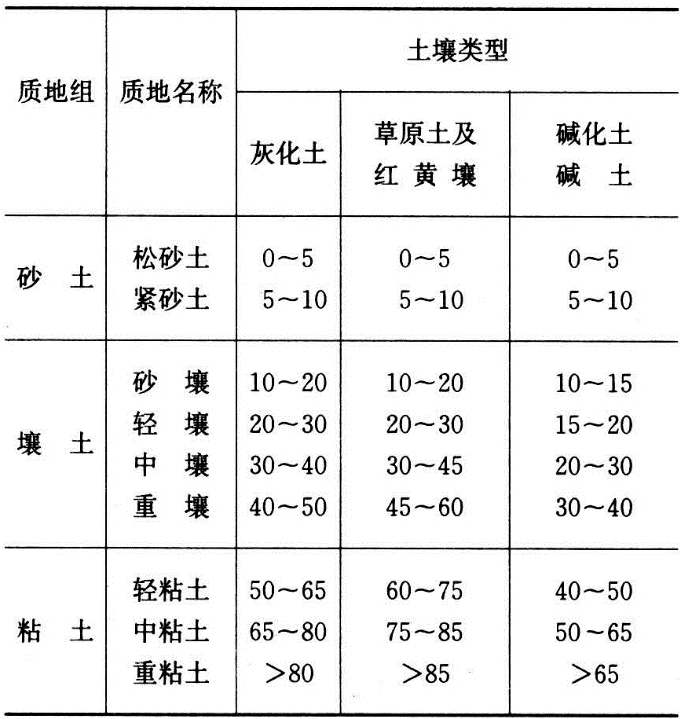
在GitHub中下载ec1.new到ec6.new土壤数据。

|  |  |  |
| --- | --- | --- |
| 土壤名称 | 翻译 | 可能对应的质地 |
| EC1-coarse | 粗粒质土壤 | 砂土 |
| EC2-medium | 中等质地土壤 | 砂壤 |
| EC3-medium fine | 中等精细土壤 | 轻壤 |
| EC4-fine | 精细土壤 | 重壤 |
| EC5-very fine | 非常精细土壤 | 粘土 |
| EC6-fine and permeable | 精细与透水土壤 | 中壤 |

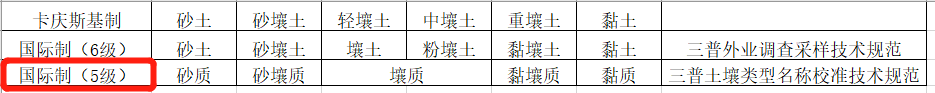
## 质地与土壤数据的转换

[https://www.pwsannong.com/c/2016-04-13/567503.shtml#h23](https://www.pwsannong.com/c/2016-04-13/567503.shtml" \l "h23)

土壤数据中的土壤质地数据为卡钦斯基制，是按直径小于0.01毫米的物理性粘粒含量划分（简制），如下表。



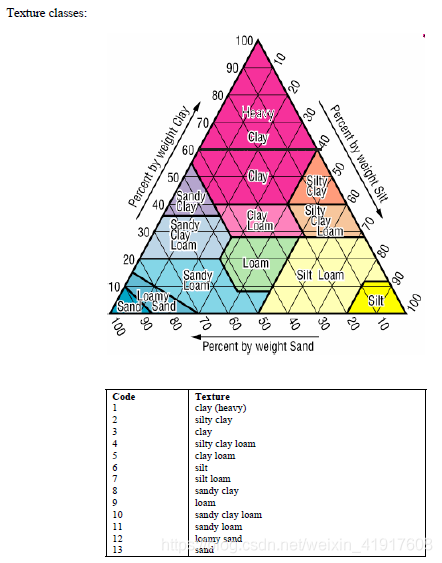
国际制和苏联制对应关系



中国制土壤质地



美国制三角：



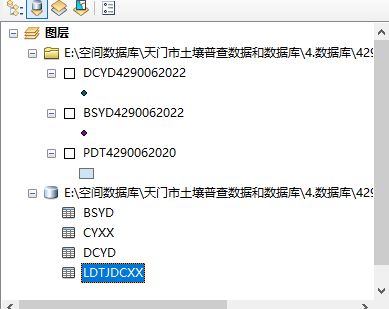
Cite: NNU\_Group (2011). 美国土壤分类级别, Concept & Semantic, OpenGMS, <https://geomodeling.njnu.edu.cn/repository/concept/73ad0abc-753c-4e2a-aba6-80135cf0367d>

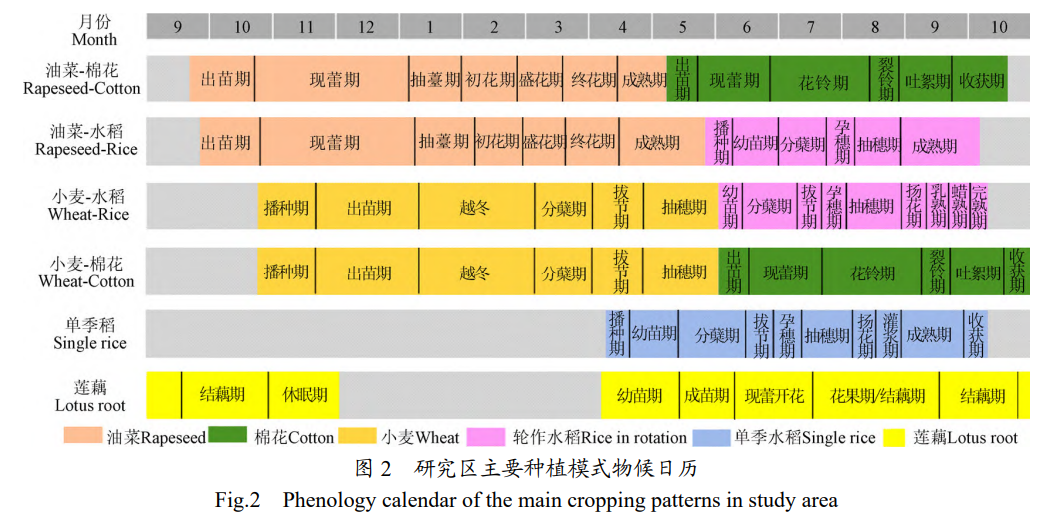
质地数据来源：全球土壤数据库

## 转换代码

# 作物数据

天门市种植制度：油菜-早稻-晚稻、小麦-水稻（农用地质量分等规程GB/T28407-2012）





## 作物参数本地化

单季稻：

双季稻：

小麦：

油菜：

## 作物物候信息

### 数据准备

#### 采样点数据

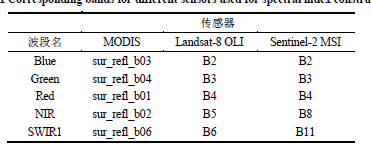
经纬度、

#### 种植区域

耕地地块信息

#### 定义相关参数



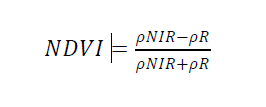


EVI：增强植被指数（EVI）是反映植被生理参数的重要指标，通过对蓝光波段、红光波段及近红外波段进行运算组合，从而减小大气气溶胶散射对植被特征的影响。EVI 选用波段范围较窄的红光和近红外波段，能够最大程度减少植被冠层结构和土壤对作物反射光线的干扰，提高其在植被密集区的敏感性，同时可以更有效的描述植被生长状况。



式（3-1）中，ρB、ρR、ρNIR 分别对应B2蓝光波段、B4红光波段和B8近红外波段的反射率值。EVI指数值范围是-1到1，其中绿色植被的取值范围通常在0.2到0.8之间。

NDVI：归一化植被指数（NDVI）是近红外波段和红光波段的非线性组合[87]。NDVI 通过增强低值区域，可以将农作物与水体、居民地等地物明显区分，但同时抑制了高值区域，因此在植被密集区的灵敏性不如其它植被指数。NDVI能够部分消除云雾和地形对地物反射特征的影响，是目前应用最广泛的植被指数。



其中ρR、ρNIR 分别对应B4 红光波段和B8 近红外波段的反射率值。NDVI取值范围是-1到1，处于生长期的农作物指数值通常在0.2-0.8 之间。

WRI：能够捕捉到油菜在开花前后的特征变化，且适用于多种作物的区分。（Zhang H, Liu W, Zhang L. Seamless and automated rapeseed mapping for large cloudy regions using time-series optical satellite imagery [J]. ISPRS Journal of Photogrammetry and Remote Sensing, 2022, 184: 45-62.）



NDWI：用于监测水体，可以帮助识别灌溉情况



### 数据处理

//分类方法选择随机森林

var rf = ee.Classifier.smileRandomForest({

numberOfTrees: 40,

bagFraction: 0.8

}).train({

features: train\_data,

classProperty: 'landcover',

// inputProperties: inputbands

});

//对哨兵数据进行随机森林分类

var img\_classfication = train\_clipped.classify(rf);

//运用测试样本分类，确定要进行函数运算的数据集以及函数

var test = testingPartition.classify(rf);

//计算混淆矩阵

var confusionMatrix = test.errorMatrix('landcover', 'classification');

print('confusionMatrix',confusionMatrix);//面板上显示混淆矩阵

print('overall accuracy', confusionMatrix.accuracy());//面板上显示总体精度

print('kappa accuracy', confusionMatrix.kappa());//面板上显示kappa值

Map.addLayer(img\_classfication.clip(roi\_TM), {min: 1, max: 3, palette: ['red', 'blue', 'green']});

var class1=img\_classfication.clip(roi\_TM)

//导出分类图

Export.image.toDrive({

image: class1,

description: 'rfclass',

fileNamePrefix: 'rf\_TM', //文件命名

folder: "class", //保存的文件夹

scale: 10, //分辨率

region: roi\_TM, //研究区

maxPixels: 1e13, //最大像元素，默认就好

crs: "EPSG:4390" //设置投影

});

### 作物物候信息信息提取

采用MODIS\_LAI时序数据反演

# 农田管理数据

## 灌溉数据

农业灌溉需要消耗大量的可利用淡水资源，是人类对自然水循环过程最直接的扰动，加速了区域水循环的同时伴随着冷却作用。因此，估算灌溉用水对于探索人类活动对自然水循环的影响、量化水资源收支、优化农业水资源管理配置等具有重要意义。然而，目前灌溉用水数据主要是基于调查统计结果，数据空间分布离散且缺乏统一性，无法满足对灌溉用水的时空变化进行估算的需求。全球灌溉农田灌溉用水量遥感估算数据集（2011-2018）是基于卫星土壤湿度、降水、植被指数以及气象资料入辐射与气温等要素，通过土壤水量平衡原理，耦合遥感蒸散发过程模块以及利用基于差分优化的数据-模型融合算法来估算全球灌溉农田实际灌溉用水量。该数据集的灌溉用水估算结果相比传统的离散调查统计数据在不同空间尺度（区域、州/省和国家）上具有较小的偏差，如中国各省2015年农业用水统计结果对比（bias = −3.10 km^3），美国各州2013年调查数据结果对比（bias = −0.42 km^3）以及粮农组织各个国家尺度对比结果（bias = −10.84 km^3）。而且，相较于基于单个降水和土壤水分卫星产品的估算结果，该集合数据显示出更低的不确定性。此外，数据统一采用全球地理经纬度格网，相关元数据存储在对应的NetCDF文件内，空间分辨率约为25公里，时间分辨率为月尺度，时间跨度为2011年−2018年。该数据集将有助于定量评估历史时期农业灌溉用水的时空格局和支撑科学农业用水管理等。

文件命名：格网灌溉用水量以netcdf格式存储，文件的名称为“IWU\_ens\_Yyyyy”，其中yyyy代表年。比如IWU\_ens\_Y2011.nc就代表这个netcdf文件描述2011年逐月全球灌溉用水量的格网数据。

张琨, 李新, 郑东海, 张凌, 朱高峰. (2021). 全球灌溉农田灌溉用水量遥感估算数据集（2011-2018）. 国家青藏高原数据中心. https://doi.org/10.11888/Hydro.tpdc.271220. <https://cstr.cn/18406.11.Hydro.tpdc.271220>.

Zhang, K., Li, X., Zheng, D., Zhang, L., Zhu, G. (2021). Satellite-based Global Irrigation Water Use data set (2011-2018). National Tibetan Plateau / Third Pole Environment Data Center. https://doi.org/10.11888/Hydro.tpdc.271220. <https://cstr.cn/18406.11.Hydro.tpdc.271220>.

# 站点数据

## 土壤养分含量

土壤供肥量=土壤养分测定值（mg/kg）×0.15×校正系数式中，0.15为换算系数，即把1mg/kg的速效养分，按每亩表土15万千克换算成每亩土壤养分量（千克/亩）

校正系数设置为0.8