python实现儒略日转公历时间

原创 LXXtime 遥感小屋 2024年04月26日 00:00 天津

LXX

读完需要 5 分钟 速读仅需 2 分钟

前言：

一些遥感数据或卫星测高数据的获取时间不是我们常见的‘年月日’等形式，因此再进行时间序列分析时需要转换时间。本文分享将儒略日转为公历的原理及实现。

希望各位同学点个关注，点个小赞，这将是更新的动力，不胜感激(^_-)

1

预备知识

儒略日：

是一个不涉及年月等概念的长期连续的记日法，在天文学，空间大地测量和卫星导航定位种经常使用。儒略日的起点定在公元前 4713 年（天文学上记为-4712 年）1 月 1 日世界时 12：00 即 JD 0 ，国际时长的 24 小时 TU24 为 1 天 即 JD 1（4713BC1 月 1 日 12：00 到 4713BC1 月 2 日 12：00）而此时儒略日的规则时 分平年和闰年 每 4 年一闰 平年 365 闰年 366 直到 1582 年 10 月 4 结束 开始用格力高历，就是现在的公历规则 每 4 年一闰， 百年不闰，400 又闰。但是格力高历其实抹掉了 10 天（1582，10 月 4 下一日直接跳到 10 月 15 日）

GPS 时：

GPS 系统内部采用的时间系统是 GPS 时间，是以 1980 年 1 月 6 日为起点,用周数（一个星期 7 天）和周数中的秒计算。

各个时间的想换转换都有严格的转换公式，其转换的主要思想是把通用时，GPS 时间转换为儒略日，然后以儒略日为媒介，实现任意两个时间系统的相互转换

儒略日的计算是从公元前 4713 年 1 月 1 日平午 12:00 开始的天数积累，这是儒略日计算的起点，儒略历的规则为，分平年和闰年每 4 年一润各月份对应天数见表 1。儒略历一直使用到 1582 年 10 月 4 结束，因为存在误差改为格力高历，从 1582 年 10 月 15 日开始

计算，使用格里高历，就是现在使用的公历。即公元纪年法在 1582 年 10 月 4 日及以前使用儒略历，在 1582 年 10 月 15 日及以后使用格里高历，格里高历闰年与平年月份-天数与儒略历一致。不同的国家使用格里高历的时间可能不一样，这个要注意，比如我们国家是新中国成立后开始使用的。

表1 月份-天数对应表

月份	1	2	3	4	5	6	7	8	9	10	11	12
天数（平年）	31	28	31	30	31	30	31	31	30	31	30	31
天数（闰年）	31	29	31	30	31	30	31	31	30	31	30	31

2

计算分析

《Astronomical Algorithms》同一章节给出了求算方法。以下是每一个表达式的含义。

这本书电子版也提供给大家：

```
链接：https://pan.baidu.com/s/1uMLmT_LNbmYBT6mZ1ZsrQg
提取码：rn1p
--来自百度网盘超级会员V6的分享
```

Calculation of the Calendar Date from the JD

The following method is valid for positive as well as for negative years, but not for negative Julian Day numbers.

Add 0.5 to the JD, and let Z be the integer part, and F the fractional (decimal) part of the result.

If $Z < 2299161$, take $A = Z$.

If Z is equal to or larger than 2291161, calculate

$$\alpha = \text{INT}\left(\frac{Z - 1867216.25}{36524.25}\right)$$

$$A = Z + 1 + \alpha - \text{INT}\left(\frac{\alpha}{4}\right)$$

Then calculate

$$B = A + 1524$$

$$C = \text{INT}\left(\frac{B - 122.1}{365.25}\right)$$

$$D = \text{INT}(365.25\,C)$$

$$E = \text{INT}\left(\frac{B - D}{30.6001}\right)$$

The day of the month (with decimals, if any) is then

$$B - D - \text{INT}(30.6001\,E) + F$$

The month number m is $E - 1$ if $E < 14$; $E - 13$ if $E = 14$ or 15

The year is $C - 4716$ if $m > 2$; $C - 4715$ if $m = 1$ or 2

Contrary to what has been said about formula (7.1), in the formula for E the number 30.6001 may *not* be replaced by 30.6, even if the computer calculates exactly. Otherwise, one would obtain February 0 instead of January 31, or April 0 instead of March 31.

1.由于儒略日的历元为正午 12 时，公历历元为半夜 12 时，为统一计算，将儒略历历元前推至 0.5 日。

```
即为：JD = JD + 0.5
```

2.其中整数部分为日数，小数部分为时刻，只对整数部分进行运算。则令：

```
Z = floor(JD)F = JD - Z
```

Z 即为所求日到-4712 年 1 月 1 日 0 时的日数。

3.由于儒略历和格里历的岁长不同，需分别处理。即自 1582 年 10 月 15 日 0 时前适用儒略历（岁长 365.25），此后适用格里历（岁长 365.2425）。为统一计算，可将格里历转换为儒略历，即假设自-4712 年 1 月 1 日 0 时起一直使用的是儒略历。则针对格里历相对儒略历少置闰（400 年 3 闰）的部分给予补上。对于使用取整公式计算置闰，只能以一个 400 年置闰周期的起点为岁首，如 1600 年 3 月 1 日，根据儒略日计算公式，得该日的儒略日为 2305507.5。

格里历除能被 400 整除的百年为 36525 日外，其余每百年仅 36524 日，此时分母较大，故分子每百年应增加 0.25 日，300 年计 0.75 日。前对 JD 取整后所得 Z 值，仍以 12 时为历元，故仅需再增 0.25 日。

```
a = floor((Z - 2305507.25) / 36524.25)
```

再补上 1582 年 10 月 4 日到 10 月 15 日跳过的 10 天，即为自-4712 年 1 月 1 日 0 时到所求日以儒略历计算的总积日。

```
A = Z + 10 + a - floor(a/4)
```

（注：书中 a 和 A 的表达式与此略有差异，是由于书中将历元推至公元元年 1 月 1 日，则至 1582 年间多置闰了 12 次，去除跳过的 10 天，共多了 2 日，反映在置闰公式中，相当于多计算了 200 年，由于在置闰公式（floor(y/100)-floor(y/400)）中，对 200 年间的置闰可能有 2 次或 1 次 2 种情况，宜推算到一次置闰周期即 400 年，则必然置闰 3 次，而多算了一次，则在表达式 A 中补足。即为调整"儒略历和格里历的置闰误差"以及"不识岁差和回归年过大的误差"再次将历元从公元元年推至公元 400 年。这种情况思虑比较复杂，可不使用书中的表达式。）

同样地，为避免对负数取整的情况，将历元前推至-4716 年 3 月 1 日 0 时，需补上相差的日数，合 1524 日。即有：

```
B = A + 1524
```

对所得的积日（儒略历），除以岁长，即为积年（表达式 C），加上历元即为所求公历年份。其中整数部分为年的积日（表达式 D），小数部分为月与日的积日（表达式 B-D）。

```
C = floor((B-122.1)/365.25)

D = floor(365.25*C)
```

将 B-D 除以每月平均日数 30.6 为积月（表达式 E），其中整数部分为月数，小数部分为日数（day）。

最后调整岁首的情况可得 month 和 year。

宜需考虑，day 的结果小于 0.5 时，即在上月末日，此时亦可能导致 E 值小于 1，即在上年末月。需要分别判断处理。但也可先求 JD-1 日，得到正常结果，在结果上加回减去的 1 日即可。

> 儒略日转公历
>
> https://blog.csdn.net/weixin_42763614/article/details/82880007

> 公元纪年法（儒略历-格里高历）转儒略日
>
> https://wxj233.blog.csdn.net/article/details/112244135?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-112244135-blog-52577940.235%5Ev43%5Epc_blog_bottom_relevance_base7&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-112244135-blog-52577940.235%5Ev43%5Epc_blog_bottom_relevance_base7&utm_relevant_index=6

3

实现代码

```
import numpy as np

def convert_julian(JD, ASTYPE=None, FORMAT='dict'):
    # 将输入的值转换为数组，如果只有单个值，则转换为数组形式
    if (np.ndim(JD) == 0):
        JD = np.array([JD])
        SINGLE_VALUE = True
    else:
        SINGLE_VALUE = False

    JDO = np.floor(JD + 0.5)
    C = np.zeros_like(JD)
    # 计算格里历日历切换前后的C值
    IGREG = 2299161.0
    ind1, = np.nonzero(JDO < IGREG)
    C[ind1] = JDO[ind1] + 1524.0
    ind2, = np.nonzero(JDO >= IGREG)
```

```
    B = np.floor((JDO[ind2] - 1867216.25)/36524.25)
    C[ind2] = JDO[ind2] + B - np.floor(B/4.0) + 1525.0
    # 计算日期转换的系数
    D = np.floor((C - 122.1)/365.25)
    E = np.floor((365.0 * D) + np.floor(D/4.0))
    F = np.floor((C - E)/30.6001)
    # 计算日、月、年和小时
    DAY = np.floor(C - E + 0.5) - np.floor(30.6001*F)
    MONTH = F - 1.0 - 12.0*np.floor(F/14.0)
    YEAR = D - 4715.0 - np.floor((7.0+MONTH)/10.0)
    HOUR = np.floor(24.0*(JD + 0.5 - JDO))
    # 计算分钟和秒
    G = (JD + 0.5 - JDO) - HOUR/24.0
    MINUTE = np.floor(G*1440.0)
    SECOND = (G - MINUTE/1440.0) * 86400.0

    # 将所有变量转换为输出类型（从浮点数）
    if ASTYPE is not None:
        YEAR = YEAR.astype(ASTYPE)
        MONTH = MONTH.astype(ASTYPE)
        DAY = DAY.astype(ASTYPE)
        HOUR = HOUR.astype(ASTYPE)
        MINUTE = MINUTE.astype(ASTYPE)
        SECOND = SECOND.astype(ASTYPE)

    # 如果最初只导入了单个值：删除单例维度
    if SINGLE_VALUE:
        YEAR = YEAR.item(0)
        MONTH = MONTH.item(0)
        DAY = DAY.item(0)
        HOUR = HOUR.item(0)
        MINUTE = MINUTE.item(0)
        SECOND = SECOND.item(0)

    # 根据输出格式返回日期变量（默认为Python字典）
    if (FORMAT == 'dict'):
        return dict(year=YEAR, month=MONTH, day=DAY,
                    hour=HOUR, minute=MINUTE, second=SECOND)
    elif (FORMAT == 'tuple'):
        return (YEAR, MONTH, DAY, HOUR, MINUTE, SECOND)
    elif (FORMAT == 'zip'):
        return zip(YEAR, MONTH, DAY, HOUR, MINUTE, SECOND)

if __name__ == '__main__':
    JDs = [2400000.5, 2459893.4, 2459694.8]  # 以儒略日为例
    for JD in JDs:
```

```
        result = convert_julian(JD)
        print(f"{int(result['year'])}年{int(result['month'])}月{int(result['d
        f"{int(result['hour'])}时{int(result['minute'])}分{int(result['second
```

输出结果：

LXXtime

喜欢作者