Usage Documentation of Astronomical Python Package

此说明文档是关于一系列天文的Python包的应用说明。

我们根据GitHub/国外高校/Python库等所提供的各类开源python-Astro库，将其简要翻译成通俗易懂且又不失专业素养的中文，并在某些重要的章节提供具体详细的例子说明和代码注释，供各类天文学科研究的程序员/本科生/研究生/天文爱好者使用。

此说明文档一共包含了17个有关天文的Python包，应用方面涵盖天文计算、单位换算、天文绘图、图像处理、WCS坐标系统、统计工具、星历计算、宇宙学应用、光谱处理、表格处理、数据库处理等。其中绝大多数都是尽5年才发布的，其中Astropy(V3.0) 是在2017年发布的，因此基本上都是科学研究最前沿的成果和最高效率的Astro-Python源代码。我们将其翻译成中文，一方面是出于方便各类天文工作者的使用，另一方面是为了推广和促进人工智能在天文学科中的应用。

作者:

陈缘 北京大学 天文系

林辞楚 成都信息工程大学 软件工程专业

谭哲韬 成都信息工程大学 大气科学专业

由于我们的水平有限，本说明文档中还存在着或多或少的不足和需要改进的地方，对一些专业名词略有模糊，还望大家多多指正，可发邮件至：cyril131747@gmail.com 或 zqtzt@126.com

2018年4月15日

索引:

由于本说明书涵盖众多功能，为方便大家使用和搜索，我们制作了索引表，供大家使用和参考

Package Name	Main Functions
Astropy	综合Astropysics包在内的大型综合天文科学计算包
Astropysics	简化、分析和可视化天文数据
AstLib	天文绘图、统计数据计算、坐标转换以及通过PyWCSTools操作带有世界坐标系统（WCS）信息的FITS图像的工具。
APLpy	天文绘图操作
Cosmocalc	宇宙学计算和天文常量、导出量的计算
Kapteyn	使用世界坐标绘制图像数据
Pyflation	模拟早期宇宙中的扰动
Galpy	银河动力学模拟
Alipy	自动识别光学天文图像之间几何变换
Pyregion	解析ds9区域文件
ChiantiPy	使用CHIANTI原子数据库执行天体物理光谱
PyEphem	天文历法计算
Pysofa	通过XPA与来自pythonshell的DS9进行通信协议
Spctrum	估计功率谱密度
pywcsgrid2	与matplotlib一起结合使用来显示天文数字图像
AperturePhotometry	孔径测光
Metpy	用于行星大气（大气层）研究

前言

作为一名刚接触科研不久的天文系三年级本科生，我不敢说自己对python有多么精通、对它在天文中的应用有多么了解，但是python对天文研究的重要性，甚至在我踏入燕园之前就已有耳闻。还记得高考后的那个暑假，远在MIT的高中天文社前辈不断叮嘱我：想做好天文研究，一定要学好python。他还说自己能进入MIT，很大程度上是因为教授看中了自己丰富的编程经验。两年之后，当我终于有勇气加入一名K所 (北京大学科维理天文与天体物理研究所的简称) 教授的研究组时，这位和蔼的美国人对我说：“虽然我只会用IDL，但我的学生必须要会用python”。

但是，除了大一两学期与物理系合上的C/C++课程外，天文系并没有单独开设有关数据处理和python编程的课程，所以基本上只能靠自学。但据我所知，北师大天文系有开设python课程，中大天文系在大一便有误差分析和数据处理课。事实上，python并不是一个历史悠久的工具，大部分已经拿到教职或还处于tenure track的天文科研人员仍然主要使用的是IDL。但不可否认的是，功能强大的python将会（甚至已然）成为天文界最重要、最常用的编程语言。作为新生代科研力量的一份子，学习并熟练掌握python是至关重要的一环。

其实，只要有了学习第一门编程语言的经验，再接触第二门时会顺利许多。况且相较于C语言，python更适合自学。首先它自身的语法十分简洁，代码长度可能要比C语言短上十倍。更重要的是，这个语言设计的重点不在于“造轮子”，而在于“用轮子”：我们不需要知道五花八门的算法和高超的编程技巧，只需要知道哪些python库有些什么样的功能，然后通过简单的代码将他们导入并组合起来以实现目的即可。

虽然python也有缺点，即速度相较C慢很多，但它的优点依然使其在各个领域的应用日趋广泛。其中天文研究绝对是一个能让python大展身手的地方，目前已经有许多软件包 (package/module) 收集了大量与天文研究相关的功能。以我作为本科生的接触到的科研工作来说，主要有以下三个方面的应用：

1. 表格数据的处理。尤其在观测领域，数据通过表格呈现，即使对于文件格式都是.txt，内部的表格格式也千差万别。如果只用python自带的一些读取.txt文件的函数，参数设置和后续处理都会比较麻烦。但实际上，`astropy.io.ascii` 已经写好了处理各种常见表格的程序，我们只需要选择相应的表格类型，就可以将表内数据以一种掩码表(masked table)的格式读入，这种格式无论对索引还是计算来说都十分方便。
2. 可视化与绘图。发表在专业期刊上的文章必定有各式各样高端大气上档次的配图，基本来源于两大类：
   - 带标识（如文本、轮廓线）的望远镜照片
   - 数据处理或数值模拟的可视化结果

   而为了绘制这些图片，我们可以从 `matplotlib` 里找到几乎所有需要的功能，也可以从天文软件包里调用现成的绘图功能。总之，只需掌握一些基本的使用方法，我们就可以像艺术家一样绘制出属于我们的科研图片。
3. 与python接洽的数值模拟程序。鉴于python在运行速度上的劣势，大型的数值运算应选择速度快的C、Fortune等语言。对于某些规模不太大且应用需求大的程序，可以用python包装成软件包，便于在python程序中进行调用。这样我们省去了分别运行的时间，使得程序更加统一、整洁。

   在上大学以前，我几乎完全没有接触过编程（初中学的VB早就忘光了），而且似乎天生也不擅长于此，所以在迫不得已地开发这项技能的过程中，也算是兜兜转转绕了不少弯路。首先，在学习python基础知识时，因为疲于应付课业，战线拉得过长，上手编程的机会也不多，学过的东西记不牢，学新知识时也感觉比较吃力。此外，我刚开始接触科研时并不是十分习惯看全英文帮助档案，有问题不知道去哪找答案、有想实现的功能不知道从何入手，一度觉得进度缓慢、效率低下。后来经历了大半年的摸爬滚打，才慢慢找到了门道。但即便如此，当我的两位组员找到我，拿出一份天文软件包的“名单”时，我才发现原来这条路上还隐藏着许多有价值的工具，其中有个功能甚至就是我之前花了一个假期才实现的。**这份文档的初衷，是帮助像我一样刚刚踏上天文研究道路的年轻学生/科研人员/天文爱好者知晓已有的天文软件包，并在实际的科研工作中利用它们。同时，也能帮助已有一定经验的科研人员开拓视野，发现更多可用、好用的模块。**

陈缘

2018.5.1 于 KIAA 一楼会议室

目录：

1. AstroPy

> 综合Astropysics包在内的大型综合天文科学计算/绘图包

1.1 数据结构和转换

> Data Structure & Transformations
>
> 以下内容比较重要，但基本astrophysics重合，就不在这一阶段整理了。如果需要我这里也有详细的个人笔记。

- 常数 (`astropy.constants`)
- 单位和物理量 (`astropy.units`)
- N维数据库 (`astropy.nddata`)
- 数据表格 (`astropy.table`)
- 时间与日期 (`astropy.time`)
- 天文坐标系统 (`astropy.coordinates`)
- 世界坐标系统 (world coordinate system) (`astropy.wcs`)]
- 模型与拟合 (`astropy.modeling`)

1.1.1单位 `.units` & 量 `Quantities` ☆

(1) 单位

- `astropy.units` 处理对量的"定义、转换、运算"，包括常见的物理量和星等、分贝之类，但不处理天球坐标（由另一个包专门处理）
- 单位化简：`q.decompose(bases[])`，把 Quantity 对象的单位化简到不能约化的形式，`bases` 参数是指定的化简后的单位，需传入 UnitBase sequence

```
>>> q = 3.0 * u.kilometer / (130.51 * u.meter / u.second)
0.022986744310780783 km s / m
```

```
>>> q.decompose()
22.986744310780782 s
```

- 自定义单位：内置单位相乘除

```
cms = u.cm / u.s
from astropy.units import imperial  # 英制单位
mph = imperial.mile / u.hour
```

相同的单位相除会变成"无量纲"，用 `''` 表示

```
>>> u.m / u.m
Unit(dimensionless)
```

(2) 量

- `Quantity` 包含数值 (value) 和单位 (unit) 属性
- 创建 Quantity 对象：
  - 用数值 (scalar, list, array均可) **乘或除一个内置单位** (built-in units)

    ```
    >>> 42.0 * u.meter
    <Quantity 42.0 m>
    >>> [1., 2., 3.] * u.m
    <Quantity [ 1., 2., 3.] m>
    ```

  - 直接用 `u.Quantity()` 方法创建，功能更多，要指定 value 和 unit

    ```
    >>> u.Quantity(15, u.m / u.s)
    <Quantity 15.0 m / s>
    >>> u.Quantity('15 m/s')     # 可自动分析字符串
    <Quantity 15.0 m / s>
    ```

- 获得 Quantity 对象的属性 `.value`、`.unit`：

  ```
  >>> q = 42.0 * u.meter
  >>> q.value
  42.0
  >>> q.unit
  Unit("m")
  ```

- 输出：可以用 `Format String Syntax` 设置打印格式

  ```
  # 指定输出的小数点位数
  >>> "{0:0.03f}".format(q)
  '0.472 m / s'
  ```

  ```
  # 不仅可以设置value的格式，还能设置unit的格式!
  >>> "{0.value:0.03f} {0.unit:FITS}".format(q)
  '0.472 m s-1'
  ```

  注：默认情况下输入的数据会被拷贝(copied)，在数组比较大的时候可以考虑在 `view()` 的时候设置 `copy = False`

- 算术运算
  - 加减法：只在量纲相同的情况下才可进行

    若单位不同，自动按照最左边的单位进行换算

    不支持 Quantity 对象和普通数字类型之间的运算（无量纲量除外）
  - 乘除法：支持任何量纲的 Quantity 对象以及与普通数字类型之间的运算

    可以与单位转换、单位化简联合使用
- Quantity 对象是从 `numpy.ndarray` 继承来的，可以使用父类几乎所有的方法

```
# 数组的统计方法
>>> q = np.array([1., 2., 3., 4.]) * u.m / u.s
>>> np.mean(q)
<Quantity 2.5 m / s>
>>> np.std(q)
<Quantity 1.118033988749895 m / s>

# 函数（三角、指数、...）
>>> q = 30. * u.deg
>>> np.sin(q)
<Quantity 0.49999999999999994>
```

第2行q数组经转化后每个元素都是 Quantity 对象，但要注意 units 不能放在 `array()` 里面！

```
>>> np.array([1, 2, 3] * u.m)
array([ 1.,  2.,  3.])
>>> np.array([1, 2, 3]) * u.m
[1, 2, 3]m
```

- 转化为 python 普通标量
  - `float()` 只对无量纲量有用，否则会报错：

```
>>> float(3. * u.m)
TypeError: only dimensionless scalar quantities can be converted to Python scalars
>>> float(3. * u.km / (4. * u.m))
750.0
```

  - 更方便的方法 `q.to_value()`，默认把 Quantity 对象转换成 value，参数可填想转换的单位
- 支持 Quantity 对象的绘图
  - 绘图前要声明：

```
>>> from astropy.visualization import quantity_support
>>> quantity_support()
<astropy.visualization.units.MplQuantityConverter ...>
```

  - 之后用 plot() 绘图时，会自动把单位标到坐标轴上，但如果有多组数据画在一张图里，会自动转换成按照最先画的单位；且如果每组数据量纲不统一会报错 `UnitConversionError`
- 可以用 `with quantity_support():` 语句块确保 unit support 在使用后被关闭

(3) 单位转换 & 换算

- 单位转换
  - `q.to()` 方法，传入的参数需为 UnitBase 对象
  - 换单位制：

```
>>> q = 2.4 * u.m / u.s
>>> q.si    # 国际制
<Quantity 2.4 m / s>
>>> q.cgs   # 高斯制
<Quantity 240.0 cm / s>
```

注意有些单位是不能直接转换的，会报错 `UnitConversionError`（这时候就需要用到下面几点介绍的 equivalence）

```
UnitConversionError: 'nm' (length) and 'Hz' (frequency) are not convertible
```

- `Equivalencies`：提供在不同环境 (context) 下的单位换算
  - 本质是元组的列表，每个元组包含4个元素：`(from_unit, to_unit, forward, backward)`
  - 直接的用法是人为设定两单位等价：`q.to(equivalencies = [unit1, unit2])`
  - 用 `.find_equivalent_units` 方法可查询某个单位与其他单位的默认转化方式
  - 详情参考：http://docs.astropy.org/en/stable/units/equivalencies.html#unit-equivalencies
- 自定义 Equivalency（略）
- 下面介绍几个 **"built-in equivalencies"**（有些看起来暂时用不到s就没写）
  - **视差** `.parallax()`：角度 ⇔ 距离

    ```
    >>> (8.0 * u.arcsec).to(u.parsec, equivalencies=u.parallax())
    <Quantity 0.125 pc>
    ```

  - **光谱** `.spectral()`：频率、波长、波数、能量之间的换算

    ```
    >>> ([1000, 2000] * u.nm).to(u.Hz, equivalencies=u.spectral())
    <Quantity [2.9979246e+14, 1.4989623e+14]Hz>
    ```

  - **多普勒效应**：速度 ⇔ 频率

    f_0 静止频率，f 实测频率，V 天体运动速度，c 光速

    以上三个式子分别应用于 `u.doppler_radio()`、`u.doppler_optical()`、`u.doppler_relativistic()`：

    ```
    >>> restfreq = 115.27120 * u.GHz
    # rest frequency (静止频率) of 12 CO 1-0 in GHz
    >>> freq_to_vel = u.doppler_radio(restfreq)
    >>> (116e9 * u.Hz).to(u.km / u.s, equivalencies=freq_to_vel)
    <Quantity -1895.4321928669085 km / s>
    ```

  - **流量** `.spectral_density(wavelength/frequency)`（没太搞懂...）

```
>>> (1.5 * u.Jy).to(u.photon / u.cm**2 / u.s / u.Hz,
equivalencies=u.spectral_density(3500 * u.AA))
<Quantity 2.6429114293019694e-12 ph / (cm2 Hz s)>
```

- **像素与底片的尺寸** `pixel/plate_scale()`

```
# 已知一个像素对应的张角（底片比例尺），求整张底片
>>> sdss_pixelscale = u.pixel_scale(0.4*u.arcsec/u.pixel)
>>> (240*u.pixel).to(u.arcmin, sdss_pixelscale)
<Quantity 1.6 arcmin>
```

```
# 与上相反
>>> tel_platescale = u.plate_scale(7.8*u.m/u.radian)
>>> (0.5*u.arcsec).to(u.micron, tel_platescale)
<Quantity 18.9077335632719 micron>
```

(4) 对数单位

- 星等(mag)、分贝(dB)、$\log_{10}$(Dex) 属于 `LogQuantity` 对象，units 的一个子类?
- 创建：和 Quantity 一样，要么直接创要么和 unit 发生关系

```
>>> u.Magnitude(-10.)
<Magnitude -10.0 mag>
```

- 转换成非对数形式，用 `logq.physical` 属性

1.1.2 常数 `.constants`

- 是 `Quantity` 对象（可以使用 Quantity 对象的所有方法和属性），有一些额外的 meta-data 描述其出处或不确定度
- Class Inheritance：ndarray ⇒ Quantity ⇒ Constant ⇒ EMConstant (电磁单位)
- 导入方法：

```
from astropy.constants import G
# 或
from astropy import constants as const
const.G
```

- 默认使用国际单位制，单位转换见上一节

  注：个别常数因为会引起歧义，需要制定单位制才能使用，碰到的话会有错误信息 TypeError 提示

- 由于astropy版本和常数本身设定的不断更新，const 被打包成不同版本的模块，之前的模块依然保留且可以使用（但 unit 使用的是最新的版本）
  - 针对 astropy：`astropyconst13`、`astropyconst20`
  - 针对 Physical CODATA (Committee on Data for Science and Technology 科学技术委员会)：`codata2010`、`codata2014`

- 针对 IAU：`iau2012`、`iau2015`

1.1.3 数据表格 `.table`

```
from astropy.table import Table, Column, MaskedColumn
```

(1) 创建和操纵表格 `table`

- 可以用已有的列表创建表格：

  ```
  t = Table([a, b, c], names=('a', 'b', 'c'))
  ```

  下面均用 **"t"** 代表 "astropy.table.table.Table" 类的一个对象

- 查看表格的信息：
  - 直接输入表格名字 `t`，返回表格行数、表格内容、每列数据类型
  - `print(t)`，以一种比较土的格式返回表格
  - `t.colnames`，返回列名列表
  - `len(t)`，返回表格行数 (table rows)
- 查看表格内容：和 numpy 中的多维数组方法相似
  - `t[1]` 返回表格第2行 (row index = 1)，及数据类型
  - `t['a']` 返回名为 'a' 的列，及数据类型
  - `t[1]['a']` 或 `t['a'][1]` 均返回第2行第'a'列的元素
  - `t[0:2]` 返回第1、2行（默认不包括 row index = 2 的那一行！）
  - `t['a', 'c']` 返回名为 'a' 和 'c' 的两列
- 修改元素：下标 + 赋值，注意左右对应即可
- 增删改行列：
  - 增加列：`t.add_column(Column(data=[1, 2, 3], name=str), index=int))`

    注意第一个参数是 "Column Object"，要事先 import Column 才行
  - 移除列：`t.remove_column('columnName')`
  - 重命名：`t.rename_column('原名', '替换的名字')`
  - 增加行：`t.add_row([-8, -9, 10])`

  【注】以上函数均可以加一个 `s` 表示增删多行、多列
- every table can have meta-data attached to it via the `meta` attribute, which can be used like a Python dictionary:

  ```
  In []: t.meta['creator'] = 'me'
         t.meta
  Out[]: OrderedDict([('creator', 'me')])
  ```

(2) 掩码表格 ☆

- 掩码的功能是，在处理有数据"缺失"或"无效"的表格时不会报错，仍能按正常表格一样处理；

- 不同于可以被直接无视的 masked values，`NaN` 会给某些 numpy 的操作带来麻烦，比如对一个 nparray 求和时，如果其中有 NaN，则求和的结果也会是 NaN。下面的网址记录了关于如何用 numpy 处理缺失数据的不同策略： http://www.numpy.org/NA-overview.html
- 一般的创建方法是：

```
import numpy as np
import numpy.ma as ma
x = np.array([1, 2, 3, -1, 5])
mx = ma.masked_array(x, mask=[0, 0, 0, 1, 0])
```

- **掩码数组** = ndarray数组 + 布尔数组，布尔数组中值为True的元素表示正常数组中对应下标的值无效，False表示有效（别搞反了！）
- 掩码数组具有三个属性：**data、mask、fill_value**；data表示原始数值数组，mask表示获得掩码用的布尔数组，fill_value表示的填充值替代无效值之后的数组
- 在 `astropy` 里的创建方法是：

```
t = Table([a, b, c], names=('a', 'b', 'c'), masked=True)
# masked参数代表这个表格是否能使用掩码功能
t['a'].mask = [True, True, False]
```

  或者直接使用 `MaskedColumn` 类，但是要先 import：

```
a = MaskedColumn([1, 2], name='a', mask=[False, True], dtype='i4')
b = Column([3, 4], name='b', dtype='i8')
Table([a, b])
```

- 查看表格个元素的掩码情况，直接输入 `t.mask`，返回布尔表格；

  要知道那些元素被掩了，使用 `t[colname].mask.nonzero()`
- 关于**填充**

```
t['a'].fill_value = -99
# 填充某一列的被掩元素，但似乎只能填入一个值?
print(t.filled())
# 查看填充后的数组
print(t['a'].filled(999)
print(t.filled(1000))
# 填充+查看 两步合一
```

(3) 读写表格

- Table 类提供了读写函数

```
t = Table.read('filename', format = '...')
t.write('filename', format = '...')
```

- **内置的读写格式：ASCII、FITS、HDF5、VO Tables**

  ”Suffix(后缀)“这一栏表示创建文件的时候需要指定文件后缀，auto表示python会自动给该文件加上相应的后缀

Format	Write	Suffix	Description
ascii	Yes		ASCII table in any supported format (uses guessing)
ascii.aastex	Yes		`AASTex`: AASTeX deluxetable used for AAS journals
ascii.basic	Yes		`Basic`: Basic table with custom delimiters
ascii.cds	No		`Cds`: CDS format table
ascii.commented_header	Yes		`CommentedHeader`: Column names in a commented line
ascii.csv	Yes	.csv	`Csv`: Basic table with comma-separated values
ascii.daophot	No		`Daophot`: IRAF DAOphot format table
ascii.ecsv	Yes	.ecsv	`Ecsv`: Basic table with Enhanced CSV (supporting metadata)
ascii.fixed_width	Yes		`FixedWidth`: Fixed width
ascii.fixed_width_no_header	Yes		`FixedWidthNoHeader`: Fixed width with no header
ascii.fixed_width_two_line	Yes		`FixedWidthTwoLine`: Fixed width with second header line
ascii.html	Yes	.html	`HTML`: HTML table
ascii.ipac	Yes		`Ipac`: IPAC format table
ascii.latex	Yes	.tex	`Latex`: LaTeX table
ascii.no_header	Yes		`NoHeader`: Basic table with no headers
ascii.rdb	Yes	.rdb	`Rdb`: Tab-separated with a type definition header line
ascii.rst	Yes	.rst	`RST`: reStructuredText simple format table
ascii.sextractor	No		`SExtractor`: SExtractor format table
ascii.tab	Yes		`Tab`: Basic table with tab-separated values
fits	Yes	auto	`fits`: Flexible Image Transport System file
hdf5	Yes	auto	HDF5: Hierarchical Data Format binary file
votable	Yes	auto	`votable`: Table format used by Virtual Observatory (VO) initiative

- 详情请见：http://docs.astropy.org/en/stable/io/unified.html#built-in-readers-writers

1.1.4 天文坐标系统 `.coordinates`

> **坐标系统的定义区分**

- Coordinate **Representation**：a particular way of describing a unique point in a vector space. ⇒ 直角坐标、球坐标、柱坐标
- Reference **System**：a scheme for orienting points in a space and describing how they transforms to other systems. ⇒ ICRS (赤道坐标系with mean equinox)、WGS84 geoid (地球经纬度)
- **Frame**：a specific realization of a reference system. ⇒ CRF, or J2000（有些 system 只有一个有用的 Frame，而有些可能有好几个）
- Coordinate = Representation + System + Frame

(1) `SkyCoord` 类

- `Quantity` 对象 (带单位) + `frame` 结构
  - 默认的 `frame = icrs`，即赤道坐标系 (ra, dec)
  - 一定要对每一个数据都指定单位，无论是 Quantity 自带的，还是通过 unit 引入的
- `frame` 其他可选的坐标系：
  - 地平坐标系 altaz (az, alt) (方位角, 高度角) / `AltAz`
  - 银道坐标系 galactic (l, b) (银经, 银纬) / `Galatic`

```
'altaz', 'barycentrictrueecliptic', 'cirs', 'fk4', 'fk4noeterms', 'fk5', 'galactic',
'galacticlsr', 'galactocentric', 'gcrs', 'geocentrictrueecliptic', 'hcrs',
'heliocentrictrueecliptic', 'icrs', 'itrs', 'lsr', 'precessedgeocentric', 'supergalactic'
```

- 下面几种输入是等价的：

```
c = SkyCoord(10.625, 41.2, frame='icrs', unit='deg')
c = SkyCoord('00h42m30s', '+41d12m00s', frame='icrs')
c = SkyCoord('00h42.5m', '+41d12m')
c = SkyCoord('00 42 30 +41 12 00', unit=(u.hourangle, u.deg))
c = SkyCoord('00:42.5 +41:12', unit=(u.hourangle, u.deg))
In []: c
Out[]: <SkyCoord (ICRS): (ra, dec) in deg
       ( 10.625,  41.2)>
```

- 可以给一个变量赋予 `SkyCoord` 对象的数组，并且对它使用 ndarry 的操作

(2) Coordinate Access

- 可以通过 `SkyCoord` 对象的名字属性获得相应的坐标值，默认的 `ICRS`

```
c = SkyCoord(ra=10.68458*u.degree,dec=41.26917*u.degree)
>>> c.ra   # 赤经-角度(默认)
<Longitude 10.68458 deg>
>>> c.ra.hour  # 赤经-时角
0.7123053333333335
>>> c.ra.hms   # 赤经-时分秒格式
hms_tuple(h=0.0, m=42.0, s=44.299200000000525)
>>> c.dec
<Latitude 41.26917 deg>
>>> c.dec.degree
41.26917
>>> c.dec.radian  # 赤纬-弧度
0.7202828960652683
```

- 还可以用 `to_string()` 方法将坐标转化为字符串 (指定格式) 输出:

```
>>> c.to_string('decimal')    # 小数
'10.6846 41.2692'
>>> c.to_string('dms')        # 度分秒
'10d41m04.488s 41d16m09.012s'
>>> c.to_string('hmsdms')     # 时分秒+度分秒
'00h42m44.2992s +41d16m09.012s'
```

(3) 坐标转换

- 用 `SkyCoord` 对象的属性:

```
>>> c_icrs.galactic
<SkyCoord (Galactic): (l, b) in deg
    ( 121.17424181, -21.57288557)>
```

- 用 `transform_to()` 方法，参数填要转成的 frame 的名字或例子:

```
>>> c_fk5 = c_icrs.transform_to('fk5')
# c_icrs.fk5 does the same thing
>>> c_fk5
<SkyCoord (FK5: equinox=J2000.000): (ra, dec) in deg
    ( 10.68459154,  41.26917146)>
```

```
>>> from astropy.coordinates import FK5
>>> c_fk5.transform_to(FK5(equinox='J1975'))
# precess to a different equinox 利用进动计算不同年份的坐标
<SkyCoord (FK5: equinox=J1975.000): (ra, dec) in deg
    ( 10.34209135,  41.13232112)>
```

(4) Representation (球-直角-柱)

- SkyCoord 的参数 `representation` 可以设置不同的坐标系：默认是球坐标，还可以是“直角/笛卡尔坐标 **cartesian (x, y, z)**“，“柱坐标 **cylindrical (rho, phi, z)**”

```
>>> c = SkyCoord(x=1, y=2, z=3, unit='kpc', representation='cartesian')
>>> c
<SkyCoord (ICRS): (x, y, z) in kpc
    ( 1.,  2.,  3.)>

>>> c.representation = 'cylindrical'
>>> c
<SkyCoord (ICRS): (rho, phi, z) in (kpc, deg, kpc)
    ( 2.23606798,  63.43494882,  3.)>
```

(5) 距离

- 指定 frame 原点、两个角度 + 一个距离 就可以在三维空间中确定一个点的位置，其坐标可以用不同的 representation 输出

```
>>> c = SkyCoord(ra=10.68458*u.degree, dec=41.26917*u.degree, distance=770*u.kpc)
>>> c.cartesian.x
# 同时利用了 transform 和 access
<Quantity 568.7128654235232 kpc>
```

- 计算3D空间中两个点的距离，可以用 `separation()` / `separation_3d()` 方法，参数为代表另一个点的 SkyCoord 对象：

```
>>> c1.separation_3d(c2)
<Distance 1.5228602415117989 pc>
```

(6) Convenience Methods

- ☆ 通过已知的天体名(Sesame)获得其坐标，但必须联网：

```
>>> SkyCoord.from_name('2MASS_J11080002-7717304')
<SkyCoord (ICRS): (ra, dec) in deg
    ( 167.000103, -77.291801)>
```

- cross-matching catalog coordinates (上次sfw说的那个？)
- 用 `EarthLocaiton` 获得地球上某点 (如天文台 / 观测点) 在以地球球心为原点的坐标系里的坐标：
  - 用 `.of_site()` 查询某些特定的位置，至于那些地点可查，请用 `astropy.coordinates.EarthLocation.get_site_names()`
  - 用 `.of_address()` 自动求助Google地图
  - 因为这些方法是为方便使用设计的，故所得坐标的精确度不能保证！要求高请另寻他法

```
>>> from astropy.coordinates import EarthLocation
>>> EarthLocation.of_site('Apache Point Observatory')
<EarthLocation (-1463969.30185172, -5166673.34223433,  3434985.71204565) m>

>>> EarthLocation.of_address('1002 Holy Grail Court, St. Louis, MO')
<EarthLocation (-26727.24739672, -4997012.16094607,  3950268.27273576) m>
```

(7) 自行 (proper motion) & 视向速度 (radial velocity)

略复杂，详情见 Working with velocities in Astropy coordinates

1.1.5 World Coordinate System `.wcs`

(1) Introduction 简介

用来完成 FITS files 的 wcs transformation，即图像像素位置和真实天球坐标之间的转换：pixel ⇔ sky

3 separate classes of WCS transformations:

- **Core WCS**：as defined in the FITS WCS standard, based on Mark Calabretta's wcslib. (Also includes `TPV` and `TPD` distortion, but not `SIP`).
- **Simple Imaging Polynomial** (SIP) convention. (See note about SIP in headers.)
- table lookup distortions as defined in the FITS WCS distortion paper.

(2) Workflow 工作流程

1. ```
   from astropy import wcs
   ```
2. Call the WCS constructor with an `astropy.io.fits Header` and/or `HDUList` object.
3. （选做）如果 FITS file 的格式不标准，需要调用 `fit` 方法对其进行修正
4. 选择其中一种转换方法，注意以下都是 `wcs` **objects 的方法名**!
   - 【pixels ⇒ world coordinates】
     - `all_pix2world`：顺序执行三种转换方式 core WCS, SIP and table lookup distortions（在不知道该用什么的时候用这个）
     - `wcs_pix2world`：只用 core WCS transformation
   - 【world \Rightarrow pixel coordinates】

     `all_world2pix` 、 `wcs_world2pix` 同上
   - 只用 SIP
     - `sip_pix2foc`: Convert from pixel to **focal plane** coordinates using SIP
     - `sip_foc2pix`
   - 只用 distortion paper transformations
     - `p4_pix2foc`: Convert from pixel to focal plane coordinates using the table lookup distortion method described in the FITS WCS distortion paper（已下载到doc中）
     - `det2im`: Convert from detector coordinates to image coordinates. Commonly used for narrow column correction.
   ```

(3) Load WCS information from a FITS file

```
hdulist = fits.open(filename)

# Parse the WCS keywords in the primary HDU / 提取wcs信息
w = wcs.WCS(hdulist[0].header)

# 输入被转换的坐标
pixcrd = numpy.array([......], numpy.float_)

# 用wcs对象的相应方法进行转换
world = w.wcs_pix2world(pixcrd, 1)
pixcrd2 = w.wcs_world2pix(world, 1)
# （误？）第二个参数是“ra_dec_order”，输入布尔值，如果是True则返回的坐标顺序一定是(ra,dec)，无论在header里的CTYPE顺序如何
```

- 可以查提取出的 wcs 信息，例如：

```
In []: w
Out[]: WCS Keywords

Number of WCS axes: 2
CTYPE : 'RA---ARC'  'DEC--ARC'
CRVAL : 165.74767294849346  -77.455490189207822
CRPIX : 861.0  1074.0
NAXIS : 1722  2146
```

1.2 文件的读入读出(I/O)与传输

1.2.1 处理 FITS 文件 (`astropy.io.fits`)

> 【注】因为这个包比较重要，不好粗略整理，就把我的个人笔记贴过来了，所以内容有点多……

astropy.io.fits包提供对FITS文件的访问。 FITS（灵活图像传输系统）是天文学界广泛使用的便携式文件标准，用于存储图像和表格。

导入 fits 模块：`from astropy.io import fits`

1.2.1.1 fits文件的结构 & 相关操作

- 第一层：`HDUlist`，是一个HDU对象的列表
- 第二层：`HDU` = Header Data Unit，通常包括一个 header 和 data array 或 table，可以通过相应属性来获取：`.header` & `.data`
  - `hdulist[0]` 是 primary HDU，`hdulist[1]` 是第一个 extension HDU，以此类推（注意python采用的是"zero-based indexing"）
- 第三层：`header`，HDU的第一个部分
  - 查看信息，直接输入header的名字，或 `print(repr(header))`
  - 可以用指标来查看部分信息，如输入 `header[:2]` 只有前两行信息显示
  - 查看所有card的关键词，用 `header.keys()`，返回字符串列表

- 第四层：`card`，HDU.header 包括一个 card 的列表，每个 card 长度为 80-bytes，每一个 **card = keyword + value + comment**，其中头尾两个只能是字符串类型
  - 用关键词和指标都可以索引 card：

```
>>> hdulist[0].header['targname']
'NGC121'
>>> hdulist[0].header[27]
96
```

  - 修改 / 更新 card 最好用关键词！有两种方法：

```
# 直接赋值
prihdr = hdulist[0].header
prihdr['targname'] = 'NGC121-a'
prihdr[27] = 99
# 使用set函数
prihdr.set('observer', 'Edwin Hubble')
```

  - 可以用元组的方式同时更新 value 和 comment：

```
>>> prihdr['targname'] = ('NGC121-a', 'the observation target')
>>> prihdr['targname']
'NGC121-a'
>>> prihdr.comments['targname']
'the observation target'
```

  - `comment` 和 `history` 这两个词可以作为重复的关键词使用，更准确的说这是两个特别的card类型
    - 添加时，会自动添加到有相同关键词的card的后面，即遵循先来后到（注意不要和普通card里面的comment搞混了！）
    - 更新已有的 COMMENT 和 HISTORY card 时，用指标来索引：

```
prihdr['history'][0] = 'I updated this file on 2/27/09'
prihdr['comment'][1] = 'I like using JWST observations'
```

1.2.1.2 fits文件的整体操作

- 打开：`hdulist = fits.open('filename.fits')`

  （如果文件太大，可以设置参数 `memmap = True`，但是这会给文件关闭带来风险）
- 关闭：`hdulist.close()`
- 获取信息：`hdulist.info()`

```
Filename: 2MASS_J11070925-7718471_cont.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU    1682   (350, 350, 1, 1)   float32
```

- 更新 (修改 / 添加) header 中的信息，使用 `fits.setval()` 函数：

```
fits.setval(fileObject, 'keyword', value = str/float, ext = int)
# ext=0 修改primaryHDU的; ext=1 修改第一个extensionHDU的值
```

1.2.1.3 处理图像数据

1.2.1.3.1 Image Data as an Array

- 如果一个 HDU 对象包含图像数据，那么它的 data 属性会返回一个 **ndarray**（这意味着一切都按照玩 ndarray 的方法玩就行了）

```
scidata = hdulist[1].data
# 需要确定 hdulist[1] 储存着所要的 image data
# 但一般情况下，图像信息都储存在 primary HDU (ext=0) 里！！！
```

  也可以直接用 `fits.getdata()` 获取：

```
image_data = fits.getdata(fileObject, ext=0)
```

- 这个 ndarray 对象有不少属性可以获得信息：

```
>>> scidata.shape
(800, 800)
>>> scidata.dtype.name
'float32'
```

  - data type 由 HDU header 中的 `BITPIX` 给出：

```
BITPIX    Numpy Data Type
8         numpy.uint8 (note it is UNsigned integer)
16        numpy.int16
32        numpy.int32
-32       numpy.float32
-64       numpy.float64
```

  - shape 由 `NAXIS` 给出：

    如果 NAXIS1 = 300, NAXIS2 = 400, 则 shape = (300, 400)

- 用 ndarray 的指标操作读取 / 重制图像的一部分：

```
scidata[30:40, 10:20]  # get values of the subsection
# from x=11 to 20, y=31 to 40 (inclusive)
```

- 还有某些特殊的玩法，如利用 “mask array” 的概念：

```
scidata[scidata < 0] = 0
>>> scidata[scidata > 0] = numpy.log(scidata[scidata > 0])
```

- 把 ndarray 中的数据以图像的方式显示出来，可用 `astropy` 里写好的画图格式 `astropy_mpl_style`：

```
import matplotlib.pyplot as plt
from astropy.visualization import astropy_mpl_style
plt.style.use(astropy_mpl_style)

plt.figure()
plt.imshow(image_data, cmap='gray')
plt.colorbar()
```

1.2.1.3.2 Scaled Data

（因为暂时还不需要处理这种fits，故略过）

- 有些图像数据并不是真实的物理数据，而是按照某种比例缩放过的，要用 `BSCALE` 和 `BZERO` 的值还原：

```
physical value = BSCALE * (storage value) + BZERO
```

如果 BSCALE = 1, BZERO = 0，说明数据就是原来的

1.2.1.3.3 Data Sections ☆

- 对于一个很大的 fits file，当只需要处理其中一小部分时，可用 `section` 属性来节省内存

```
f1 = fits.open('file1.fits')
for i in range(50):
    j = i * 100
    k = j + 100
    x1 = f1[1].section[j:k,:]
```

- 对于 `section` 做的修改不会被保存到原始的文件中

1.2.1.4 处理表格数据

1.2.1.5 便捷函数

- 这些函数将一些功能封装起来，用起来非常方便，但代价是效率低下，不建议在 application code 中应用。如果代码要经常重复使用，最好直接写成低级形式...
- `fits.info('filename')` 打印该文件的所有信息：

```
No. Name    Type    Cards    Dimensions    Format
```

- [ ] 看到 Creating a New FITS File 就停！

1.2.2 ASCII 表格 (`astropy.io.ascii`)

astropy.io.ascii提供了读取和写入各种ASCII数据表格的方法（通过内置的Extension Reader类）。它的重点在于灵活性和易用性，但读者也可以选择使用灵活性较低但性能更好的C / Cython引擎进行读写。

以下显示了一些可用的ASCII格式：（Supported formats 包含了完整列表）

- Basic：带有可定制分隔符和标题结构的表格（基本形式）
- CD：CDS格式表（也是Vizier和ApJ可读的格式）
- Daophot：来自IRAF DAOphot包的表格
- Ecsv：ECSV格式，用于数据表的无损往返
- FixedWidth：具有固定宽度列的表（另请参阅固定宽度的图库）
- Ipac：IPAC格式表
- HTML：包含在标记中的HTML格式表
- Latex：在`tabular`环境中带有数据值的LaTeX表
- Rdb：制表符分隔的值，在列定义行之后加上一行
- SExtractor：SExtractor格式表

函数名称	函数功能
`convert_numpy`(numpy_type)	返回一个元组，其中包含将列表转换为numpy数组的函数，以及由转换函数生成的类型
`get_read_trace`()	返回最后一次尝试读取表格的回溯，并读取能够识别的部分
`get_reader`([Reader, Inputter, Outputter])	初始化表格阅读器，可设置常见的自定义
`get_writer`([Writer, fast_writer])	初始化的表格写入器，可设置常见的自定义
`read`(table[, guess])	读取并输入表格内容，并返回
`set_guess`(guess)	设置`read()`函数的`guess`参数的默认值
`write`(table[, output, format, Writer, ...])	将以参数形式传入的表格写到指定文件名的文件中

1.2.2.1 支持的格式

Format	Write	Fast	Description
`aastex`	Yes		`AASTex`: AASTeX deluxetable used for AAS journals
`basic`	Yes	Yes	`Basic`: Basic table with custom delimiters
`cds`			`Cds`: CDS format table
`commented_header`	Yes	Yes	`CommentedHeader`: Column names in a commented line
`csv`	Yes	Yes	`Csv`: Basic table with comma-separated values
`daophot`			`Daophot`: IRAF DAOphot format table
`ecsv`	Yes		`Ecsv`: Enhanced CSV format
`fixed_width`	Yes		`FixedWidth`: Fixed width
`fixed_width_no_header`	Yes		`FixedWidthNoHeader`: Fixed width with no header
`fixed_width_two_line`	Yes		`FixedWidthTwoLine`: Fixed width with second header line
`html`	Yes		`HTML`: HTML format table
`ipac`	Yes		`Ipac`: IPAC format table
`latex`	Yes		`Latex`: LaTeX table
`no_header`	Yes	Yes	`NoHeader`: Basic table with no headers
`rdb`	Yes	Yes	`Rdb`: Tab-separated with a type definition header line
`rst`	Yes		`RST`: reStructuredText simple format table
`sextractor`			`SExtractor`: SExtractor format table
`tab`	Yes	Yes	`Tab`: Basic table with tab-separated values

1.2.2.2 读取表格

- 大部分遇到的ASCII表格可以直接用 `read()` 读取，返回一个 `table` 对象

```
>>> from astropy.io import ascii
>>> data = ascii.read("sources.dat")
>>> print(data)
obsid redshift  X    Y      object
----- -------- ---- ---- -----------
 3102     0.32 4167 4085 Q1250+568-A
  877     0.22 4378 3892   Source 82
```

- 一般来说读取时 `read()` 会尝试所有支持的表格形式，如果还是不行就需要在 `line` 参数里手动输入所需的读取格式：

```
>>> lines = ['objID     & osrcid    & xsrcid        ',
...          '--------- & --------- & ---------------',
...          '277955213 & S000.7044P00.7513 & XS04861B6_005',
...          '889974380 & S002.9051P14.7003 & XS03957B7_004']
>>> data = ascii.read(lines, data_start=2, delimiter='&')
```

- 如果表格形式在"支持列表"里，那么直接在 `format` 参数里指明即可

(1) 确定 header 和 data 的位置

- `header_start`，`data_start` 和 `data_end` 三个参数可以让我们避开不相关的非表格数据（下面的例子中最左边一列不是表格的一部分，只是为了显示 `io.ascii` 是怎么计算行数的）

```
Index    Table content
------ ------------------------------------------------------------
   -  | # This is the start of my data file
   -  |
   0  | Automatically generated by my_script.py at 2012-01-01T12:13:14
   1  | Run parameters: None
   2  | Column header line:
   -  |
   3  | x y z
   -  |
   4  | Data values section:
   -  |
   5  | 1 2 3
   6  | 4 5 6
   -  |
   7  | Run completed at 2012:01-01T12:14:01
```

(2) 损坏/缺失的数据

- 如果表格有缺失的数据，`read()` 在读取时会返回一个 masked table，即在缺失处设置 mask value = True
- 如果想在缺失处填入特定的值，就设置表格某一列的 `fill_value` 属性：

```
>>> dat['precip'].fill_value = -999
>>> dat['type'].fill_value = 'N/A'
>>> print(dat.filled())
day  precip type
---- ------ ----
 Mon    1.5 rain
Tues -999.0  N/A
 Wed    1.1 snow
```

- 如果想把多个不同类型的值统一设为“缺失值”并替换掉，用 `fill_values` 参数，具体设置的格式如下：

```
fill_values = [<missing_spec1>, <missing_spec2>, ...]
<missing_spec> = (<match_string>, '0', <optional col name 1>, <optional col name 2>,
...)
```

  - 注意第二个参数必须设置为字符串 ‘0’（老版本遗留下来的）
  - 如果后面的几个参数(column name)不给的话，将会默认应用于所有列
- `read()` 默认的 `fill_values = ('', '0')`，如果不像启用该功能需手动设置 `fill_values = None`

(3) 数据类型转换

- 利用 `convert_numpy()` 可以将一整列的数据类型转换，但原来的数据类型必须是有效的 numpy type
- 在 `read()` 的 `converters` 参数里设置要转换的列和类型：

```
>>> converters = {'col1':[ascii.convert_numpy(np.uint)],
                  'col2':[ascii.convert_numpy(np.float32)]}
>>> ascii.read('file.dat', converters=converters)
```

1.2.2.3 写入表格 (略)

`write(table, file, ...)`

1.2.3 处理 VOTable XML 文件 (`astropy.io.votable`)

astropy.io.votable将Votable XML文件与Numpy记录数组来回转换。

1.2.3.1 astropy.io.votable

这个程序包读取和写入虚拟天文台（VO）倡议使用的数据格式，特别是VOTable XML格式。

函数名称	函数功能
parse(source[, columns, invalid, pedantic, ...])	解析VOTABLE xml文件 (或文件类对象)，并返回一个 VOTableFile 对象
parse_single_table(source, **kwargs)	解析一个VOTABLE xml文件 (或类似文件的对象)，只读取和返回第一个Table实例
validate(source[, output, xmllint, filename])	打印给定文件的验证报告
from_table(table[, table_id])	给定一个Table对象，返回一个包含该单个表的 VOTableFile 文件结构
is_votable(source)	读取文件头(header)以确定它是否为VOTable文件
writeto(table, file[, tabledata_format])	将 VOTableFile 写入VOTABLE xml文件

1.2.3.2 astropy.io.votable.tree

1.2.3.3 astropy.io.votable.converters 模块

该模块处理各种VOTABLE的数据类型与 TABLEDATA 和 BINARY 类型的转换。

函数名称	函数功能
get_converter(field[, config, pos])	为给定字段获取适当的转换器实例
table_column_to_votable_datatype(column)	给定一个 astropy.table.Column 实例，返回与该列类型相对应的、创建VOTable FIELD元素时所需的属性

1.2.3.4 astropy.io.votable.ucd 模块

该模块用于验证UCD(unified content descriptor)/统一内容描述符字符串的正确性。s

函数名称	函数功能
parse_ucd(ucd[, ...])	将UCD(含义见下)解析为其组成部分
check_ucd(ucd[, ...])	如果*ucd*不是有效的统一内容描述符，则返回False

1.2.3.5 astropy.io.votable.util 模块

提供各种各样的应用和类似指导手册里的内容。

函数名称	函数功能
convert_to_writable_filelike(fd[, compressed])	返回一个适合streaming output/流式输出的可写文件类对象
coerce_range_list_param(p[, frames, numeric])	强制或验证对象*p*为有效的范围列表格式(range-list-format)参数

1.2.3.6 astropy.io.votable.validator 模块

验证大量可通过Web访问的VOTable文件，并生成报告作为HTML文件的目录树。

函数名称	函数功能
make_validation_report([urls, destdir, ...])	验证大量可通过网络访问的 VOTable 文件

1.2.3.7 astropy.io.votable.xmlutil 模块

各种与XML有关的应用。

函数名称	函数功能
check_id(ID[, name, config, pos])	如果ID不是有效的XML ID，则引发 VOTableSpecError
fix_id(ID[, config, pos])	任意给定一个字符串，创建一个可以用作xml id的字符串
check_token(token, attr_name[, config, pos])	如果标记不是有效的XML标记，则引发 ValueError
check_mime_content_type(content_type[, ...])	如果content_type不是有效的MIME内容类型，则引发 VOTableSpecError
check_anyuri(uri[, config, pos])	如果uri不是有效的URI，则引发 VOTableSpecError
validate_schema(filename[, version])	根据适当的VOTable模式验证给定的文件

1.2.3 各种文件类型: HDF5, YAML, ASDF, pickle (astropy.io.misc)

astropy.io.misc模块包含不能在其它地方使用的各种输入/输出程序，并且经常被其他Astropy子包使用。 例如， astropy.io.misc.hdf5 包含用于从/向HDF5文件读取或写入表格对象的功能，但这些功能不应直接由用户导入。 相反，用户可以通过 Table 类本身访问此功能 (请参阅 Unified file read/write interface)。 astropy.io.misc部分列出了打算由用户直接使用的例程

函数名称	函数功能
`fnpickle`(object, fileorname[, usecPickle, ...])	将一个对象嵌入到指定的文件中
`fnunpickle`(fileorname[, number, usecPickle])	将嵌入到特定文件中的对象取出并并返回其内容

1.2.3.1 astropy.io.misc.hdf5 模块

This package contains functions for reading and writing HDF5 tables that are not meant to be used directly, but instead are available as readers/writers in `astropy.table`. See Unified file read/write interface for more details.

函数名称	函数功能
`read_table_hdf5`(input[, path])	从HDF5文件中读出`Table`对象
`write_table_hdf5`(table, output[, path, ...])	向HDF5文件中写入`Table`对象

1.2.3.2 astropy.io.misc.yaml 模块

该模块包含通过 YAML 协议使核心Astropy对象的函数序列化。

注：该模块需要安装 PyYaml 3.12 或以上的版本。

函数名称	函数功能
`load`(stream)	用AstropyLoader解析数据流中的第一个YAML文档，并生成相应的Python对象
`load_all`(stream)	用AstropyLoader解析数据流中的所有YAML文档，并生成相应的Python对象
`dump`(data[, stream])	使用AstropyDumper类将Python对象序列化为YAML流

1.2.3.3 astropy.io.misc.asdf 包

该子包包含用于序列化Astropy类型的代码，以便可以使用高级科学数据格式 (Advanced Scientific Data Format, ASDF) 来表示和存储它们。 这个子包定义了类，称为*tags*，它们实现了序列化和反序列化的逻辑。

1.2.4 SAMP (Simple Application Messaging Protocol (`astropy.samp`))

astropy.samp是SAMP信息系统的Python实现。

SAMP/简单应用消息传递协议是一种进程间通信系统，它允许不同客户端程序 (通常是运行在同一台计算机上的) 通过交换可能引用外部数据文件的短信息来相互通信。该协议是在IVOA (国际虚拟天文台联盟) 内开发的，许多电脑上天文软件都能兼容，包括TOPCAT、SAO、Ds9和Aladin。

因此，通过使用astropy.samp中的类，Python代码可以与其他正在运行的桌面客户端进行交互，例如在Ds9中显示一个名为FITS的文件，使Aladin在给定的天空位置重新居中，或者当用户点到TOPCAT中的绘制点时显示那一行的消息。

1.3 计算与应用

1.3.1 宇宙学计算（`astropy.cosmology`）

astropy.cosmology 子包含用于表示宇宙学的类，以及用于计算依赖于宇宙学模型的常见物理量的函数。 如与测量的红移相对应的距离、年龄和回溯时间，或者与测量的角距离相对应的横向间隔。

大部分功能都由`FLRW`对象启用。 这代表了一种均匀且各向同性的宇宙（以**Friedmann-Lemaitre-Robertson-Walker**度规为特征，以解出爱因斯坦场方程的人们命名）。 但你无法直接使用它，你必须使用其中一个子类 (如`FlatLambdaCDM`) 来指定暗能量模型。

函数名	函数功能
`z_at_value`(func, fval[, zmin, zmax, ztol, ...])	由`func(z) = fval`解出红移值 z.

1.3.2 卷积与滤波（`astropy.convolution`）

astropy.convolution 提供了卷积函数以及在scipy scipy.ndimage基础上改进的内核 (kernels)，包括：

- 正确处理NaN值 (在卷积过程中忽略它们并用内插值替换NaN像素)
- 一维，二维和三维卷积的单一函数
- 改进了处理边缘的选项
- 直接和快速傅立叶变换 (FFT)
- 天文学中常用的内置内核

函数名称	函数功能
`convolve`(array, kernel[, boundary, ...])	将一个array与指定内核进行卷积
`convolve_fft`(array, kernel[, boundary, ...])	将ndarray与指定内核进行卷积
`convolve_models`(model, kernel[, mode])	用`convolve_fft`卷积两个模型
`discretize_model`(model, x_range[, y_range, ...])	网格法估计解析形式的模型函数
`interpolate_replace_nans`(array, kernel[, ...])	给定一个含有NaN的数据集，将NaN置换为内核中对应的相邻数据点的插值
`kernel_arithmetics`(kernel, value, operation)	相加、相减或相乘两个内核

1.3.3 数据可视化（`astropy.visualization`）

astropy.visualization 提供了可视化数据时有用的函数。 这包括用Matplotlib（之前是独立的wsaxes包）绘制天文图像的坐标，与图像归一化相关的功能（包括缩放和拉伸），智能直方图绘制，从单张图像创建RGB彩色图像，以及自定义Matplotlib的绘图样式。

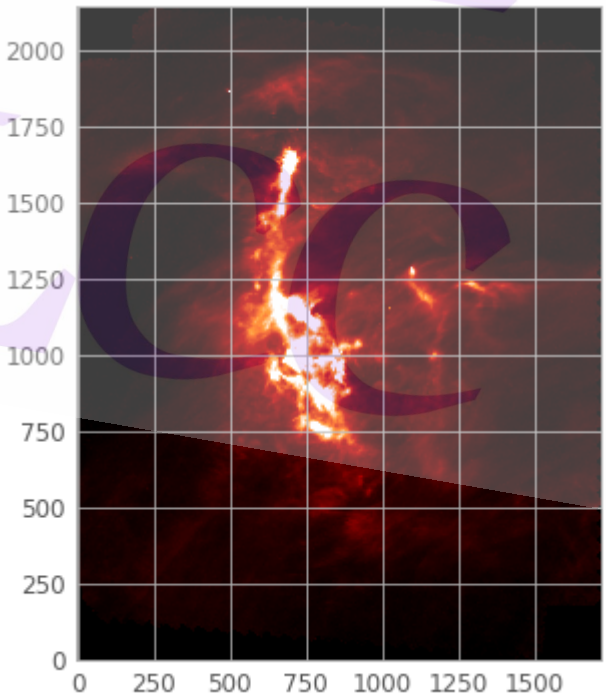
函数名称	函数功能
`hist`(x[, bins, ax])	升级的柱状图函数
`make_lupton_rgb`(image_r, image_g, image_b[, ...])	使用asinh拉伸从最多3张图像返回红/绿/蓝彩色图像
`quantity_support`([format])	启用对绘制 `astropy.units.Quantity` 类型数据的支持
`simple_norm`(data[, stretch, power, asinh_a, ...])	返回一个可以用Matplotlib绘图的Normalization类

1.3.3.0 Astropy Matplotlib style

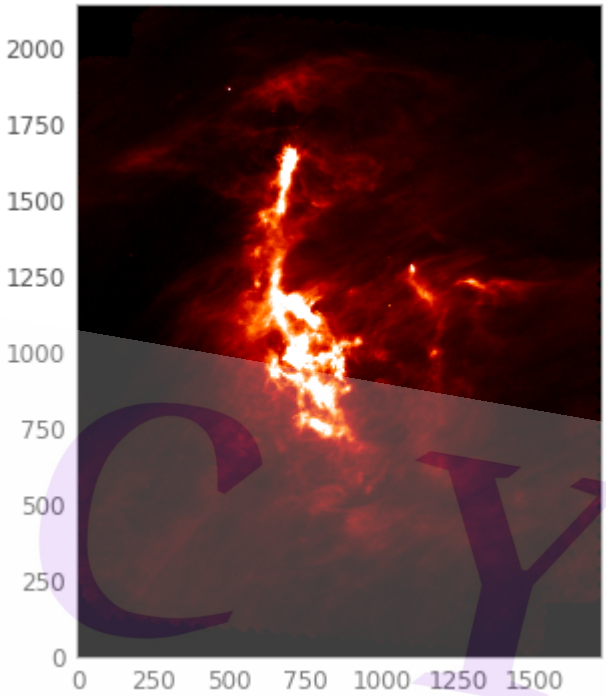
`.visualization` 包含两个可以用来设置绘图风格的字典：

- `astropy_mpl_style`：在 mpl 默认风格上进行了某些改良的版本

```
import matplotlib.pyplot as plt
from astropy.visualization import astropy_mpl_style
plt.style.use(astropy_mpl_style)
```



- `astropy_mpl_docs_style`：Astropy docs 用的风格



1.3.3.1 Making plots with world coordinates (WCSAxes)

(1) Initializing axes with world coordinates

- 创建一个 `WCSAxes` 对象的基本方法：利用 `wcs` 类，把它作为 `projection` 参数传递到 `plt.subplot()` 中去。当然也可以传递到其他函数，如 `axes()`、`add_subplot()`、`add_axes()`
  - `get_pkg_data_filename()` 下载 fits 文件并返回它的文件名

```
from astropy.wcs import WCS
from astropy.io import fits
from astropy.utils.data import get_pkg_data_filename

filename = get_pkg_data_filename('galactic_center/gc_msx_e.fits')
hdu = fits.open(filename)[0]
wcs = WCS(hdu.header)
# key: 从header里面提取wcs信息
ax = plt.subplot(projection=wcs)
# 注意 keeping a reference to the axes object!
```

  - 如果想认为设置坐标轴上下限，可用 `ax.set_xlim()`

- 更直接的创建方法：直接用 `WCSAxes()` 创建对象，然后把它加到 figure 中（这一步不能少！）

```
from astropy.visualization.wcsaxes import WCSAxes
fig = plt.figure()
ax = WCSAxes(fig, [0.1, 0.1, 0.8, 0.8], wcs=wcs)
fig.add_axes(ax)   #这行不能漏
```

  - `WCSAxes()` 的主要参数
    - fig：要添加过去的 figure 对象
    - rect：axes 在 figure 对象里的相对位置 `[left, bottom, width, height]`

- wcs：the WCS for the data

(2) 绘图 & 轮廓线 Plotting images & contours

- `ax.imshow()` / `plt.imshow()`
- `ax.contour()` / `plt.contour()`

(3) 刻度 tick、刻度标签 tick label、网格 grid

- 坐标对象
  - 复习：可以通过 `plt.gca()` (get current axes) 来获得目前活跃的 axes 对象
- 坐标可以通过 `ax` 对象的 `.coords` 属性获得：

```
lon = ax.coords[0]
lat = ax.coords[1]
```

- 如果坐标是某些特别的天球坐标系，可以通过指定名字来获得：

```
lon = ax.coords['glon']   # ['ra']
lat = ax.coords['glat']   # ['dec']
```

- 坐标轴**标签** (axis labels)
  - 使用坐标对象的 `.set_axislabel('...')` 方法
  - label 离 axes 的远近可以通过上述方法中的 `minpad` 参数来设置，默认为1，可以是负数 (这样标签和刻度会挨得比较近甚至重叠...)
- 刻度标签**格式** (tick label **format**) ☆
  - 可以通过 `.set_major_formatter()` 方法和"字符串"形式设置，如：

```
lon.set_major_formatter('dd:mm:ss.s')
lat.set_major_formatter('dd:mm')
```

  - 可选的字符串格式有： 'dd:mm:ss'、'hh:mm:ss'、'd.dd'、'm.m'、'x.xxx'、'%.2f'、'%d'。注意和 d, h, m, s 有关的可以用于单位是角度的坐标，但 x 形式应该用于非角度坐标
- **刻度** / 标签的间隔和性质 (tick/label spacing and properties)
  - `.set_ticks()` 方法 + Quantity 手动设置刻度位置、间隔、数量 ☆：

```
lon.set_ticks([242.2, 242.3, 242.4] * u.degree)
lon.set_ticks(spacing=5. * u.arcmin)
lon.set_ticks(number=4)
```

  - 用 `.set_ticks()` 还可以设置刻度的样式，比如颜色、大小、是否被覆盖等

```
lon.set_ticks(spacing=10 * u.arcmin, color='white', exclude_overlapping=True)  #不被覆盖
```

- 设置副刻度 (minor tick)

- 默认是“不显示 + 每两个主刻度间有5个”，这些可以手动修改：

```
lon.display_minor_ticks(True)
lat.set_minor_frequency(10)
```

- 刻度、刻度标签、坐标轴在图上的位置 (position)
  - `.set_ticks_position()`、`.set_ticklabel_position()`、`set_axislabel_position()`，参数是代表位置的字符串：`l`, `b`, `r`, `t`, `all` 分别代表 left, bottom, right, top, lbrt都有，前四种可以互相组合?
- 不显示刻度和标签 (hide)

  有时候需要几幅图组成网格状的组图，不想要刻度和坐标轴的标签，用

  `.set_ticks_visible(False)`、`.set_ticklabel_visible(False)`、`.set_axislabel('')`
- 坐标网格 (coordinate grid)
  - 设置网格性质用坐标对象的 `.grid()` 方法即可：

```
lon.grid(color='yellow', alpha=0.5, linestyle='solid')
```

  - 或者进行统一设置：

```
ax.coords.grid(color='white', alpha=0.5, linestyle='solid')
```

(4) 添加 markers & artists

首先要注意添加标记时所处的坐标系统：

- 像素坐标 / pixel coordinate

  `WCSAxes` 和正常的 axes 对象没什么不同，所以绘图时默认用像素坐标
  - 首先，防止添加标记时原图会自动缩放：

```
# The following line makes it so that the zoom level no longer changes, otherwise
Matplotlib has a tendency to zoom out when adding overlays
ax.set_autoscale_on(False)
```

```
from matplotlib.patches import Rectangle
r = Rectangle((30., 50.), 60., 50., edgecolor='yellow', facecolor='none')
# 参数分别是：中心坐标，宽度，高度
ax.add_patch(r)
```

```
# 参数分别为：x，y坐标，面积，颜色(RGB,alpha)
ax.scatter([40, 100, 130], [30, 130, 60], s=100, edgecolor='white', facecolor=(1, 1, 1,
0.5))
```

- WCS 坐标 / world coordinate

- 在使用上面的绘图方法时，可通过添加参数 `transform =` 来把输入的坐标转换成像素坐标
- WCSAxes 对象还有一种方法是 `.get_transform()` 来从 wcs 系统中获取正确的转换对象，最简单的参数是 'world' 和 'pixel'（但是用'pixel'就和transform参数什么都没传入一样）

```
ax.scatter([34], [3.2], transform=ax.get_transform('world'))
```

- 天球坐标 / celestial coordinate

  任何 `astropy.coordinates` 里面有效的 frame 都可以作为参数传入 `.get_transform()`，几个典型代表：

  - `'fk4'`: B1950 FK4 equatorial coordinates
  - `'fk5'`: J2000 FK5 equatorial coordinates
  - `'icrs'`: ICRS equatorial coordinates
  - `'galactic'`: Galactic coordinates

不仅是 scatter() 能进行坐标转换，其他标记也行：

- patches / shapes / lines
  - 先设置各种“补丁”：

```
r = Rectangle((266.0, -28.9), 0.3, 0.15, edgecolor='green', facecolor='none',
transform=ax.get_transform('fk5'))
# 这个矩形会画在坐标为 FK5 J2000 (266deg, -28.9deg) 的位置
ax.add_patch(r)
```

  需要注意的是这个矩形的宽不一定是严格的 0.3°(because longitude is not the same as the angle on the sky ???) ，就像输入圆形 / 正方形但画出来的不一定是圆的 / 方的一样

```
from matplotlib.patches import Circle
c = Circle((266.4, -29.1), 0.15, edgecolor='yellow',
facecolor='none',transform=ax.get_transform('fk5'))
ax.add_patch(c)
```

  - 所以如果纯粹是想圈出某个源，最好用 `scatter()`，它能保证画出来的是一个圆圈
  - 最后一定要用 `.add_patch()` 把你设置好的标记添加到图里去！
- Spherical patches

  如果想圈出以某一经纬度范围内的区域，需要用 `SphericalCircle` 而非 `Circle`，因为后者会有畸变

```
  from astropy.visualization.wcsaxes import SphericalCircle
  # 注意输入的坐标和半径参数都要带单位！
  r = SphericalCircle((266.4 * u.deg, -29.1 * u.deg), 0.15 * u.degree, edgecolor='yellow',
facecolor='none', transform=ax.get_transform('fk5'))
  ax.add_patch(r)
```

- 轮廓线 / contours
  - 第一个参数指定被画的图像，不用 add_patch
  - transform 参数要传入图像的 WCS

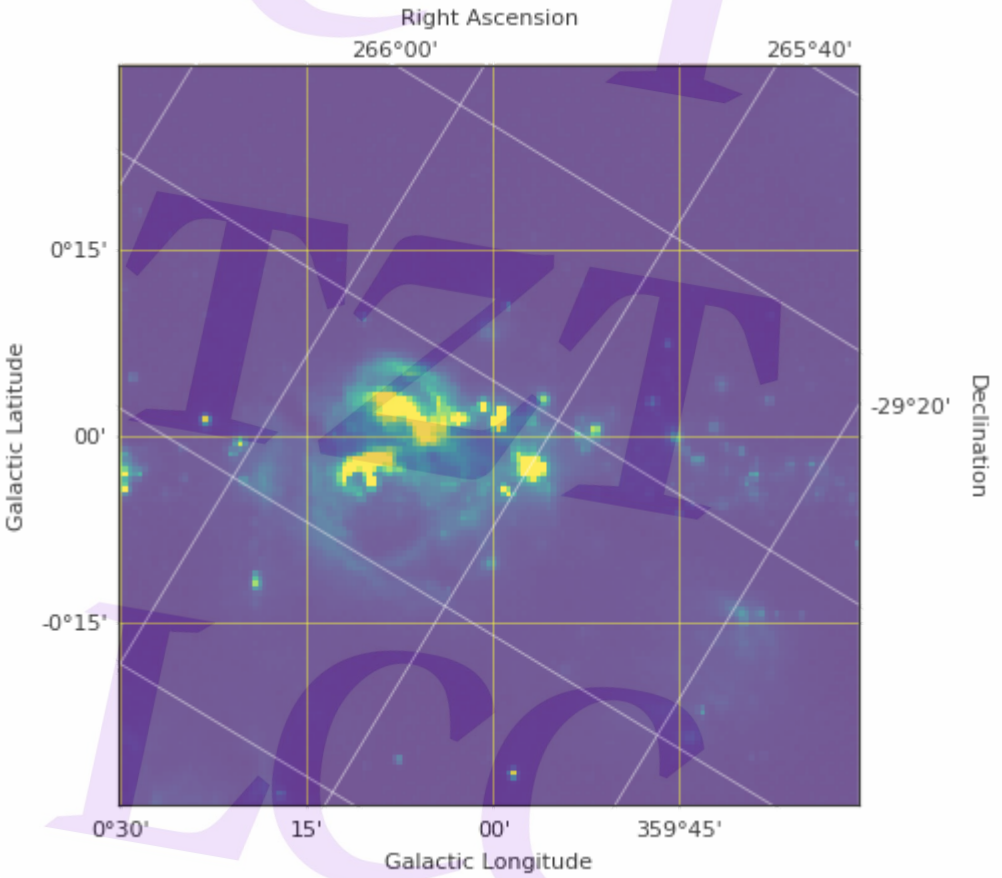
```
ax.contour(hdu.data, transform=ax.get_transform(WCS(hdu.header)), levels=[1,2,3,4,5,6],
colors='white')
```

(5) 同时画不同的坐标系 Overlaying coordinate system

- 用 `.get_coords_overlay('frame')` 会返回一个 **`CoordinatesMap`** 对象，它和 `ax.coord` 是一种类型，有设置刻度、标签、坐标轴的各种方法：

```
ax.coords['glon'].set_axislabel('Galactic Longitude')
ax.coords['glat'].set_ticks(color='white')

overlay['ra'].set_ticks(color='white')
overlay['dec'].set_axislabel('Declination')
overlay.grid(color='white', linestyle='solid', alpha=0.5)
```



(6) 选取数据维度 Slicing multidimentional data ☆

- 在创建 WCSAxes 对象的时候设置参数 `slices`

```
ax = plt.subplot(projection=wcs, slices=(50, 'y', 'x'))
# 这里我们把第二个维度设置在'y'轴上，第三个维度设置在'x'轴
# '50'表示选择 not shown dimension 的第50个切片(slice)
```

(7) 设置坐标轴 Controling axes

- 改变坐标轴单位： **`.set_format_unit()`**

```
ax.coords[2].set_format_unit(u.km / u.s)
ax.coords[2].set_major_formatter('x.x')
# 注意'x.x'只能用于非角度单位情况
```

- 改变坐标轴方向：`ax.invert_xaxis()`

1.3.4 天文数据工具（`astropy.stats`）

astropy.stats 包含天文学中使用的统计函数或算法。 虽然 scipy.stats 和statsmodel 包含广泛的统计工具，但它们是通用的程序包，并且缺少一些特别有用或特定于天文学的工具。 这个包旨在提供这样的功能，但不能取代能满足天文学家需求的 scipy.stats 部分。

函数名称	函数功能
akaike_info_criterion(log_likelihood, ...)	计算 Akaike Information Criterion (AIC).
akaike_info_criterion_lsq(ssr, n_params, ...)	假设观测值是高斯分布的，计算AIC
bayesian_blocks(t[, x, sigma, fitness])	用Scargle的贝叶斯块计算数据的最优分割
bayesian_info_criterion(log_likelihood, ...)	给定在估计（或分析导出）参数处评估的似然函数的对数，参数数量和样本数量，计算 Bayesian Information Criterion/贝叶斯信息准则 (BIC)
bayesian_info_criterion_lsq(ssr, n_params, ...)	假设观测值来自高斯分布，计算BIC
binned_binom_proportion(x, success[, bins, ...])	连续变量x的bin中的二项式比例和置信区间
binom_conf_interval(k, n[, conf, interval])	给定n次实验k次成功的二项式比例置信区间
biweight_location(data[, c, M, axis])	计算 biweight 的位置
biweight_midcorrelation(x, y[, c, M, ...])	计算两个变量的 biweight midcorrelation (中相关性)
biweight_midcovariance(data[, c, M, ...])	计算多个变量对之间的 biweight midcovariance (中协方差)
biweight_midvariance(data[, c, M, axis, ...])	计算 biweight midvariance (中方差)
biweight_scale(data[, c, M, axis, ...])	计算 biweight scale (量级)
bootstrap(data[, bootnum, samples, bootfunc])	对numpy array执行引导重采样
cdf_from_intervals(breaks, totals)	从一对数组构造一个可调用的分段线性CDF
circcorrcoef(alpha, beta[, axis, ...])	计算两个循环数据阵列之间的circular correlation coefficient
circmean(data[, axis, weights])	计算循环数据数组的 circular mean angle
circmoment(data[, p, centered, axis, weights])	计算循环数据数组的第p个 trigonometric circular moment
circvar(data[, axis, weights])	计算循环数据数组的circular variance
fold_intervals(intervals)	将加权间隔折叠到间隔(0,1)
freedman_bin_width(data[, return_bins])	使用Freedman-Diaconis规则返回最佳直方图bin宽度

函数名称	函数功能
`histogram`(a[, bins, range, weights])	更强的直方图函数，提供自适应binning
`histogram_intervals`(n, breaks, totals)	分段恒定权重(piecewise-constant weight)的直方图函数
`interval_overlap_length`(i1, i2)	计算两个区间的重叠长度
`jackknife_resampling`(data)	在numpy阵列上执行jackknife重采样
`jackknife_stats`(data, statistic[, conf_lvl])	根据jackknife重采样执行jackknife估计
`knuth_bin_width`(data[, return_bins, quiet])	使用Knuth规则返回最优的直方图bin宽度
`kuiper`(data[, cdf, args])	计算Kuiper统计量
`kuiper_false_positive_probability`(D, N)	计算Kuiper统计量的假阳性概率
`kuiper_two`(data1, data2)	通过计算Kuiper统计量来比较两个样本
`mad_std`(data[, axis, func, ignore_nan])	使用中值绝对偏差/median absolute deviation (MAD)计算可靠的标准偏差
`median_absolute_deviation`(data[, axis, ...])	计算中值绝对偏差 (MAD).
`poisson_conf_interval`(n[, interval, sigma, ...])	在给定观察计数下的泊松参数置信区间Poisson
`rayleightest`(data[, axis, weights])	执行均匀性的瑞利测试 (Rayleigh test of uniformity)
`scott_bin_width`(data[, return_bins])	使用Scott的规则返回最优的直方图bin宽度
`sigma_clip`(data[, sigma, sigma_lower, ...])	对提供的数据执行sigma-clipping
`sigma_clipped_stats`(data[, mask, ...])	对提供的数据计算sigma-clipped统计
`signal_to_noise_oir_ccd`(t, source_eps, ...)	计算在光学/ 红外波段用CCD中观测的源的信噪比
`vonmisesmle`(data[, axis])	为von Mises分布参数计算最大似然估计量 (Maximum Likelihood Estimator, MLE)
`vtest`(data[, mu, axis, weights])	当替代假设H1具有已知的平均角度`mu`时，执行均匀性瑞利测试

1.4 Nuts and Bolts

1.4.1 配置系统(`astropy.config`)

astropy的配置系统是为了让用户能够对astropy或附属程序包中使用的各种参数尽心控制，而无需深入研究源代码来进行更改。

函数名称	函数功能
`get_cache_dir` ()	确定Astropy缓存目录名称，如果该目录不存在则创建该目录
`get_config` ([packageormod, reload])	获取与特定包或模块关联的配置对象或配置部分
`get_config_dir` ([create])	确定Astropy配置目录名称，如果该目录不存在则创建该目录
`reload_config` ([packageormod])	从要求的程序包/模块的根程序包的配置文件中，重新加载配置设置

1.4.2 I/O 注册表(`astropy.io.registry`)

I/O registry子模块用来定义可用于Table和NDData类的读取器/写入器

注：这只是给需要自定义的情况下用的

函数名称	函数功能
`register_reader` (data_format, data_class, ...)	注册阅读器功能.
`register_writer` (data_format, data_class, ...)	注册表格写入功能
`register_identifier` (data_format, data_class, ...)	将标识符函数与特定数据类型相关联
`identify_format` (origin, data_class_required, ...)	通过标识符循环查看哪些格式可以匹配
`get_reader` (data_format, data_class)	获取 `data_format` 的读取器
`get_writer` (data_format, data_class)	获取 `data_format` 的写入器
`read` (cls, *args[, format])	读入数据
`write` (data, *args[, format])	写入数据
`get_formats` ([data_class, readwrite])	以表格形式获取已注册I / O格式的列表
`delay_doc_updates` (cls)	让上下文管理器(contextmanager)在注册读写器时禁用文档更新

1.4.3 日志记录系统 (Logging System)

Astropy日志记录系统旨在为用户提供灵活的选择，如决定显示、获得、发送(到文件)哪些日志消息。

按Astropy设定的步骤打印的所有消息都应使用内置的日志记录工具（正常的print() 调用只能由明确要求打印输出的步骤完成）。 消息的级别有如下几个：

- DEBUG：详细信息，通常只有在诊断问题时才有意义。
- INFO：传达有关当前任务信息的消息，并确认事情按预期工作
- WARNING：表明发生了意外事件，并且可能需要用户操作。
- ERROR：表示一个更严重的问题，包括例外情况

  默认情况下，只显示WARNING和ERROR消息，并将其发送到位于`~/.astropy/astropy.log`的日志文件（假设该文件是可写的）

(后面有几节是给developer/开发人员看的，就没有整理了)

---

2. Astropysics

2.1 总概述

Astropysics是一个包含各种用于简化、分析和可视化天文数据的实用程序和算法的库。最重要的是，它鼓励使用者利用Python的现有功能，使这种快速、简单，并且像尖端科学甚至可以实现的那样轻松。当然，还有其他Python软件包具有这个项目的一些功能，但是这个项目的目标是将所有这些功能集成在一起，并以尽可能直接的方式进行交互。

> Astropysics的功能目前正在纳入Astropy项目。Astropy是一个更广泛的包，为了避免浪费资源，最初进入发展中的Astropysics的大部分功能已转移到Astropy。因此，Astropysics不再获得新的功能。在接下来的几个Astropy版本中，我们预计它将包含Astropysics的所有功能（以及更多）

2.2 功能目录

Astropysics分为两个主要部分——核心模块包含用于计算和组织数据的函数和类别，以及包含许多有用的小范围天文应用的GUI模块。

2.2.1 Core Modules核心模块

Module	中文名称
constants –physical constants and cosmological calculations	物理常数和宇宙学计算
coords – coordinate classes and coordinate conversions	坐标类和坐标转换
models –astronomy-specific models for pymodelfit	pymodelfit的天文专用模型
objcat –flexible, dynamically updated object catalog	适用性强的动态更新的目标目录
ccd – image processingtools	CCD图像处理工具
spec – spectrumand SED classes and tools	光谱和SED光谱能量分布工具
phot –photometry/flux measurement classes and tools	光度/光流量测量类工具
obstools –miscellaneous tools for observation	各种观测工具
plotting –astronomy-oriented matplotlib and mayavi plotting tools	天文类专用绘图工具
pipeline –classes for data reduction and analysis pipelines	数据压缩和分析
publication –tools for preparing astronomy papers in LaTeX	在LaTeX中准备论文的工具
utils – utilityclasses and functions	实用程序类和函数
config –configuration and setup	配置和设置

2.2.2 GUI applications（GUI模块）

GUI MODULE	中文名称
Spylot –Spectrum Plotter	光谱绘图仪
Spectarget – MOSTargeting	MOS

2.3.2 coords 坐标类和坐标转换

> coordinate classes and coordinate conversions

（1）概述

该模块包含指定天体位置的函数，以及坐标转换和距离计算，包括宇宙距离和红移计算。

在这之前应先导入：

```
from astropysics.coords import CoordinateSystem
```

参考网址

(2) coords.coordsys– coordinate systems and transforms

> 该模块包含表示空间，天体和地球坐标系统坐标的归类，以及许多坐标系之间的转换函数

包含以下类和函数:

class `astropysics.coords.coordsys.AngularCoordinate(inpt=None,sghms=None, range=None, radians=False)`

> `getDmsStr(secform='%05.2f', sep=(u'xb0',"'", '"'), sign=True, canonical=False, inclzero=True)`
>
> `getHmsStr(secform=None, sep=('h', 'm','s'), canonical=False, inclzero=True)`

class `astropysics.coords.coordsys.AngularSeparation(*args)`

> `separation3d(*zord1*, *zord2*, *usez=False*, **\*kwargs*)`

class `astropysics.coords.coordsys.CIRSCoordinates(*args,**kwargs)`

class `astropysics.coords.coordsys.CoordinateSystem`

> `delTransform(fromclass, toclass)`
>
> `convert(tosys)`
>
> `delTransform(fromclass, toclass)`
>
> `getTransform(fromclass, toclass)`

class `astropysics.coords.coordsys.EclipticCoordinatesCIRS`

class `astropysics.coords.coordsys.EclipticCoordinatesEquinox`

class `astropysics.coords.coordsys.EpochalLatLongCoordinates`

class `astropysics.coords.coordsys.EquatorialCoordinatesBase`

> `convert(tosys, optimize=False)`

class `astropysics.coords.coordsys.EquatorialCoordinatesEquinox`

class `astropysics.coords.coordsys.FK4Coordinates`

class `astropysics.coords.coordsys.FK5Coordinates`

class `astropysics.coords.coordsys.GCRSCoordinates`

class `astropysics.coords.coordsys.GalacticCoordinates`

class `astropysics.coords.coordsys.HorizontalCoordinates`

class `astropysics.coords.coordsys.ICRSCoordinates`

class `astropysics.coords.coordsys.ITRSCoordinates`

class `astropysics.coords.coordsys.LatLongCoordinates`

> `convert(tosys, optimize=False)`
>
> `getCoordinateString(sep=' ', labels=False,canonical=False, hmslong=False)`
>
> `matrixRotate(matrix, apply=True,fixrange=True, unitarycheck=False)`

class `astropysics.coords.coordsys.RectangularCoordinates`

class `astropysics.coords.coordsys.RectangularGCRSCoordinates`

class `astropysics.coords.coordsys.RectangularGeocentricEclipticCoordinates`

class `astropysics.coords.coordsys.RectangularICRSCoordinates`

class `astropysics.coords.coordsys.SupergalacticCoordinates`

class `astropysics.coords.coordsys.angular_string_to_dec`

class `astropysics.coords.coordsys.objects_to_coordinate_arrays`

(3) coords.ephems——ephemerides and proper motions

> 该模块包含用于计算太阳系物体的星历表和物理位置的工具和实用程序，以及用于对外太阳系物体进行自适应计算。

包含的类有

`astropysics.coords.ephems.EphemerisObject(name,validjdrange=None`

`astropysics.coords.ephems.KeplerianObject(**kwargs)`

`astropysics.coords.ephems.ProperMotionObject(name,ra0, dec0, dra=0, ddec=0, epoch0=2000, distpc0=None, rv=0, coordclass=None)`

`astropysics.coords.ephems.earth_pos_vel(jd,barycentric=False, kms=True)`

`astropysics.coords.ephems.get_solar_system_ephems(objname,jds=None, coordsys=None)`

`astropysics.coords.ephems.list_solar_system_objects()`

`astropysics.coords.ephems.set_solar_system_ephem_method(meth=None)`

(4) coords.funcs –coordinate and distance utility functions

> 该模块包含坐标转换和坐标系计算的功能。它还包括距离相关的计算，包括扩展宇宙学中的距离

包含的类有:

`astropysics.coords.funcs.angular_to_physical_size(angsize,zord, usez=False, **kwargs)`

`astropysics.coords.funcs.cartesian_to_cylindrical(x,y, z, degrees=False)`

`astropysics.coords.funcs.cartesian_to_polar(x,y, degrees=False)`

`astropysics.coords.funcs.cartesian_to_spherical(x,y, z, degrees=False)`

`astropysics.coords.funcs.colatitude_to_latitude(theta,degrees=False)`

`astropysics.coords.funcs.cosmo_dist_to_z(d,derr=None, disttype=0, inttol=1e-06, normed=False, intkwargs={})`

`astropysics.coords.funcs.cosmo_z_to_H(z,zerr=None)`

`astropysics.coords.funcs.cosmo_z_to_dist(z,zerr=None, disttype=0, inttol=1e-06, normed=False, intkwargs={})`

`astropysics.coords.funcs.cylindrical_to_cartesian(s,t, z, degrees=False)`

`astropysics.coords.funcs.earth_rotation_angle(jd,degrees=True)`

`astropysics.coords.funcs.equation_of_the_equinoxes(jd)`

`astropysics.coords.funcs.equation_of_the_origins(jd)`

`astropysics.coords.funcs.geocentric_to_geographic_latitude(geoclat)`

`astropysics.coords.funcs.geographic_to_geocentric_latitude(geoglat)`

`astropysics.coords.funcs.greenwich_sidereal_time(jd,apparent=True)`

`astropysics.coords.funcs.latitude_to_colatitude(lat,degrees=False)`

`astropysics.coords.funcs.match_coords(a1,b1, a2, b2, eps=1, mode='mask')`

`astropysics.coords.funcs.match_nearest_coords(c1,c2=None, n=None)`

`astropysics.coords.funcs.obliquity(jd,algorithm=2006)`

`astropysics.coords.funcs.offset_proj_sep(rx,ty, pz, offset, spherical=False)`

`astropysics.coords.funcs.physical_to_angular_size(physize,zord, usez=True, objout=False, **kwargs)`

`astropysics.coords.funcs.polar_to_cartesian(r,t, degrees=False)`

`astropysics.coords.funcs.radec_str_to_decimal(*args)`

`astropysics.coords.funcs.separation_matrix(v,w=None, tri=False)`

`astropysics.coords.funcs.sky_sep_to_3d_sep(pos1,pos2, d1, d2)`

`astropysics.coords.funcs.spherical_to_cartesian(r,t, p, degrees=False)`

2.3.3 models 建模模块

> astronomy-specific models for pymodelfit

（1）概述

该模块使用pymodelfit软件包来定义一些天文学特有的数据拟合模型，请注意，pymodelfit最初是作为astropysics的一部分编写的，因此它经常使用本模块中的模型，并与其他部分的astropysics紧密集成。

在使用之前需要导入:

```
from pymodelfit import ...
```

（2）所包含的API有

class `astropysics.models.BlackbodyModel(unit='wl')`

> `getFlux(x,radius=None, distance=None)`
>
> `getFluxDistance(radius)`
>
> `getFluxRadius(distance)`
>
> `setFlux(radius,distance)`
>
> `setIntensity()`
>
> `wienDisplacementLaw(peakval)`

class `astropysics.models.BurkertModel`

class `astropysics.models.DeVaucouleursModel`

class `astropysics.models.EinastoModel`

class `astropysics.models.ExponentialDiskModel`

class `astropysics.models.ExponentialSechSqDiskModel`

class `astropysics.models.GaussHermiteModel`

> `gaussHermiteMoment(l, f=None, lower=-inf,upper=inf)`

class `astropysics.models.HernquistModel`

class `astropysics.models.InclinedDiskModel`

class `astropysics.models.JaffeModel`

class `astropysics.models.King2DrModel`

class `astropysics.models.King3DrModel`

class `astropysics.models.MoffatModel`

class `astropysics.models.NFWModel`

> `static Cvir_to_Mvir(Cvir, z=0)`
>
> `staticMvir_to_Cvir(Mvir, z=0)`
>
> `staticMvir_to_Rvir(Mvir, z=0)`
>
> `staticMvir_to_Vmax(Mvir, z=0)`
>
> `staticMvir_to_Vvir(Mvir, z=0)`
>
> `staticRvirMvir_to_Vvir(Rvir, Mvir)`
>
> `getMv(z=0)`
>
> `getRhoMean(r)`
>
> `getV(r,**kwargs)`
>
> `getVmax()`
>
> `setC(c,Rvir=None, Mvir=None, z=0)`
>
> `toAlphaBetaGamma()`

class `astropysics.models.NFWProjectedModel`

> `integrateCircular(lower,upper, method=None, kwargs)`

class `astropysics.models.PlummerModel`

class `astropysics.models.RoundBulgeModel`

class `astropysics.models.SchechterLumModel`

class `astropysics.models.SchechterMagModel`

class `astropysics.models.SersicModel`

> `sbfit(r, sb, zpt=0, **kwargs)`
>
> `sbplot(lower=None,upper=None, data=None, n=100, zpt=0, clf=True)`

2.3.4 objcat 模块

> flexible, dynamically updated object catalog

(1) 概述

objcat模块包含用于生成对象目录的类和函数。 该目录的独特之处在于它存储源信息，并且可以动态导出数量，这些数量在源更改时动态更新。 它还提供了将这些对象目录安全地保存在数据库中的机制，并且不久将包括一个Web界面以通过互联网与目录进行交互。

参考链接

（2）所包含的API

class astropysics.objcat.ActionNode(parent,name=None)

class astropysics.objcat.AstronomicalObject(parent=None, name='default Name')

class astropysics.objcat.Catalog(name='default Catalog', parent=None)

> getFieldNames()
>
> getFieldValueNodes(fieldname,value)
>
> insertInto(insertnode)
>
> isRoot()
>
> locateName(name)
>
> mergeNode(node,skipdup=False, testfunc=None)

class `astropysics.objcat.CatalogNode(parent)`

exception `astropysics.objcat.CycleError(message)`

exception `astropysics.objcat.CycleWarning(message)`

class `astropysics.objcat.DerivedValue`

class `astropysics.objcat.Field`

class `astropysics.objcat.FieldNode`

static `setToSourceAtNode`

class `astropysics.objcat.FieldValue`

class `astropysics.objcat.GraphAction`

class `astropysics.objcat.LinkField`

class `astropysics.objcat.LinkValue`

class `astropysics.objcat.MatplotAction`

class `astropysics.objcat.ObservedErroredValue`

class `astropysics.objcat.ObservedValue`

class` `astropysics.objcat.PlottingAction2D`

class `astropysics.objcat.SEDField`

class `astropysics.objcat.Source`

exception `astropysics.objcat.SourceDataError`

class `astropysics.objcat.StructuredFieldNode`

class `astropysics.objcat.TextTableAction`

2.3.5 ccd 模块

> image processing tools （CCD图像处理工具）

（1）概述

CCD模块包含用于处理CCD图像或类似图像的类和函数。详细的分析工具包括光照和光谱 - 该模块主要用于低级操作，如直接查看图像、校准等操作。

参考链接

（2）所包含的API

class `astropysics.ccd.ASinhScaling`

class `astropysics.ccd.ArrayImage`

class `astropysics.ccd.CCDImage)`

> `activateRange(range)`
>
> `clipInvalids(action='mask')`
>
> `clipOutliers(limits=(1,99), percentage=True, action='noclipmedian')`
>
> `clipSigma(sigma=12,fullsig=True, action='noclipmedian')`
>
> `clipThreshold(thresh=0,action='mask', comp='<')`
>
> `offsetData(offset='minp1')`
>
> `plotHist(perc=None,gauss=None, clf=True, **kwargs)`
>
> `plotImage(valrange='p99',flipaxis=None, invert=False, cb=True, scalebar=None, axes='image',clickinspect=True, clf=True, colormap='gray', **kwargs)`
>
> `setScaling(scalefunc=None,**kwargs)`

class `astropysics.ccd.CCDNoise`

class `astropysics.ccd.DataScaling`

class `astropysics.ccd.ExponentialScaling`

class `astropysics.ccd.FitsImage`

class `astropysics.ccd.GaussianNoise`

class `astropysics.ccd.ImageBiasSubtractor`

> `plInteract(data,pipeline, elemi)`
>
> `plProcess(data,pipeline, elemi)`
>
> `subtractFromImage(image)`

class `astropysics.ccd.ImageCombiner`

class `astropysics.ccd.ImageFlattener`

class `astropysics.ccd.LinearScaling`

class `astropysics.ccd.LogScaling`

class `astropysics.ccd.NoiseModel`

class `astropysics.ccd.PoissonNoise`

class `astropysics.ccd.PowerScalingastropysics.ccd.SurfaceBrightnessScaling`

class `astropysics.ccd.UniformNoise`

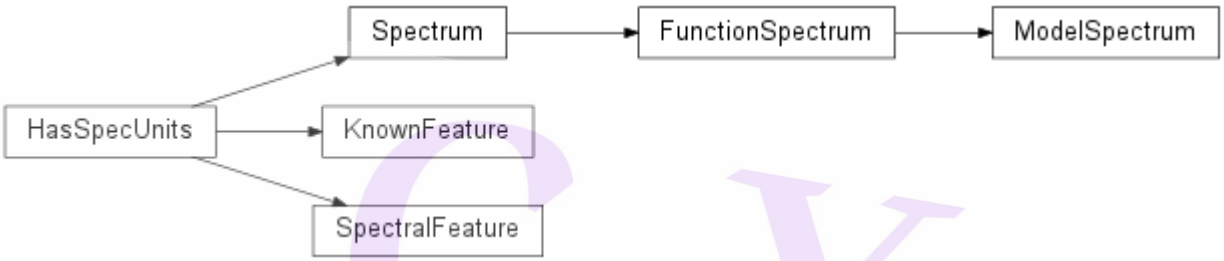
2.3.6 spec 模块

> spectrum and SED classes and tools

（1）概述

spec模块包含侧重于绘制和分析光谱、SED以及相关实用函数。

下面是该模块的结构：



具体的函数说明请参考链接

(2) 所包含的API

class `astropysics.spec.FunctionSpectrum(xi,fluxf, errf=None, ivarf=None, unit='wl')`

class `astropysics.spec.HasSpecUnits(unit)`

class `astropysics.spec.KnownFeature(loc,name='', strength=None, unit='wavelength')`

class `astropysics.spec.ModelSpectrum(xi,model, noisetype=None, noisefunc=None, unit='wl')`

class `astropysics.spec.SpectralFeature(extent, unit='wavelength',continuum='fromspec')`

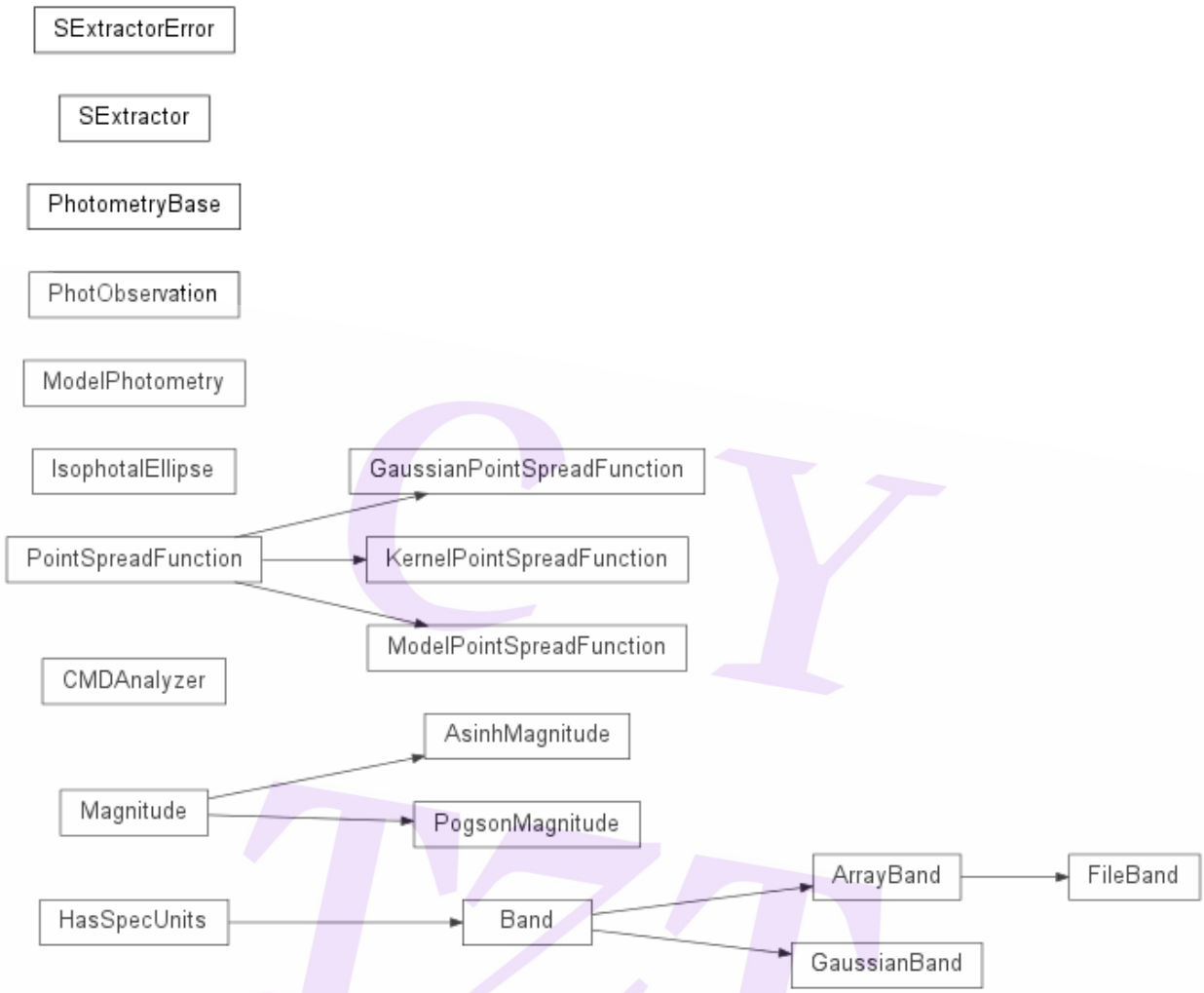
class `astropysics.spec.Spectrum(x, flux,err=None, ivar=None, unit='wl', name='', copy=True, sort=True)`

2.3.7 phot 模块

> photometry/flux measurement classes and tools

（1）概述：phot模块包含用于测光或其他基于流量的类别和功能。相关链接请参考此链接

结构图：



(2) 所包含的API

class `astropysics.phot.ArrayBand(x, S, copyonaccess=True, normalized=True,unit='angstroms', name=None)`

class `astropysics.phot.AsinhMagnitude(b=1e-10)`

class `astropysics.phot.Band`

class `astropysics.phot.CMDAnalyzer`

class `astropysics.phot.FileBand(fn, type=None)`

class `astropysics.phot.GaussianBand`

class `astropysics.phot.GaussianPointSpreadFunction(sigma=1)`

class `astropysics.phot.IsophotalEllipse(imdata,isolevel=None)`

class `astropysics.phot.KernelPointSpreadFunction(kernelarr2d)`

class `astropysics.phot.Magnitude`

class `astropysics.phot.ModelPhotometry(model, magzpt=0)`

class `astropysics.phot.ModelPointSpreadFunction(model)`

class `astropysics.phot.PhotObservation`

class `astropysics.phot.PhotometryBase`

class `astropysics.phot.PogsonMagnitude`

class `astropysics.phot.PointSpreadFunction`

class `astropysics.phot.SExtractor`

2.3.8 obstools 模块

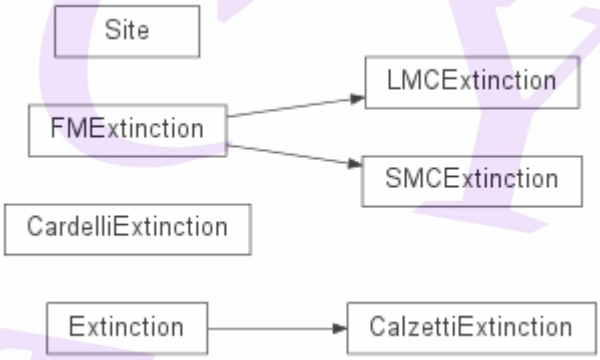
> miscellaneous tools for observation

（1）概述

obstools模块存储了用于观测的函数，以及用作各种校正和简单计算的模块。参考链接

> 请注意，在整个模块中，假定datetime中使用的UTC与UT1（用于计算的UT）相同。

结构图:



（2）所包含的API

class `astropysics.obstools.CalzettiExtinction(A0=1)`

class `astropysics.obstools.CardelliExtinction(EBmV=1, Rv=3.1)`

class `astropysics.obstools.Extinction(f=None, A0=1)`

class `astropysics.obstools.FMExtinction(C1, C2, C3, C4, x0, gamma,EBmV=1, Rv=3.1)`

class `astropysics.obstools.LMCExtinction(EBmV=0.3, Rv=3.41)`

class `astropysics.obstools.SMCExtinction(EBmV=0.2, Rv=2.74)`

class `astropysics.obstools.Site(lat, long, alt=0, tz=None,name=None)`

2.3.9 plotting 模块

> astronomy-oriented matplotlib and mayavi plotting tools

（1）概述

绘图模块包含类和功能，以帮助制作对天体物理有用的图。其中，这个模块中的二维图都使用matplotlib软件包，而三维图是matplotlib和mayavi的结合使用。

具体参考文档链接

（2）所包含的分类和函数（API）

class `astropysics.plotting.ScatterLasso`

> `callback(verts)`
>
> `deactivateLasso()`
>
> `getLastInds()`
>
> `getLastKeys(d=None,xattr='x', yattr='y')`

> `getLastXYs()`
>
> `matchLastXYs(xyin,map=None)`

class `astropysics.plotting.add_mapped_axis`

class `astropysics.plotting.ax3d_animate`

> 使用mplot3d 3D绘图工具为matplotlib中的图形制作动画。

class `astropysics.plotting.cumulative_plot`

class `astropysics.plotting.dual_value_plot`

class `astropysics.plotting.logerrplot`

class `astropysics.plotting.make_movie`

class `astropysics.plotting.mlab_anaglyph`

class `astropysics.plotting.mlab_animate_rotzoom`

> 这会使用`mlab.animate()`机制生成一个动画，该机制可旋转场景并可能使用固定焦点进行放大和缩小。

class `astropysics.plotting.mlab_camera`

class `astropysics.plotting.mlab_checkerboard`

class `astropysics.plotting.mpl_context`

class `astropysics.plotting.mvi_texture`

class `astropysics.plotting.plot_histogram`

class `astropysics.plotting.scatter4d`

class `astropysics.plotting.scatter_select`

> 将快速散点图绘制到套索对象并返回将随选择更新的对象

class `astropysics.plotting.split_histograms`

class `astropysics.plotting.square_subplot_dims`

class `astropysics.plotting.subplots_adjust_points`

2.3.10 pipeline 模块

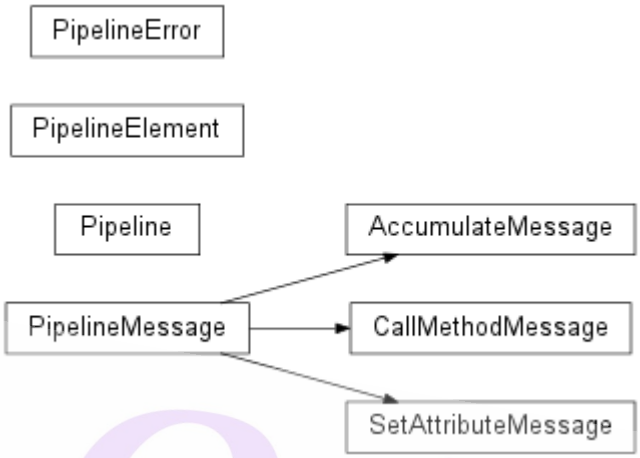
> classes for data reduction and analysis pipelines

(1) 概述

pipeinne模块包含用于构建数据pipe的类和实用程序——处理输入数据的线性结构，将其传递到所有pipe阶段。

具体说明文档的参考链接

结构图:



（2）所包含的类和函数（API）

class `atropysics.pipeline.AccumulateMessage`

class `astropysics.pipeline.CallMethodMessage`

class `astropysics.pipeline.Pipeline`

class `astropysics.pipeline.PipelineElement`

> `plInteract(data,pipeline, elemi)`
>
> `plProcess(data,pipeline, elemi)`
>
> `resaveData(data,pipeline, elemi)`

exception `astropysics.pipeline.PipelineError`

class `astropysics.pipeline.PipelineMessage`

> `deliverMessage(elem)`
>
> `isTarget(elem)`

class `astropysics.pipeline.SetAttributeMessage`

2.3.11 publication 模块

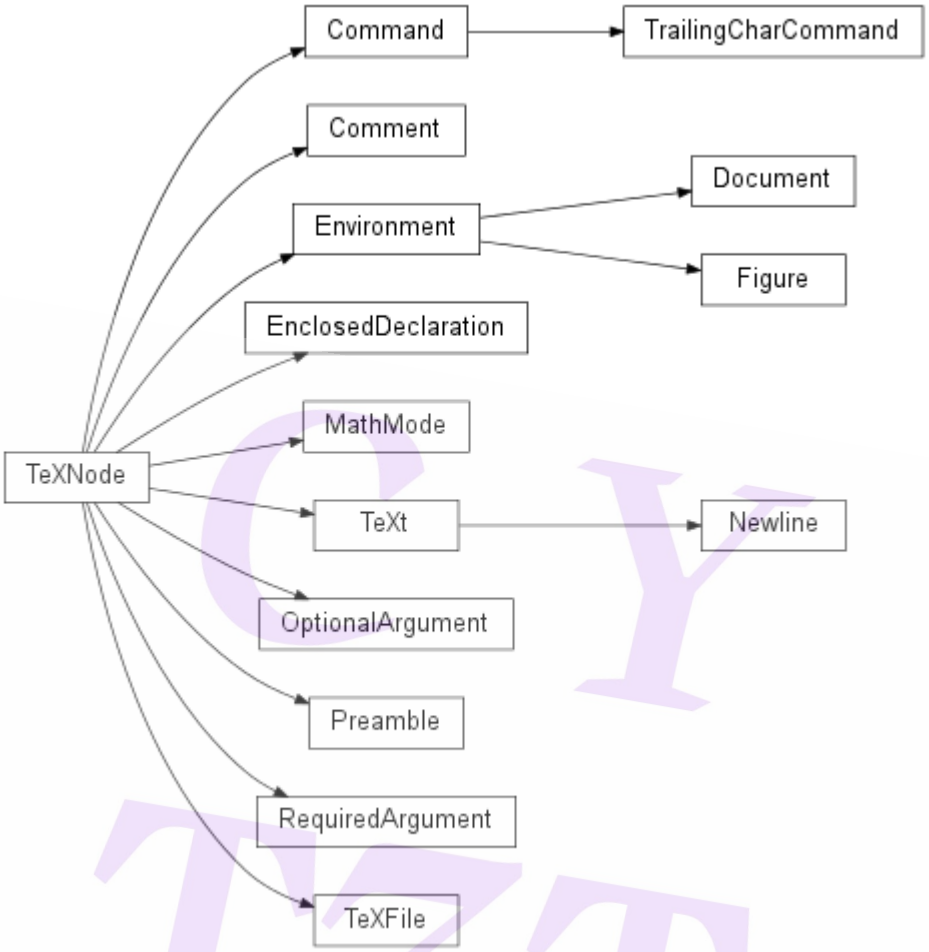
> tools for preparing astronomy papers in LaTeX

（1）概要

该模块用于帮助常见天文学期刊上的论文或其他出版物的类和函数。这里的工具是基于使用LaTeX作为实际的创作工具。

具体使用说明参考链接

结构图：



(2) 所包含的类和函数（API）

class `astropysics.publication.Command`

> ALaTeX command

class `astropysics.publication.Comment`

> TeX文件的单行注释

class `astropysics.publication.Document`

class `astropysics.publication.EnclosedDeclaration`

class `astropysics.publication.Environment`

> A LaTex environment

static `getEnvironments()`

> `getSelfText()`.
>
> `postParse(nodes)`

static `registerEnvironment(envclass)`

static `unregisterEnvironment(envclass)`

class `astropysics.publication.Figure(parent, content, envname=None)`

class `astropysics.publication.MathMode(parent, content)`

class `astropysics.publication.Newline(parent)`

class `astropysics.publication.OptionalArgument(parent,text)`

class `astropysics.publication.Preamble(parent, content)`

`getSelfText()`

class `astropysics.publication.RequiredArgument(parent, text)`

class `astropysics.publication.TeXFile(fn=None, flatteninputs=False)`

class `astropysics.publication.TeXNode(parent)`

> `children = ()`
>
> `getSelfText()`
>
> `isLeaf()`——ReturnsTrue if this node is a leaf (has no children)
>
> `isRoot()`——Returns True if this node is aroot (has no parent)
>
> `prune(prunechildren=True)`
>
> `visit(func)`

class `astropysics.publication.TeXt(parent, text)`

> 存储通用文本的节点

class `astropysics.publication.TrailingCharCommand(parent, content)`

class `astropysics.publication.environment_factory(parent,texstr)`

class `astropysics.publication.prep_for_apj_pub`

class `astropysics.publication.prep_for_arxiv_pub`

class `astropysics.publication.text_to_nodes`

> 将字符串转换为相应TeXNode对象的列表

2.3.12 utils 模块

> utility classes and functions

（1）概述

utils模块包含整个astropysics中多个地方使用的通用实用程序的类和功能。 其中一些是特定于宇宙物理学的算法，而另一些则是更多的python技巧。utils模块由三个子模块组成。这些子模块彼此相当不同，但主要的目的是使所有这些子模块并不特定于天文应用。因此，它们可以在astropysics或其他任何被认为有用的地方重复使用。

具体参考链接

（2）所包含的类和函数（API）

第一个子模块：utils.gen（gen模块包含用于整个astropysics各个地方的通用实用程序的类和函数。）

class `astropysics.utils.gen.DataObjectRegistry`

> 用于注册整个模块中使用的数据集的类，并使用字符串名称访问数据

class `astropysics.utils.gen.SymmetricMapping`

> 两个方向上映射

class `astropysics.utils.gen.add_docs`

> 该类是一个装饰器，指示装饰对象应该将其部分（或全部）文档字符串替换为另一个对象的文档字符串。

class `astropysics.utils.gen.add_docs_and_sig`

class `astropysics.utils.gen.change_indentation`

class `astropysics.utils.gen.check_type`

> 调用此函数检查值是否与提供的类型匹配

第二个子模块：utils.alg

utils.alg – common/repeated algorithms（alg模块包含基本算法，数学运算技巧和通用数据处理任务）

`astropysics.utils.alg.angle_axis(matrix, degrees=True)`

> 计算给定旋转矩阵的旋转角度和旋转轴

`astropysics.utils.alg.centroid(val, axes=None, offset=None)`

`astropysics.utils.alg.crossmask(x, threshold=0, belowtoabove=True)`

`astropysics.utils.alg.estimate_background(arr, method='median')`

`astropysics.utils.alg.intrinsic_to_observed_ellipticity(ei, i,degrees=True)`

`astropysics.utils.alg.lin_to_log_rescale(val, lower=1, upper=3,base=10)`

`astropysics.utils.alg.nd_grid(*vecs)`

`astropysics.utils.alg.nearestsorted(a, val)`

> 搜索排序的序列以获取最接近的值

`astropysics.utils.alg.observed_to_intrinsic_ellipticity(eo, i,degrees=True)`

`astropysics.utils.alg.rotation_matrix(angle, axis='z', degrees=True)`

> 在笛卡尔坐标系中生成一个3x3旋转矩阵，以围绕指定的轴进行转置。

`astropysics.utils.alg.rotation_matrix_xy(x, y)`

`astropysics.utils.alg.sigma_clip(data, sig=3, iters=1,cenfunc='median', varfunc=<function var at 0x10c747e60>, maout=False)`

`classastropysics.utils.stats.Pca(data, names=None)`

> 主成分分析（PCA）类

`astropysics.utils.stats.binned_weights(values,n, log=False)`

> 产生一个权重数组，这些权重是通过将这些值细分为n个分箱生成的，以便每个分箱在总数中具有相同的份额。

`astropysics.utils.stats.biweight_midvariance` `astropysics.utils.stats.interquartile_range`

`astropysics.utils.stats.moments`

第三个子模块：utils.io

> **utils.io（io模块包含用于加载和保存天文学中使用的各种相关格式的数据的类和函数，以及用于检索内置数据的便利功能）**

`astropysics.utils.io.FixedColumnDataParser(skiprows=0, firstcolindx=1,commentchars='#')`

`astropysics.utils.io.VOTable(*args,**kwargs)`

`astropysics.utils.io.VOTableReader(s, filename=True)`

`astropysics.utils.io.fpickle(object,fileorname, usecPickle=True, protocol=None, append=False)`

`astropysics.utils.io.funpickle(fileorname,number=0, usecPickle=True)`

`astropysics.utils.io.get_data(dataurl,asfile=False, localfn=None)`

`astropysics.utils.io.get_data_download_reporter()`

`astropysics.utils.io.get_data_store()`

`astropysics.utils.io.get_package_data(dataname)`

`astropysics.utils.io.load_all_deimos_spectra(dir='.',pattern='spec1d*', extraction='horne', smoothing=None, verbose=True)`

`astropysics.utils.io.load_deimos_spectrum(fn,plot=False, extraction='horne', retdata=False, smoothing=None)`

`astropysics.utils.io.load_tipsy_format(fn)`

`astropysics.utils.io.loadtxt_text_fields(fn,fieldline=1, asrecarray=True, updatedict=None, **kwargs)`

`astropysics.utils.io.open_with_pyfits(*args,**kwargs)`

`astropysics.utils.io.set_data_store(store=True)`

2.3.13 config 模块

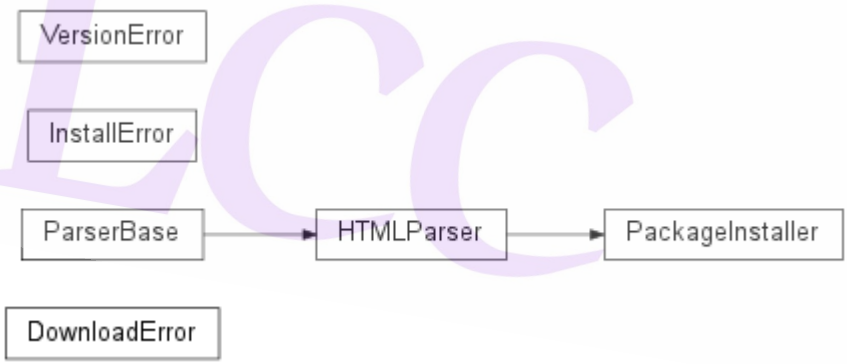
> configuration and setup

（1）概述

config模块包含管理和访问持久性astropysics配置的功能，它还包括安装推荐软件包并设置ipython环境的实用程序。

使用说明文档参考链接

结构图如下：



（2）所包含的类和函数（API）

exception `astropysics.config.DownloadError`

exception `astropysics.config.InstallError`

class `astropysics.config.PackageInstaller`

> 表示要下载并安装的python软件包

class `astropysics.config.add_project(name,dir=None, scriptfile=None)`

> 将新项目添加到项目注册表中

class `astropysics.config.del_project(name)`

class `astropysics.config.get_config(name)`

class `astropysics.config.get_config_dir(create=True)`

class `astropysics.config.get_data_dir(create=True)`

class `astropysics.config.get_projects()`

class `astropysics.config.run_install_tool(sudo='auto')`

class `astropysics.config.run_ipython_setup()`

2.4 GUI Application应用程序

2.4.1 总概述

Astropysics包含各种图形应用程序，用于各种数据分析任务。在交互式使用或作为脚本的一部分使用时，它们的全部功能是可行的，但是有些时候则作为独立的命令行工具运行。

astropysics中使用的另一个重要的相关GUI是来自pymodelfit的FitGUI 。这个图形用户界面可以在任何需要交互式曲线拟合的地方使用。它包括两个GUI，分别是：

Spylot - 光谱绘图助手

Spectarget - MOS瞄准

具体使用说明请参考链接

2.4.2 Spylot GUI

（1）概述

此应用程序是一个基于Traits的谱图绘制/交互式分析工具。它本质上是一个围绕着`astropysics.spec.Spectrum`对象集合的GUI包装和界面。

它可以作为python脚本的一部分运行，也可以通过生成`astropysics.gui.spylot.Spylot`对象或调用`astropysics.gui.spylot.spylot_specs ()`函数在ipython中以交互方式运行。

当您安装astropysics时，还会安装命令行脚本'spylot'。

具体的安装和使用步骤请参考链接

（2）`astropysics.gui.spylot.Spylot`对象

class `astropysics.gui.spylot.Spylot(specs, **traits)`

> 表示spylot应用程序状态

（3）调用`astropysics.gui.spylot.spylot_specs ()`函数

`astropysics.gui.spylot.spylot_specs(specs,block=False, newapp=False)`

> 生成包含提供的光谱序列并显示的Spylot实例

2.4.3 Spectarget GUI

（1）概述

此应用程序是基于Traits的工具，用于交互式识别与多天体谱图一起使用的光谱目标。

（2）分类

class `astropysics.gui.spectarget.DEIMOSMaskMaker(spectargobj)`

class `astropysics.gui.spectarget.MaskMaker(spectargobj)`

class `astropysics.gui.spectarget.ODHandler`

class `astropysics.gui.spectarget.OffsetDialog(cmda, st)`

class `astropysics.gui.spectarget.SpecTarget(*args,**kwargs)`

class `astropysics.gui.spectarget.spec_target(*args)`

> 生成并显示一个SpecTarget GUI界面

---

3. astLib

3.1 概述

`astLib`是一套Python模块，为天文学家提供了一些可以用于天文绘图、统计数据计算、坐标转换以及通过PyWCSTools操作带有世界坐标系统（WCS）信息的FITS图像的工具。其中，PyWCSTools作为astLib的一部分进行开发。

> 具体参考链接：http://astlib.sourceforge.net/

3.2 包含模块

astLib.astCalc	常用计算模块
`astLib.astCoords`	天球坐标操作模块（转换，计算等）
`astLib.astImages`	简单图像任务的模块
`astLib.astPlots`	天文绘图的模块
`astLib.astSED`	光谱能量分布（SED）计算模块
`astLib.astStats`	统计计算模块
`astLib.astWCS`	处理世界坐标系统（WCS）模块

3.3 模块具体功能

3.3.1 astCalc

用于执行常用计算的模块

> 目前该模块的重点是计算给定宇宙学中的距离。 宇宙模型的参数使用模块名称空间中的变量`OMEGA_M0`，`OMEGA_L0`，`OMEGA_R0`，`H0`进行设置

函数名	函数功能
`dl(z)`	以红移z计算光度距离
`da(z)`	以红移z计算角直径距离
`dm(z)`	以红移z计算横向相交距离（自身运动距离）
`dc(z)`	以红移z计算视线相交距离
`dVcdz(z)`	在红移z时计算视线相交体积元
`dl2z(distanceMpc)`	计算给定的亮度距离相对应的红移量z
`dc2z(distanceMpc)`	计算对应距离的红移z
`t0()`	在当前宇宙学参数集中计算z = 0的Gyr中宇宙的年龄
`tl(z)`	计算回溯时间（Gyr），以红移z为当前宇宙学参数集
`tz(z)`	用红移z计算当前宇宙学参数集的宇宙年龄
`tl2z(tlGyr)`	计算与Gyr中给出的回溯时间（tlGyr）相对应的红移z
`tz2z(tzGyr)`	计算对应于Gyr中给出的宇宙年龄（tzGyr）的红移z
`absMag(appMag, distMpc)`	计算给定发光度距离处物体的绝对大小
`Ez(z)`	计算E（z）的值，该值描述当前的一组宇宙参数在红移 z处的哈勃参数演变
`Ez2(z)`	描述当前的一组宇宙参数在红移z处时的哈勃参数演变
`OmegaMz(z)`	在红移z处计算宇宙的物质密度
`OmegaLz(z)`	在红移z处计算宇宙的暗能量密度
`OmegaRz(z)`	在红移z处计算宇宙的辐射密度
`DeltaVz(z)`	计算一维列化区域的密度对比度
`RVirialXRayCluster(kT, z, betaT)`	用X射线温度(K)计算红移z处的星系团的维里半径

3.3.2 astCoords

等值操作（如转换，计算等）

函数名	函数功能
`hms2decimal`	将H:M:S格式的分隔字符串转换为十进制度数
`dms2decimal`	将D:M:S格式转为十进制度数格式。
`decimal2hms`	将十进制度数格式转为H:M:S格式。
`decimal2dms`	将十进制度数格式转为D:M:S格式。
`calcAngSepDeg`	假设有一个切平面投影（十进制度数格式），以十进制度计算天空中两个天体位置（从指定原点分别到两天体的两条向量的夹角必须<90度）。
`shiftRADec`	通过已有的一些与计算新的赤经和赤纬。（十进制格式）
`convertCoords`	在J2000，B1950和Galactic之间转换指定的坐标。（十进制格式）
`calcRADecSearchBox`	计算赤经的极值。

3.3.3 astImages

简单的.fits图像作业（旋转，裁剪部分，制作.png等）。

函数名	函数功能
`clipImageSectionWCS`	从给定天体坐标的图像矩阵中剪取**正方形**或**矩形**部分。 可选地返回剪切部分的更新的WCS，以及原始图像中对应于剪切部分的x，y像素坐标。
`clipImageSectionPix`	从给定像素坐标处的图像数组中剪切正方形或矩形部分。
`clipRotatedImageSectionWCS`	从给定天体坐标的图像矩阵中剪取正方形或矩形部分。 所产生的剪辑旋转和/或翻转，使得North位于顶部，而East位于左侧。 剪辑后的部分的更新后的WCS也会返回。
`clipUsingRADecCoords`	在对应于给定天体坐标的像素坐标处，从图像矩阵中剪取一段。
`scaleImage`	按照给定的比例因子缩放图像矩阵和WCS。
`intensityCutImage`	创建一个 `matplotlib.pylab` 绘图，其中应用了指定的切割强度。
`resampleToTanProjection`	将图像和WCS重采样到切平面投影。 纯粹是为了绘图目的（例如，确保RA，dec。坐标轴垂直）。
`resampleToWCS`	将对应于图像2（具有数据im2Data，WCS im2WCS）的数据重采样到图像1（im1Data，im1WCS）的WCS上。 输出重采样图像的像素与图像1的像素尺寸相同。 此例程用于帮助绘图 - 不建议在输出上执行测光。
`generateContourOverlay`	重新调整用作轮廓叠加的图像矩阵，以便其与背景图像具有相同的尺寸，并生成一组轮廓，该过程使用高斯滤波器对边缘进行平滑处理。
`saveBitmap`	从图像矩阵中创建位图图像; 图像格式由文件扩展名指定。 （例如“.jpg”= JPEG，“.png”= PNG）。
`saveContourOverlayBitmap`	从图像矩阵中创建位图图像，并重叠第二个图像数组生成的一组轮廓。 （在生成轮廓的图像阵列的过程中，可以选择使用高斯滤波器进行预平滑。）
`saveFITS`	将图像矩阵写入新的.fits文件。
`histEq`	对输入的 `numpy.array` 执行直方图均衡。
`normalise`	指定强度，对输入矩阵进行归一化处理。

3.3.4 astPlots

制作天文模块

> 提供SED类的操作，特别是目前支持的Bruzual＆Charlot 2003，Maraston 2005和Percival et al 2009恒星群合成模型。并且提供函数用于计算这些模型中红移的颜色和大小等的演变，以及使用这些模型拟合光度的测量。

函数名	函数功能
DEC_TICK_STEPS	在六十进制模式下定义可能的坐标轴标记步。 dict(字典)格式：{'deg'，'unit'}
RA_TICK_STEPS	在六十进制模式下定义赤经轴上可能的坐标标记步骤。dict(字典)格式：{'deg'，'unit'}
DECIMAL_TICK_STEPS	以十进制度模式定义两个坐标轴上可能的坐标标记步骤。

3.3.5 astSED

执行光谱能量分布（SED）计算的模块

> 该模块提供操作SED的类，特别是目前支持的Bruzual＆Charlot 2003，Maraston 2005和Percival et al 2009恒星种群合成模型。 提供函数用于计算这些红移等模型中颜色和大小的变化，以及使用这些模型拟合宽带光度测量。

函数名	函数功能
makeModelSEDDictList	这个例程为SEDDict字典list（参见mags2SEDDict）使用fitSEDDict进行拟合。
fitSEDDict	使用给定的SED字典列表来匹配给定的SED字典（使用mags2SEDDict制作）。
mags2SEDDict	需要一组相应的AB幅值，不确定性和通带，并返回一个带有“通量”，“通量误差”，“波长”的字典。
mag2Flux	将给定AB幅度和不确定度转化为通量，单位为erg / s / cm ^ 2 / Angs。
flux2Mag	将erg / s / cm ^ 2 / Angs中给定的通量和不确定度转化为AB值。
mag2Jy	在Jy中将AB幅值转换为磁通密度
Jy2Mag	将Jy中的通量密度转换为AB量级

*AB：*平谱SED，用于计算AB系统的幅度。

3.3.6 astStats

执行统计计算。

> 这个模块提供了很少的统计程序。 然而，它确实提供了位置和尺度的重量（健壮）估计量，如Beers等人所述。 1990（AJ，100,32），以及使用双重权重变换的鲁棒最小二乘拟合程序。
>
> 对于广泛的统计模块，建议使用GNU R（http://rpy.sourceforge.net）或SciPy（http://www.scipy.org）的Python绑定。

函数名	函数功能
`mean`	计算一组数字的**平均值**。
`weightedMean`	计算数字的二维列表（值，权重）的**加权平均值**。
`stdev`	计算一组数字的（样本）**标准差**。
`rms`	计算一组数字的**均方差的根**。
`weightedStdev`	计算一组数字的（样本）的**加权标准差**。
`median`	计算一组数字的**中位数**。
`modeEstimate`	模式平均的估计
`MAD`	计算一组数字的中值绝对偏差。
`biweightLocation`	计算一组数字的权重位置估计量（如robust平均值）。
`biweightScale`	计算一组数字的双重比例尺估计量（如一个稳健的标准偏差）。
`biweightClipped`	迭代计算biweight权重的位置和比例，使用sigma裁剪，获取值列表。（在多维列表中，该计算只在第一列上执行，其余列被忽略。）
`biweightTransform`	计算一组值的权重变换。
`OLSFit`	在二维数字列表上执行普通最小二乘拟合。(数据点数>=5)。
`clippedMeanStdev`	计算一组数字的chipped均值和标准偏差。
`clippedWeightedLSFit`	用sigma函数剪取一组数字，再进行加权最小二乘拟合。 最少数据点数是5。
`weightedLSFit`	在数字[x, y, y_error]的三维列表上执行加权最小二乘拟合。
`biweightLSFit`	加权最小二乘拟合，其中使用的权重是残差到先前最佳拟合的权重变换.i.e。
`cumulativeBinner`	累计输入数据。
`binner`	分配输入数据。
`weightedBinner`	对输入数据进行分箱，记录的频率是分箱中的加权总和。

3.3.7 astWCS

用于处理世界坐标系统（WCS）的模块

> 这是PyWCSTools中的一些例程（使用astLib分发）的更高级别的接口。 PyWCSTools是Jessica Mink（http://tdc-www.harvard.edu/software/wcstools/）对WCSTools的简单SWIG包装。 它的目的是使这个接口足够完整，因此不需要直接使用PyWCSTools。

函数名	函数功能
`findWCSOverlap`	查找wcs1和wcs2之间重叠的最小，最大WCS坐标。 返回这些坐标，加上每个wcs的相应像素坐标。 用于剪切两个图像之间的重叠区域。

4. APLpy

4.1总概述

APLpy(Astronomical Plotting Library in Python)，主要用于天文绘图操作。在此之前，需要到官网下载所需的安装包，然后需要提前导入： `import aplpy` 和 `import numpy` 。具体的实用例子请参考链接

4.2 所包含的Functions

函数	功能
`make_rgb_cube`	从三张FITS图像中制作RGB三维图
`make_rgb_image`	从RGB多维数据集或三个FITS文件制作RGB图像
`test([package,test_path, args, plugins, ...])`	运行相关python 脚本

4.3 该模块包所包含的类

该模块一共包含9大类，具体如下：

4.3.1 `aplpy.AxisLabels(parent)` 关于坐标轴的设定

`hide()` Hide the x- andy-axis labels.

`hide_x()` Hidethe x-axis label.

`hide_y()` Hidethe y-axis label.

`set_font(family=None,style=None, variant=None, stretch=None, weight=None, size=None,fontproperties=None)` Set the font of the axis labels.

`set_xpad(pad)` Set the x-axis label displacement, in points.

`set_xposition(position)` Set the position of the x-axis label (‘top’ or ‘bottom’)

`set_xtext(label)` Set the x-axis label text.

`set_ypad(pad)` Set the y-axis label displacement, in points.

`set_yposition(position)` Set the position of the y-axis label (‘left’ or ‘right’)

`set_ytext(label)` Set the y-axis label text.

`show()` Showthe x- and y-axis labels.

`show_x()` Showthe x-axis label.

`show_y()` Showthe y-axis label.

4.3.2 `aplpy.Beam(parent)` 关于光束的设定

`hide()` Hidethe beam隐藏光束

`set(**kwargs)` Modify the beam properties.

`set_alpha(alpha)` Set the alpha value (transparency)

`set_angle(angle)` Set the position angle of the beam on thesky, in degrees.

`set_borderpad(borderpad)` Set the amount of padding within the beamobject, relative to the canvas size.

`set_color(color)` Set the beam color.

`set_corner(corner)` Set the beam location.

`set_edgecolor(edgecolor)` Set the color for the edge of the beam.

`set_facecolor(facecolor)` Set the color for the interior of thebeam.

`set_frame(frame)` Set whether to display a frame around thebeam.

`set_hatch(hatch)` Set the hatch pattern.

`set_linestyle(linestyle)` Set the line style for the edge of the beam.

`set_linewidth(linewidth)` Set the line width for the edge of thebeam, in points.

`set_major(major)` Set the major axis of the beam, in degrees.

`set_minor(minor)` Set the minor axis of the beam, in degrees.

`set_pad(pad)` Set the amount of padding between the beamobject and the image corner/edge, relative to the canvas size.

`show([major,minor, angle, corner, frame, ...])` Displaythe beam shape and size for the primary image

4.3.3 `aplpy.FITSFigure` 关于fits图像绘制和设定

> 该分类基于aplpy.layers.Layers,aplpy.regions.Regions,aplpy.deprecated.Deprecated

`add_beam(*args, **kwargs)` Add a beam to the current figure.

`add_colorbar(*args,**kwargs)` Add a colorbar to thecurrent figure.

`add_grid(*args,**kwargs)` Add a coordinate to thecurrent figure.

`add_label(x, y,text[, relative, color, ...])` Add atext label.

`add_scalebar(length,*args, **kwargs)` Add a scalebar tothe current figure.

`close()` Close the figure and free up the memory.

`hide_colorscale()`

`hide_grayscale(*args,**kwargs)`

`pixel2world(xp,yp)` Convert pixel to world coordinates.

`recenter(x, y[,radius, width, height])` Center the imageon a given position and with a given radius.

`refresh([force])` Refresh the display.

`remove_beam([beam_index])` Removes the beam from the current figure.

`remove_colorbar()` Removes the colorbar from the current figure.

`remove_grid()` Removes the grid from the current figure.

`remove_scalebar()` Removes the scalebar from the current figure.

`save(filename[,dpi, transparent, ...])` Save the currentfigure to a file.

`set_auto_refresh(refresh)` Set whether the display should refreshafter each method call.

`set_nan_color(color)` Set the color for NaN pixels.

`set_system_latex(usetex)` Set whether to use a real LaTeXinstallation or the built-in matplotlib LaTeX.

`set_theme(theme)` Set the axes, ticks, grid, and image colors toa certain style (experimental).

`set_title(title,**kwargs)` Set the figure title

`set_xaxis_coord_type(coord_type)` Set the type of x coordinate.

`set_yaxis_coord_type(coord_type)` Set the type of y coordinate.

`show_arrows(x,y, dx, dy[, width, ...])` Overlay arrowson the current plot.

`show_circles(xw,yw, radius[, layer, zorder])` Overlaycircles on the current plot.

`show_colorscale([vmin,vmid, vmax, pmin, ...])` Show a colorscaleimage of the FITS file.

`show_contour([data,hdu, layer, levels, ...])` Overlaycontours on the current plot.

`show_ellipses(xw,yw, width, height[, ...])` Overlayellipses on the current plot.

`show_grayscale([vmin,vmid, vmax, pmin, ...])` Show agrayscale image of the FITS file.

`show_lines(line_list[,layer, zorder])` Overlay lines on thecurrent plot.

`show_markers(xw,yw[, layer])` Overlay markers on thecurrent plot.

`show_polygons(polygon_list[,layer, zorder])` Overlay polygons onthe current plot.

`show_rectangles(xw,yw, width, height[, ...])` Overlayrectangles on the current plot.

`show_rgb([filename,interpolation, ...])` Show a 3-colorimage instead of the FITS file data.

`show_vectors(pdata,adata[, phdu, ahdu, ...])` Overlayvectors on the current plot.

`world2pixel(xw,yw)` Convert world to pixelcoordinates.

4.3.4 `aplpy.Frame(parent)` 画图框架设定

`set_color(color)` Set color of the frame

`set_linewidth(linewidth)` Set line width of the frame

4.3.5 `aplpy.Grid(parent)` 图像网格点有关参数设定

`hide()`

`set_alpha(alpha)`

> 设置网格线的透明度

`set_color(color)`

> 设置网格线的颜色

`set_linestyle(linestyle)`

`set_linewidth(linewidth)`

`set_xspacing(xspacing)`

> 在经度方向上设置网格线间距

`set_yspacing(yspacing)`

> 在纬度方向上设置网格线间距

4.3.6 `aplpy.Scalebar(parent)` 比例尺/标签设定

`hide()` Hidethe scalebar.

`set(**kwargs)` Modify the scalebar and scalebar properties.

`set_alpha(alpha)` Set the alpha value (transparency).

`set_color(color)` Set the label and scalebar color.

`set_corner(corner)` Set where to place the scalebar.

`set_font([family,style, variant, stretch, ...])` 设置标签的字体样式

`set_font_family(family)`

`set_font_size(size)`

`set_font_style(style)`

`set_font_weight(weight)`

`set_frame(frame)` Set whether to display a frame around thescalebar.

`set_label(label)` Set the label of the scale bar.

`set_length(length)` Set the length of the scale bar.

`set_linestyle(linestyle)` Set the linestyle of the scalebar.

`set_linewidth(linewidth)` Set the linewidth of the scalebar, inpoints.

`show(length[,label, corner, frame, ...])` Overlaya scale bar on the image.

4.3.7 `aplpy.TickLabels(parent)` 坐标轴刻度有关设定

`hide()` Hidethe x- and y-axis tick labels.

`hide_x()` Hide the x-axis tick labels.

`hide_y()` Hide the y-axis tick labels.

`set_font([family,style, variant, stretch, ...])` Setthe font of the tick labels.

`set_style(style)` Set the format of the x-axis ticklabels.

`set_xformat(format)` Set the format of the x-axis tick labels.

`set_xposition(position)` Set the position of the x-axis tick labels('top' or 'bottom')

`set_yformat(format)` Set the format of the y-axis tick labels.

`set_yposition(position)` Set the position of the y-axis tick labels('left' or 'right')

`show()` Show the x- and y-axis tick labels.

`show_x()` Show the x-axis tick labels.

4.3.8 `aplpy.Ticks(parent)` 刻度设定

`hide()` Hidethe x- and y-axis ticks

`hide_x()` Hide the x-axis ticks

`hide_y()` Hide the y-axis ticks

`set_color(color)` Set the color of the ticks

`set_length(length[,minor_factor])` Set the length of theticks (in points)

`set_linewidth(linewidth)` Set the linewidth of the ticks (inpoints)

`set_minor_frequency(frequency)` Set the number of subticks per major tick.

`set_xspacing(spacing)` Set the x-axis tick spacing, in degrees.

`set_yspacing(spacing)` Set the y-axis tick spacing, in degrees.

`show()` Show the x- and y-axis ticks

`show_x()` Show the x-axis ticks

`show_y()` Show the y-axis ticks

---

5. Cosmocalc

5.1 总概述

计算给定宇宙学的有用值。

本模块使用从CC.py（JamesSchombert）改编而来的代码，它是宇宙计算器的Python版本。具体使用请参考官网

该模块包里面包含了众多有关天文的常量和导出量，可以直接使用。因此，只包含有两个实用函数。

在这之前，需要先下载安装包cosmocalc-.tar.gz，然后解压安装到系统中。

导入方法：`from cosmocalc import cosmocalc`

5.2 calculated values

以下导出值可以直接查看和使用

Name	Value	Units
z	红移量输入	/
H0	哈勃常数	/
WR	Omega(辐射)	/
WK	Omega曲线 = 1-Omega(总)	/
WM	Omega物质	/
WV	Omega真空	/
DTT	从红移量z到现在的时间	Gyr
age	宇宙年龄	Gyr
zage	在红移量为Z时的宇宙年龄	Gyr
DCMR	修正径向距离	Gyr Mpc cm
VCM	Comoving volume within redshift	Gpc3
DA	角尺寸距离	Gyr Mpc cm
DL	光度距离	Gyr Mpc cm
PS	每弧秒的距离	kpc cm

5.3 包含的函数

5.3.1 `cosmocalc.cosmocalc`

> 计算所提供的宇宙学常量、导出量

以<名称>、<值>的形式返回。此外，还会完成各种有用的单位转换，并将其作为 _ ： 存储在内存中。 例如：角大小距离：

`DA` 0.38250549415474988,

`DA_Gyr` 5.2678010166833023,

`DA_Mpc` 1615.1022857909447,

`DA_cm'` 4.9836849147807571e+27

5.3.2 `cosmocalc.get_options()`

6. Kapteyn

作者：Hans Terlouw

Kapteyn软件包为软件提供构建模块，专注于使用世界坐标，或使用世界坐标绘制图像数据。

6.1 wcs 模块

wcs模块是与Mark Calabretta的WCSLIB的接口，并且其提供了一套独立的天体转换。 WCSLIB按照惯例"实现了FITS世界坐标系统（WCS）标准，该标准定义**了图像像素坐标**与**世界坐标**的**相互计算**方法。"

- 给定一个带有关于世界坐标系统（WCS）头部信息的FITS头部或Python字典，可以在像素坐标和世界坐标之间进行转换。
- 不同的坐标表示是可能的（标量元组，NumPy数组等）
- 天空和参考系统之间的转换。（支持**任意**时期**赤道和黄道的坐标**、**参考系统FK4，FK4-NO-E，FK5，ICRS**和**动态J2000**以及**银河系和超星系坐标系**。）
- 支持'备用'标题（标题可以有多个WCS描述）
- 支持混合坐标转换（即混合输入的像素坐标和世界坐标）。
- 光谱坐标转换，例如将频率轴转换为光学速度轴。

6.2 Celestial 模块

Celestial模块为程序员提供了一套基本的例程：

- 在实时天体和参考系统之间进行坐标转换。也可通过模块wcs获得
- 时期转换。也可通过模块wcs获得
- 许多实用功能例如转换时代，解析定义sky和参考系统的字符串，计算儒略日，进动角（旋转角）的角度等。
- 其中**Sky**包括sky system（赤道、黄道、银河、超星系），参考系统，春分点和观测历元。它可以是（一个字符串、包含一个或多个元素的元组、一个单一的元素）。

6.3 wcsgrat 模块

- 该模块中的大多数功能都是通过模块maputils中用户友好的方法提供的。
- 根据经度变化计算显示恒定纬度的网格线，反之亦然。
- 设置各种绘图元素（如刻度线，刻度线标签和轴标签）属性的方法。
- 计算图内标签位置的方法（例如，对于所有天空图）。

6.4 maputils 模块

- 易于与Matplotlib结合使用
- 模块wcs，天体，wcsgrat的方法的便利方法
- 不同格线（每个代表不同的天空系统），来自具有两个以上轴的数据集的数据切片图（例如，具有来自无线电干涉仪观察的通道图的FITS文件）
- 具有"光谱平移"的频谱轴的图（例如频率到射频速度）
- 在世界坐标中具有距离的统治者，对投影进行校正。
- 覆盖整个天空的数据图（allsky图）
- 多个图像的马赛克（例如，HI通道图）
- 一个简单的电影循环程序来查看'频道'地图。
- 交互式颜色地图选择和修改。

6.5 positions 模块

将字符串转换为像素和世界坐标中的位置。

6.6 Rulers 模块

用世界坐标中的恒定距离标记一条直线。它的功能在模块maputils中可用

6.7 shapes 模块

高级绘图与用户交互。用户在图像中定义形状（多边形，椭圆，圆形，矩形，样条曲线），并且形状以其他图像传播（以世界坐标）。形状对象跟踪其面积（以像素为单位）以及形状内的像素总和。从这些流量可以计算出来。

6.8 tabarray 模块

磁盘上ASCII文件数据的快速I / O。

6.9 mplutil 模块

用于Matplotlib中事件处理的各种高级实用程序。它的大部分功能都在模块maputils中使用。

6.10 kmpfit 模块

该模块提供Fitter类，该类使用MPFIT中的C实现，Craig Markwardt针对IDL的非线性最小二乘曲线拟合例程。

MPFIT使用Levenberg-Marquardt技术来解决最小二乘问题，这是迭代搜索最佳方法的特殊策略。在其典型应用中，MPFIT将被用于通过调整一组参数来将用户提供的功能（“型号”）提供给用户提供的数据点（“数据”）。 MPFIT基于Moré及其合作者的强大例行MINPACK-1（LMDIF.F）。

> 例如，研究人员可能认为一组观测数据点最好用高斯曲线建模。高斯曲线通过其均值，标准偏差和标准化进行参数化。 MPFIT将在一定的限制范围内找到最适合数据的一组参数。在最小二乘意义上，文件是“最好的”;也就是说，模型和数据之间的加权平方差的总和被最小化。

---

7. Pyflatuion

7.1 总概述

`Pyflation` (Cosmologicalsimulations in Python)是一个用于模拟早期宇宙中的扰动的功能包，可以通过利用一阶和二阶摄动的克莱因-高登方程(Klein-Gordon equation)研究这些摄动的演化和行为。

一旦在pyflation中安装了模块，Python包可以用来模拟早期宇宙的不同标量场的模型。主要的类（ **class** ）包含在cosmomodels模块中，包括背景场和一阶和二阶扰动的模拟。另外，sourceterm软件包包含计算二阶扰动演化所需术语所需的模块。

在此之前，需要先到官网下载并安装安装包

该包还包括3个分包，分别是：

`analysisPackage`

`solutions Package`

`sourceterm Package`

说明书文档参考链接

7.2 所包含的模块

Pyflation包包含6个大模块，分别是：

`cmpotentialsModule`

`cosmographsModule`

`cosmomodelsModule`

`helpers Module`

`rk4 Module`

`sohelpers Module`

> 下面分块讲解这6个模块

7.2.1 `cmpotentials` 模块

> 该模块提供可以与cosmomodels.py一起使用的功能

函数
`pyflation.cmpotentials.bump_nothirdderiv(y,params=None)`
`pyflation.cmpotentials.bump_potential(y,params=None)`
`pyflation.cmpotentials.hilltopaxion(y,params=None)`
`pyflation.cmpotentials.hybrid2and4(y,params=None)`
`pyflation.cmpotentials.hybridquadratic(y,params=None)`
`pyflation.cmpotentials.hybridquartic(y,params=None)`
`pyflation.cmpotentials.inflection(y,params=None)`
`pyflation.cmpotentials.lambdaphi4(y,params=None)`
`pyflation.cmpotentials.linde(y,params=None)`
`pyflation.cmpotentials.msqphisq(y,params=None)`
`pyflation.cmpotentials.msqphisq_withV0(y,params=None)`
`pyflation.cmpotentials.nflation(y,params=None)`
`pyflation.cmpotentials.phi2over3(y,params=None)`
`pyflation.cmpotentials.productexponential(y,params=None)`
`pyflation.cmpotentials.quartictwofield(y,params=None)`
`pyflation.cmpotentials.resonance(y,params=None)`
`pyflation.cmpotentials.ridge_twofield(y,params=None)`
`pyflation.cmpotentials.step_potential(y,params=None)`

7.2.2 cosmographs模块

> **概述**

> **cosmographs.py** - Graphing functions for pyflation package

该模块提供了帮助函数，用于使用Matplotlib软件包绘制pyflation包的模拟结果。特别是对于`multi_format_save`函数，它根据请求将指定图形保存为不同的格式。

函数	功能
`pyflation.cosmographs.LogFormatterTeXExponent(*args, **kwargs)`	扩展标记刻度
`pyflation.cosmographs.calN_figure(ts,ys, fig=None, plot_fn=None, models_legends=None, ylabel=None, size='large',ls=['r-', 'g--', 'b:'], halfticks=False)`	/
`pyflation.cosmographs.generic_figure(xs,ys, fig=None, plot_fn=None, models_legends=None, xlabel=None, ylabel=None,size='large', ls=['r-', 'g--', 'b:'], halfticks=False)`	用标准x轴创建一个通用图形
`pyflation.cosmographs.makeklegend(fig,k)`	将图例附加到指定的图形中
`pyflation.cosmographs.multi_format_save(filenamestub,fig=None, formats=None, overwrite=False, **kwargs)`	一次保存多种格式的图形
`pyflation.cosmographs.reversexaxis(a=None)`	反转x轴方向
`pyflation.cosmographs.save(fname,dir=None, fig=None)`	/
`pyflation.cosmographs.save_with_prompt(fname)`	/
`pyflation.cosmographs.set_size(fig, size='large')`	/
`pyflation.cosmographs.set_size_half(fig)`	/
`pyflation.cosmographs.set_size_large(fig)`	/
`pyflation.cosmographs.set_size_small(fig)`	/

7.2.3 comomodels 模块

> **(1)概述：cosmomodels.py- Cosmological Model simulations**

提供建模宇宙学通胀情景的类，特别重要的类有：

`FOCanonicalTwoStage` - 驱动一阶计算

`SOCanonicalThreeStage` - 驱动二阶计算

`CanonicalFirstOrder` - 包含一阶计算的衍生物的类

`CanonicalRampedSecondOrder` - 包含二阶计算的导数和增强源项的类。

> **(2) 主要的类和函数**

class

`pyflation.cosmomodels.BasicBgModel`

class
`pyflation.cosmomodels.CanonicalBackground`

class
`pyflation.cosmomodels.CanonicalFirstOrder`

> 一阶模型使用efold作为具有多个字段的时间变量

class
`pyflation.cosmomodels.CanonicalHomogeneousSecondOrder`

class
`pyflation.cosmomodels.CanonicalRampedSecondOrder`

class
`pyflation.cosmomodels.CanonicalSecondOrder`

class
`pyflation.cosmomodels.CombinedCanonicalFromFile`

class
`pyflation.cosmomodels.CosmologicalModel`

> `createhdf5structure(filename,grpname='results', yresultshape=None, hdf5complevel=2, hdf5complib='blosc')`
>
> `derivs(yarray, t)`
>
> `findH(potential, y)`
>
> `gethf5paramsdict()`
>
> `potentials(y, pot_params=None)`
>
> `run(saveresults=True)`
>
> `saveallresults(filename=None, filetype='hf5',yresultshape=None, **kwargs)`
>
> `saveresultsinhdf5(rf, grpname='results')`

class
`pyflation.cosmomodels.FOCanonicalTwoStage`

class
`pyflation.cosmomodels.FONoPhase`

class
`pyflation.cosmomodels.FOSuppressOneField`

class
`pyflation.cosmomodels.FixedainitTwoStage`

exception `pyflation.cosmomodels.ModelError`

class
`pyflation.cosmomodels.MultiStageDriver`

> findHorizoncrossings(factor=1)
>
> find_efolds_after_inflation(Hend,Hreh=None)

class `pyflation.cosmomodels.PhiModels`

class
`pyflation.cosmomodels.SOCanonicalThreeStage`

class
`pyflation.cosmomodels.SOHorizonStart`

class
`pyflation.cosmomodels.TestModel`

> `pyflation.cosmomodels.make_wrapper_model(modelfile,*args, **kwargs)`

> `pyflation.cosmomodels.profile(f)`

7.2.4 helpers模块

> **(1) 概述**
>
> **helpers.py**- Helper functions 提供辅助函数供其他地方的软件包使用。

> **(2) 主要的函数**

`pyflation.helpers.cartesian(arrays,out=None)`

`pyflation.helpers.cartesian_product(lists,previous_elements=[])`

`pyflation.helpers.ensurepath(path)`

`pyflation.helpers.eto10(number)`

`pyflation.helpers.find_nearest(array, value)`

`pyflation.helpers.find_nearest_ix(array, value)`

`pyflation.helpers.getintfunc(x)`

`pyflation.helpers.getkend(kinit, deltak, numsoks)`

`pyflation.helpers.invmpc2mpl(x=1)`

`pyflation.helpers.ispower2(n)`

`pyflation.helpers.klegend(ks, mpc=False)`

`pyflation.helpers.mpl2invmpc(x=1)`

`pyflation.helpers.nanfillstart(a, l)`

`pyflation.helpers.removedups(l)`

`pyflation.helpers.seq(min=0.0, max=None, inc=1.0,type=<type 'float'>, return_type='NumPyArray')`

`pyflation.helpers.startlogging(log, logfile, loglevel=20, consolelevel=None)`

7.2.5 rk4模块

> **(1) 概述**

rk4.py- Runge-Kutta（龙格-库塔算法）ODE solver，提供基于Runge-Kutta的ODE求解器，用于pyflation模型。

> **(2) 主要包含的函数**

`exceptionpyflation.rk4.SimRunError`

`pyflation.rk4.rk4stepks(x,y, h, dydx, dargs, derivs)`

> 做一步经典的四阶Runge Kutta方法，从时间步长为h的x开始，由deriv给出的导数

`pyflation.rk4.rkdriver_tsix(ystart,simtstart, tsix, tend, allks, h, derivs)`

> 经典Runge Kutta四阶法的驱动函数。

7.2.6 sohelpers模块

（1）概述

sohelpers.py- Second order helper functions 为来自cosmomodels.py的二阶数据提供辅助函数。

（2）主要的函数

`pyflation.sohelpers.combine_source_and_fofile(sourcefile,fofile, newfile=None)`

> 复制源代码

8. Galpy

8.1 总概述

galpy是一个用于银河动力学的Python包。它支持各种potentials的轨道积分、评估和采样各种分布函数，以及计算所有静态potential的角坐标。galpy是一个使用astropy的附属软件包，可以为变量提供对astropy 数量框架的全面支持。

galpy最基本的特点是能够显示旋转曲线并对任意组合的potentials进行轨道积分。其中，最基本的功能包括在这两个分包里：`galpy.potential` 和 `galpy.orbit` 。

在此之前，你需要先安装Galpy包，可通过如下命令直接安装：

```
condainstall galpy -c conda-forge
#或
pip install galpy
```

8.2 主要功能

该包主要包括5个大功能以及众多小功能

8.2.1 Potentialsin galpy

```
import galpy.potential
```

Funtion	用途
Potentials andforces	势能和强迫
Densities	密度
Modifyingpotential instances using wrappers（NEW in v1.3）	/
Close-to-circularorbits and orbital frequencies	轨道频率/（接近）正圆轨道
Usinginterpolations of potentials	势能插值
Initializingpotentials with parameters with units	初始化位势能
Generaldensity/potential pairs with basis-function expansions（UPDATED inv1.3）	基函数扩展的一般密度/潜在配对
The potential of N-body simulations	N体模拟
Conversion toNEMO potentials	NEMO转化
Addingpotentials to the galpy framework	/
Adding wrapperpotentials to the galpy framework（NEW in v1.3）	/

8.2.2 Two-dimensional disk distribution functions 二维圆盘分布函数

> 这对于研究恒星相对靠近星系中平面的动力学非常有用

```
from galpy.df import dehnendf
from galpy.df import shudf
from galpy.df import schwarzschilddf
```

功能	用途
Types of diskdistribution functions	圆盘分布类型
Evaluatingmoments of the DF	DF时刻
Using correcteddisk distribution functions	/
Oort constantsand functions	Oort常数和函数
Sampling datafrom the DF	从DF采样数据
Non-axisymmetric,time-dependent disk distribution functions	/

8.2.3 A closer look at orbit integration 近距离轨道整合

```
from galpy.orbit import Orbit
```

功能	用途
Orbitinitialization	轨道初始化轨道初始化
Orbitintegration	轨道整合
Displaying theorbit	显示轨道
Animating theorbit（NEW in v1.3）	动态轨道
Orbitcharacterization	轨道特征
Fast orbitcharacterization（NEW in v1.3）	快速定位轨道
Accessing theraw orbit	进入原始轨道
Fast orbitintegration	快速轨道整合
Integration ofthe phase-space volume	/

8.2.4 Action-angle coordinates 动作-角坐标系

> `galpy.actionAngle`

功能	用途
Action-anglecoordinates for the isochrone potential	等时线位势
Action-anglecoordinates for spherical potentials	球形电位
Action-anglecoordinates using the adiabatic approximation	绝热近似
Action-anglecoordinates using the Staeckel approximation	Staeckel近似
Action-anglecoordinates using an orbit-integration-based approximation	轨道积分近似
Action-anglecoordinates using the TorusMapper code	TorusMapper代码

8.2.5 Three-dimensional disk distribution functions 三维圆盘分布功能

> 需要导入

```
from galpy.potential import MWPotential2014
from galpy.actionAngle import actionAngleAdiabatic
from galpy.df import quasiisothermaldf
```

功能	用途
Setting up the DF and basic properties	设置DF和基本属性
Evaluatingmoments	测试时刻
Evaluating andsampling the full probability distribution function	测试和臭氧概率分布函数

8.3 包含的函数

由于包含的函数非常非常多，在此不一一列出，需要的请自行前往：http://galpy.readthedocs.io/en/latest/genindex.html 查看函数使用说明。

9.alipy

9.1 总概述

Alipy包是一个可以快速、自动识别光学天文图像之间几何变换的python软件包。

一共有6个模块，分别是：

align Module

ident Module

imgcat Module

pysex Module

quad Module

star Module

具体的使用说明书请参考官网

9.2 模块分块简介

> 具体的函数说明请查看这里

9.2.1 align 模块

（1）概述：主要用于图像对齐

（2）所包含的函数

`alipy.align.affineremap(filepath,transform, shape, alifilepath=None, outdir='alipy_out', makepng=False, hdu=0,verbose=True)`

> 对图像应用简单的仿射变换，并将结果保存为FITS.

`alipy.align.irafalign(filepath,uknstarlist, refstarlist, shape, alifilepath=None, outdir='alipy_out',makepng=False, hdu=0, verbose=True)`:

> 使用iraf geomap和gregister对齐图像

`alipy.align.shape(filepath,hdu=0, verbose=True)`：

> 返回二维图像（FITS）

`alipy.align.tofits(outfilename,pixelarray, hdr=None, verbose=True)`:

> 采用二维 numpyjuzh矩阵并将其写入FITS文件。

9.2.2 ident 模块

（1）概述：主要用于图像识别

（2）所包含的类和函数

class `alipy.ident.Identification(ref, ukn)`

> 表示两个ImgCat对象之间的转换标识
>
> `calcfluxratio(verbose=True)`
>
> `findtrans(r=5.0,verbose=True)`
>
> `showmatch(show=False,verbose=True)`
>
> `alipy.ident.run(ref, ukns, hdu=0,visu=True, skipsaturated=False, r=5.0, n=500, sexkeepcat=False, sexrerun=True,verbose=True)`:——用于识别图像间的顶层函数。返回包含所有信息的标识对象的列表，以便进一步查找

9.2.3 imgcat 模块

（1）概述：顾名思义，即有关图像的绘制

（2）主要包含的类和函数

class `alipy.imgcat.ImgCat(filepath, hdu=0,cat=None)`

> 代表一个单独的图像及其相关的目录和星表等
>
> `makecat(rerun=True,keepcat=False, verbose=True)`
>
> `makemorequads(verbose=True)`
>
> `makestarlist(skipsaturated=False,n=200, verbose=True)`
>
> `showquads(show=False,flux=True, verbose=True)`
>
> `showstars(verbose=True)`
>
> `alipy.imgcat.ccworder(a)`——排列坐标数组CCW以绘制多边形

9.2.4 pysex模块

（1）概述:

Small module wrapping sextractor. The idea is to have a singlefunction taking an image and returning a sextractor catalog.

（2）所包含的函数

`alipy.pysex.run(image='',imageref='', params=[], conf_file=None, conf_args={}, keepcat=True,rerun=False, catdir=None)`:

> 使用给定参数在给定图像上运行sextractor，返回包含sextractor输出的asciidata目录。

9.2.5 quad模块

（1）概述

Quads are asterisms of 4 stars, used to match the catalogs（用于恒星间的匹配）

（2）所包含的函数

class `alipy.quad.Quad(fourstars)`

> `alipy.quad.makequads1(starlist, n=7, s=0,d=50.0, verbose=True)`——制作出最亮的恒星组合
>
> `alipy.quad.makequads2(starlist, f=5.0, n=6,s=0, d=50.0, verbose=True)`—和上面一个函数类似，只是s允许跳过每个区域最亮的恒星
>
> `alipy.quad.mindist(fourstars)`
>
> `alipy.quad.proposecands(uknquadlist,refquadlist, n=5, verbose=True)`
>
> `alipy.quad.quadtrans(uknquad, refquad)`
>
> `alipy.quad.removeduplicates(quadlist,verbose=True)`

9.2.6 star 模块

（1）概述：用于修改cosmouline的恒星模块，这个模块包含几何匹配算法。

（2）所包含的类和函数

class `alipy.star.SimpleTransform(v=(1, 0, 0,0))`

> 用于由旋转、各向同性缩放和平移组成的仿射变换
>
> `apply((x,y))`
>
> `applystar(star)`
>
> `applystarlist(starlist)`
>
> `getrotation()`
>
> `getscaling()`
>
> `inverse()`
>
> `matrixform()`

class `alipy.star.Star(x=0.0, y=0.0,name='untitled', flux=-1.0, props={}, fwhm=-1.0, elon=-1.0)`

> 简单分类恒星——Simple class to represent a single source (usually stars, butnot necessarily)
>
> `coords(full=False)`
>
> `copy()`
>
> `distance(otherstar)`
>
> `distanceandsort(otherstarlist)`
>
> `trigangle(otherstar)`
>
> `alipy.star.area(starlist, border=0.01)`——返回恒星覆盖的区域
>
> `alipy.star.findstar(starlist, nametofind)`
>
> `alipy.star.fitstars(uknstars, refstars,verbose=True)`
>
> `alipy.star.identify(uknstars, refstars,trans=None, r=5.0, verbose=True, getstars=False)`

> `alipy.star.listtoarray(starlist, full=False)` —— 将starlist转换为2D阵列以进行快速操作
>
> `alipy.star.printlist(starlist)`
>
> `alipy.star.readmancat(mancatfilepath, verbose='True')` ——数据以恒星对象列表的形式返回
>
> `alipy.star.readsexcat(sexcat, hdu=0,verbose=True, maxflag=3, posflux=True, minfwhm=2.0, propfields=[])`
>
> `alipy.star.sortstarlistby(starlist, measure)`
>
> `alipy.star.sortstarlistbyflux(starlist)`

10.Pyregion

pyregion是一个解析ds9区域文件的python模块。 它也支持ciao区域文件。该软件包的主要重点是读取由ds9本身生成的区域文件。 它读取由ds9创建的大部分区域文件。 但是，即使ds9可以成功读取某些用户创建的（或由其他程序创建的）区域文件，也可能无法读取它们。 标尺，指南针和投影类型被忽略。

10.1 Read Region Files

`pyregion.open`将区域名称作为参数并返回一个`ShapeList`对象，该对象基本上是一个`Shape`对象列表（ShapeList 是Python内置列表类的子类）。

例子

```
import pyregion

region_string = """
# Region file format: DS9 version 4.1
# Filename: test01.fits
global color=green dashlist=8 3 width=1 font="helvetica 10 normal" select=1 highlite=1
dash=0 fixed=0 edit=1 move=1 delete=1 include=1 source=1
fk5
circle(11:24:24.230,-59:15:02.20,18.5108") # color=cyan background
box(11:24:39.213,-59:16:53.91,42.804",23.616",19.0384) # width=4
"""
r = pyregion.parse(region_string)
```

- `name` : 形状(shape)的名称。例如 circle, box等等

  ```
  >>> print r[0].name
  circle
  ```

- `coord_format` : 坐标格式。例如“fk5”, “image”, “physical”等等

  ```
  >>> print r[0].coord_format
  fk5
  ```

- `coord_list` : `coord_format`中的坐标列表。

```
>>> print r[0].coord_list
[171.10095833333332, -59.250611111111112, 0.0051418888888888886]
```

- `comment`：与形状(shape)关联的注释字符串（可以是None）

```
>>> print r[0].comment
color=cyan background
```

- `attr`：形状(shape)的属性。

```
>>> print r[0].attr[0]
['background']
>>> print r[0].attr[1]
{'color': 'cyan',
 'dash': '0 ',
 'dashlist': '8 3 ',
 'delete': '1 ',
 'edit': '1 ',
 'fixed': '0 ',
 'font': '"helvetica 10 normal"',
 'highlite': '1 ',
 'include': '1 ',
 'move': '1 ',
 'select': '1 ',
 'source': '1',
 'width': '1 '}
```

10.2 Draw Regions with Matplotlib

pyregion可以使用matplotlib绘制ds9区域。`ShapeList.get_mpl_patches_texts`返回一列`matplotlib.artist.Artist`对象

```
r2 = pyregion.parse(region_string).as_imagecoord(f[0].header)
patch_list, artist_list = r2.get_mpl_patches_texts()
```

第一项是`matplotlib.patches.Patch`的列表，第二项是其他类型的艺术家（通常是文本）。

```
# ax is a mpl Axes object
for p in patch_list:
    ax.add_patch(p)
for t in artist_list:
    ax.add_artist(t)
```

10.3 Functions

get_mask(region, hdu[, origin])	获取掩码
open(fname)	读取和解析DS9区域文件
parse(region_string)	将DS9区域字符串解析为ShapeList。
test([package, test_path, args, plugins, ...])	使用py.test运行测试

10.4 Classes

Shape(shape_name, shape_params)	形状
ShapeList(shape_list[, comment_list])	Shape对象列表

11. ChiantiPy

11.1 概述

`ChiantiPy`是一个纯粹的Python软件包，用于使用CHIANTI原子数据库执行天体物理光谱的计算。ChiantiPy的0.6.4版本与CHIANTI数据库版本8.0兼容。

CHIANTI由一个原子数据库组成，可以用来解释从高温，光学薄天体物理源发出的谱线和连续谱。CHIANTI项目提供了一套使用交互式数据语言（IDL）编写的例程，用于访问数据库并计算观察到的光谱或合成光谱的各种用量。

11.2 包含模块

11.3 模块具体功能

11.3.1 Level populations

产生一个matplotlib绘图窗口，将前10位（默认）水平的数量绘制为温度的函数。

如果尚未计算级别数量，`popPlot()`将调用`populate()`方法来计算级别数量并将它们存储在Population字典中，其中键为['protonDensity', 'population', 'temperature', 'density']。

> protonDensity： 质子密度
>
> population：人口
>
> temperature：温度
>
> density：密度

11.3.2 A ChiantiPy Convention

函数名称	函数功能
`fe14.populate()`	创建fe14.Population，其中包含级别的人口信息
`fe14.emiss()`	创建包含线发射率信息的fe14.Emiss
`fe14.intensity()`	创建的fe14。强度包含线强度信息
`fe14.spectrum()`	创建fe14。频谱包含线和连续光谱信息

11.3.3 Spectral Line Intensities

将绘制指定波长范围内顶部（默认= 10）线的强度。 如果Intensity属性尚未计算，它将计算它。由于涉及21个温度，所以选择单个温度（21/2 = 10）。 否则，绘制温度= t [2] = 7.9e + 5的强度，在这种情况下。 并且，通过指定相对= 1，发射率将被绘制为相对于最强线。

11.3.4 G(n,T) function

提出了一个`matplotlib`绘图窗口，其中显示了从210到220埃波长范围内顶部（最强）3条线的发射率。

11.3.5 Intensity Ratios

绘制显示Fe XIV线相对放射率的图

11.3.6 Spectra of a single ion

计算指定波长范围内的光谱，并可自行设置“分辨率“，使用该“分辨率”的默认滤波器（`filters.gaussianR`）对其进行滤波。

11.3.7 Free-free and free-bound continuum

模块提供了计算大量单个离子的自由自由和自由束缚谱的能力。

11.3.8 The multi-ion class Bunch

多离子类束[v0.6新增]继承了ion类继承的许多相同的方法，例如intensityList，intensityRatio和intensityRatioSave。 Widing和Feldman（1989，ApJ，344,1046）作为其有用性的简短证明，使用Mg VI和Ne VI的线比作为太阳大气中元素丰度变化的诊断。 为了准确，两个离子的谱线必须具有相同的温度响应。

11.3.9 Spectra of multiple ions and continuum

可以计算CHIANTI数据库中所有离子的谱图。

也可以计算CHIANTI数据库中所有离子的选择谱。

有3个光谱类别。

> spectrum - 可以在任何地方使用的单处理器实现
>
> mspectrum - 使用Python多处理类，不能在IPythonqtconsole或笔记本中使用
>
> ipymspectrum [v0.6中的新功能] - 使用IPython并行类，可用于IPython qtconsole或笔记本

11.3.10 Radiative loss rate

综合光谱是通过对所有温度的光谱进行求和而形成的。

> 对于minAbund = 1.e-6，计算在3.5 GHz处理器上需要209 s。
>
> 对于minAbund = 1.e-5，计算在3.5 GHz处理器上需要122 s。

该过滤器不适用于连续体。 使用Spectrum模块进行计算可能非常耗时。

一种控制计算时间长度的方法是使用ionList关键字限制离子数，并通过将doContinuum关键字设置为0或False来避免连续计算。

12.PyEphem

12.1 概述

> **作者： Elwood Downey**

PyEphem是Python下用来进行天文历法计算的一个程序包，由国外大学天文爱好者编写的。 因其使用VOS87行星运动数据，计算精度达到了很高的精度，足以满足一般的观测需要。

考虑到地球表面的日期和位置，它可以计算太阳和月球，行星及其卫星，以及用户可以提供轨道元素的任何小行星，彗星或地球卫星的位置。 附加功能用于计算天空中两个物体之间的角度分离，确定物体所处的星座，以及查找物体在特定日期上升，过渡和设置的时间。

12.2 模块具体功能

12.2.1 Bodies

1. 恒星(Sun)，行星(Moon, Mars etc.)，彗星，小行星与恒星等各自拥有自己的类。如：

```
>>> m = ephem.Mars()
>>> m.name
'Mars'
```

2. PyEphem包含一个简单的著名星体的适度目录。如：

```
>>> a = ephem.star('Arcturus')
>>> a.name
'Arcturus'
```

3. Body实例知道他们 *name* （你可以设置任何你想要的）。

```
>>> m = ephem.Mars('2003/8/27')
>>> print('%s %s %.10f' % (m.name, m.elong, m.size))
Mars -173:00:34.2 25.1121063232
```

4. 用于创建Body时的额外参数用于执行初始操作 *compute()*

12.2.1.1body.compute(date) (以星体为中心点)

输出指定 *body* (星体)在指定时期的**参心坐标系位置、地心坐标系位置、 距角、星等、角速度、线速度**、其在该时段处于在**地平线之上(下)**。

> 每个`body`，无论是行星，彗星，小行星还是恒星，都会在您要求它计算其坐标时返回三组坐标位置(**标准升落**(Astrometric geocentric)))。
>
> `a_ra`, `a_dec` — **Astrometric Geocentric Position**for the star atlas `epoch` you've specified
>
> `g_ra`, `g_dec` — **Apparent Geocentric Position**for the epoch-of-date
>
> `ra`, `dec` — **Apparent Topocentric Position**for the epoch-of-date
>
> For Earth satellites that are given an `Observer` to compute,there is an important difference:
>
> `a_ra`, `a_dec` — **Astrometric Topocentric Position**for the `epoch` of your Observer
>
> `g_ra`, `g_dec` — (same as above)
>
> `ra`, `dec` — (same as above)
>
> `elong` — Elongation (angle to sun)
>
> `mag` — Magnitude
>
> `size` — Size (diameter in arcseconds)
>
> `radius` — Size (radius as an angle)
>
> `circumpolar` — whether it stays above the horizon
>
> `neverup` — whether is stays below the horizon

Body为太阳系(Solar System)时

> `hlon` 黄经
>
> `hlat` 黄纬
>
> `sun_distance` 太阳距离(AU)
>
> `earth_distance` 地球距离(AU)
>
> `phase` 相位

body为作为行星的月球(planetary moons)时

> 月球相对于地球的位置(以行星半径测量)
>
> `x` - 偏移量 +E 或 -W
>
> `y` - 偏移量 +N 或 -N
>
> `z` - 偏移 +F 或 -B
>
> 月亮是否可见:
>
> `earth_visible` — 地球视角
>
> `sun_visible` — 太阳视角

body为人造卫星(artificial satellites)时

> 地理坐标系下的卫星位置:
>
> `sublat` — 纬度 (+N)
>
> `sublong` — 精度 (+E)

> `elevation` — 海拔高度 (m)
>
> `range` — 从观察者位置到卫星的距离 (m)
>
> `range_velocity` — 变化率区间(m/s)
>
> `eclipsed` — 卫星是否在地球的阴影中

body 为月球时

> 实时振动(libration):
>
> `libration_lat` — 纬度上
>
> `libration_long` — 精度上
>
> `colong` — Selenographic colongiude(考特妮陨石坑)
>
> `moon_phase` — 月面照亮百分比
>
> `subsolar_lat` — 月球的太阳直射纬度

body 为木星(Jupiter)时

> `cmlI` — 在Jovial赤道上自转的中心经线经度
>
> `cmlII` — 在温带纬度上测量自转的中心子午线经度

body为土星(Saturn)时

> `earth_tilt` — 倾向地球
>
> `sun_tilt` — 倾向太阳

12.2.1.2 body.compute(observer) (以观察者为中心点)

```
>>> gatech = ephem.Observer()
>>> gatech.lon = '-84.39733'
>>> gatech.lat = '33.775867'
>>> gatech.elevation = 320
>>> gatech.date = '1984/5/30 16:22:56'
>>> v = ephem.Venus(gatech)
>>> print('%s %s' % (v.alt, v.az))
72:19:44.8 134:14:25.3
```

使用date观察者。

使用epoch观察者。

设置所有的Body前一节中列出的属性（但ra并dec会得到不同的值，见下文）。

还计算观察者身体出现在天空（或地平线下）的位置，并设置另外四个Body属性：

Apparent topocentric position

`ra` — Right ascension

`dec` — Declination

Apparent position relative to horizon

`az` — Azimuth east of north

`alt` — Altitude above horizon

这些视位置包括调整以模拟大气折射为观察者的temp和presure; 将观察者设置pressure为零以忽略折射。

对于地球卫星天体而言，天体坐标不是地心坐标a_ra，a_dec而是地心坐标，而不是地心坐标，因为找出卫星出现在J2000（或使用的任何时代）星图上的位置对于位于地球。

12.2.1.3 catalog format

```
>>> line = "C/2002 Y1 (Juels-
Holvorcem),e,103.7816,166.2194,128.8232,242.5695,0.0002609,0.99705756,0.0000,04/13.2508/200
3,2000,g  6.5,4.0"
>>> yh = ephem.readdb(line)
>>> yh.compute('2007/10/1')
>>> print('%.10f' % yh.earth_distance)
14.8046731949
>>> print(yh.mag)
23.96
```

- Bodies可以用XEphem格式导入和导出。
- 当处理小行星和彗星时，其轨道参数会经常被修改，你通常会发现你下载了一个XEphem文件并阅读它的内容。
- 要解释XEphem格式的一行，请调用 `readdb()` 函数：
- 要以XEphem格式导出body，请调用 `writedb()` 主体 `body` 的方法：
- 卫星元素通常以称为TLE的格式打包，卫星名称在一行上，元素在下面两行上。
- 调用 `readtle()` 函数将TLE条目转换为PyEphem `Body` 。

12.2.1.4 bodies with orbital elements

- 加载像**慧星**和**小行星**这样的小对象时，生成的对象指定允许XEphem预测其位置的轨道元素。
- 轨道元素可供您检查和更改。
- 如果缺少要从中加载对象的目录，则可以先创建以下类型之一的原始主体并填充其元素。
- 元素属性名称以下划线开头，以区分它们与 `Body` 作为调用结果设置的常规属性 `compute()` 。
- 每个FixedBody只有三个必要的元素：

  `_ra` , `_dec` — 位置

  `_epoch` — 该位置的时段

  `FixedBody` 的其他属性：

  `_class` — 类别

  `_spect` — 频谱代码

  `_ratio` — 大直径和较小直径之间的比率

  `_pa` — 北纬（°）以东测量的长轴位于天空中的角度
- `EllipticalBody` 成分：

  `_inc` — 倾角 (°)

  `_Om` — 升交点的经度 (°)

  `_om` — 升交点的纬度 (°)

`_a` — 与太阳的平均距离 (AU)

`_M` — 近日点进动 (°)

`_epoch_M` — 测量日期 `_M`

`_size` — 角度大小(1 AU时的角秒数)

`_e` — 离心率

`_epoch` — Epoch for `_inc`, `_Om`, and `_om`

`_H`, `_G` — Parameters for the H/G magnitude model

`_g`, `_k` — Parameters for the g/k magnitude model

12.2.1.5 other functions

- `ephem.constellation()` 函数返回一个元组(tuple)，其中包含的参数为其在指定时间所在星座的缩写名称和全名。

  你可以传递一个被计算的 `body` 的位置，或者一个元组 `(ra, dec)` 坐标 - 在这种情况下假定时间2000，除非你还传递 `epoch =` 关键字参数指定另一个值。

- `ephem.delta_t()` 函数返回地面时间和世界时间之间给定日期的差异（以秒为单位）。 提出 `Date` 或 `Observer` 的论点。

  没有参数，使用 `ephem.now()` 。

```
>>> ra, dec = '7:16:00', '-6:17:00'
>>> print(ephem.uranometria(ra, dec))
V2 - P274
>>> print(ephem.uranometria2000(ra, dec))
V2 - P135
>>> print(ephem.millennium_atlas(ra, dec))
V1 - P273
```

- `ephem.uranometria` 以ra和dec作为参数,返回给定星图上坐标所在的体积和页面。

  Johannes Bayer的Uranometria。 Wil Tirion编辑的Uranometria 2000.0。 由Roger W. Sinnott和Michael A. Perryman创作的Millennium Star Atlas。

12.2.2 Coordinate Conversion

```
>>> np = Equatorial('0', '90', epoch='2000')
>>> g = Galactic(np)
>>> print('%s %s' % (g.lon, g.lat))
122:55:54.9 27:07:41.7
```

- 有三个坐标类，每个坐标类有三个属性：

1. 赤道

   `ra` — right ascension

   `dec` — declination

   `epoch` — epoch of the coordinate

2. 黄道

   `lon` — ecliptic longitude (+E)

   `lat` — ecliptic latitude (+N)

   `epoch` — epoch of the coordinate

3. 银道

   `lon` — galactic longitude (+E)

   `lat` — galactic latitude (+N)

   `epoch` — epoch of the coordinate

- 创建新坐标时，您可以传递主星体或其他坐标或一对原始角度（始终首先放置经度或右上角）。
- 在创建坐标时，您可以选择传递指定坐标系历元的 `epoch =` 关键字。 否则，将从正在使用的主体或其他坐标中复制历元，或者将J2000用作默认值。
- 有关更多详细信息，请参阅坐标转换文档。

12.2.3 Observers

```
>>> lowell = ephem.Observer()    # 描述地球表面的一个位置。
>>> lowell.lon = '-111:32.1'     # Longitude (+E)
>>> lowell.lat = '35:05.8'       # Latitude (+N)
>>> lowell.elevation = 2198      # Elevation (m)
>>> lowell.date = '1986/3/13'    # Date and time(默认为now)
>>> j = ephem.Jupiter()
>>> j.compute(lowell)
>>> print(j.circumpolar)
False
>>> print(j.neverup)
False
>>> print('%s %s' % (j.alt, j.az))
0:57:44.7 256:41:01.3
```

> `ephem.Observer()` 的其余参数:
>
> - `epoch` — Epoch for astrometric RA/dec (默认公元2000)
> - `temp` — Temperature (°C) (默认25°C)
> - `pressure` — Atmospheric pressure (mBar) (默认为1010mBar。
> - 其余属性默认为0

```
# 使用国际标准大气计算观察者当前高程处的压力。
>>> lowell.compute_pressure()
>>> lowell.pressure
775.6025138640499
```

```
>>> boston = ephem.city('Boston') #只设置纬度，经度和海拔。
>>> print('%s %s' % (boston.lat, boston.lon))
42:21:30.4 -71:03:35.2
```

- XEphem包含一个世界城市的小型数据库。

- 每次使用`ephem.city()`都会返回一个新的`Observer`。

```
# XEphem还可以执行Google地理编码查询
>>> from ephem import cities
>>> ven = cities.lookup('Ven, Sweden')
```

12.2.3.1 transit, rising, setting（上升、经过、设置）

```
>>> sitka = ephem.Observer()
>>> sitka.date = '1999/6/27'
>>> sitka.lat = '57:10'
>>> sitka.lon = '-135:15'
>>> m = ephem.Mars()
>>> print(sitka.next_transit(m))
1999/6/27 04:22:45
>>> print('%s %s' % (m.alt, m.az))
21:18:33.6 180:00:00.0
>>> print(sitka.next_rising(m, start='1999/6/28'))
1999/6/28 23:28:25
>>> print('%s %s' % (m.alt, m.az))
-0:00:05.8 111:10:41.6
```

- `ephem.Observer`从8个方面观察

  > `previous_transit()`
  > `next_transit()`
  > `previous_antitransit()`
  > `next_antitransit()`
  > `previous_rising()`
  > `next_rising()`
  > `previous_setting()`
  > `next_setting()`

12.2.3.2 observer.horizon

```
>>> sun = ephem.Sun()
>>> greenwich = ephem.Observer()
>>> greenwich.lat = '51:28:38'
>>> print(greenwich.horizon)
0:00:00.0
>>> greenwich.date = '2007/10/1'
>>> r1 = greenwich.next_rising(sun)
>>> greenwich.pressure = 0
>>> greenwich.horizon = '-0:34'
>>> greenwich.date = '2007/10/1'
>>> r2 = greenwich.next_rising(sun)
>>> print('Visual sunrise: %s' % r1)

Visual sunrise: 2007/10/1 05:59:30
```

```
>>> print('Naval Observatory sunrise: %s' % r2)
Naval Observatory sunrise: 2007/10/1 05:59:50
```

`ephem.Observer().horizon`(地平线)性定义了你的视野，默认为零度

- 要确定`body`何时会在雾霾或障碍物上方“足够高”，请将视界设置为正数度。
- 当观察者身高高于地面时，可以使用地平线的负值。

12.2.3.3 Other observer methods

- `ephem.city().sidereal_time()`: 无需参数调用, 返回观察者情况下的恒星时间。

```
>>> madrid = ephem.city('Madrid')
>>> madrid.date = '1978/10/3 11:32'
>>> print(madrid.sidereal_time())
12:04:28.09
```

- `ephem.city().radec_of(az, alt)`: 返回天空中给定点后面的明显的地心坐标。

```
>>> ra, dec = madrid.radec_of(0, '90')
>>> print('%s %s' % (ra, dec))
12:05:35.12 40:17:49.8
```

12.2.5 Equinoxes & Solstices

```
>>> d1 = ephem.next_equinox('2000')
>>> print(d1)
2000/3/20 07:35:17
>>> d2 = ephem.next_solstice(d1)
>>> print(d2)
2000/6/21 01:47:51
>>> t = d2 - d1
>>> print("Spring lasted %.1f days" % t)
Spring lasted 92.8 days
```

> 函数采用`Date`参数。
>
> 返回`Date`。

可用功能:

```
previous_solstice()
next_solstice()

previous_equinox()
next_equinox()

previous_vernal_equinox()
next_vernal_equinox()
```

12.2.6 Phases of the Moon

```
>>> d1 = ephem.next_full_moon('1984')
>>> print(d1)
1984/1/18 14:05:10
>>> d2 = ephem.next_new_moon(d1)
>>> print(d2)
1984/2/1 23:46:25
```

> 函数采用 `Date` 参数。
>
> 返回 `Date` 。

可用功能：

```
previous_new_moon()
next_new_moon()

previous_first_quarter_moon()
next_first_quarter_moon()

previous_full_moon()
next_full_moon()

previous_last_quarter_moon()
next_last_quarter_moon()
```

12.2.7 Angles

```
>>> a = ephem.degrees('180:00:00')
>>> print(a) # 大多数 Angle 都是以度数衡量的。
180:00:00.0
>>> a
3.141592653589793 # 每个角度就像一个Python的浮点型。
>>> print("180° is %f radians" % a)
180° is 3.141593 radians
>>> h = ephem.hours('1:00:00')
>>> deg = ephem.degrees(h)
>>> print("1h right ascension = %s°" % deg)
1h right ascension = 15:00:00.0°
```

- 许多 `Body` 和 `Observer` 属性将其值作为角度 `Angle` 返回。
- 只有赤经是以小时计算的。
- 可以通过两个 `ephem` 函数自己创建角度：
  - `degrees()` — return an `Angle` in degrees
  - `hours()` — return an `Angle` in hours

    几种不同格式

```
ephem.degrees(ephem.pi / 32)
ephem.degrees('5.625')
ephem.degrees('5:37.5')
ephem.degrees('5:37:30')
ephem.degrees('5:37:30.0')
ephem.hours('0.375')
ephem.hours('0:22.5')
ephem.hours('0:22:30')
ephem.hours('0:22:30.0')
```

- 在角度上进行数学计算时，结果往往会超过角度的正常范围。 因此为每个角度提供了两个属性：

  `norm` — returns angle normalized to [0, 2π)

  `znorm` — returns angle normalized to [-π, π)

12.2.8 Dates

```
>>> d = ephem.Date('1997/3/9 5:13') # 返回float值
>>> print(d)
1997/3/9 05:13:00   # 只有打印时，传递给str()或使用'%s'格式化时，日期才会表示为日历日期和时间的字符串。日期始终使用世界时间，而不是当地时区。
>>> d
35496.717361111114
>>> d.triple() # 调用.triple () 将日期分为年，月和日。
(1997, 3, 9.21736111111386)
>>> d.tuple() # 调用.tuple () 将日期分为年，月，日，小时，分钟和秒。
(1997, 3, 9, 5, 13, 2.3748725652694702e-07)
>>> d + ephem.hour
35496.75902777778
>>> print(ephem.date(d + ephem.hour))
1997/3/9 06:13:00
>>> print(ephem.date(d + 1))
1997/3/10 05:13:00
```

现代公历日历用于最近的日期，以及1582年10月15日之前的旧儒略日历。

`Date` 的格式

```
ephem.Date(35497.7197916667)
ephem.Date('1997/3/10.2197916667')
ephem.Date('1997/3/10 05.275')
ephem.Date('1997/3/10 05:16.5')
ephem.Date('1997/3/10 05:16:30')
ephem.Date('1997/3/10 05:16:30.0')
ephem.Date((1997, 3, 10.2197916667))
ephem.Date((1997, 3, 10, 5, 16, 30.0))
```

日期存储自1899年的最后一天中午世界时间以来已过去的天数。通过从日期中加上和减去整数，可以将过去或未来移动几天。 如果您想要移动较小的数量，以下常量可能会有所帮助：

```
ephem.hour
ephem.minute
ephem.second
```

12.2.9.1 local time

```
>>> d = ephem.Date('1997/3/9 5:13')
>>> ephem.localtime(d) # 将PyEphem日期转换为在本地时区中表示的Python的datetime对象。
datetime.datetime(1997, 3, 9, 0, 13, 0, 6)
>>> print(ephem.localtime(d))
1997-03-09 00:13:00.000006
```

12.2.9 Stars and Cities

```
# PyEphem提供了一个明亮的星星目录。
>>> rigel = ephem.star('Rigel') # 每次调用star()都会返回一个新的FixedBody，其坐标是指定星号的坐标。
>>> print('%s %s' % (rigel._ra, rigel._dec))
5:14:32.30 -8:12:06.0

# PyEphem知道122个世界城市。
>>> stuttgart = ephem.city('Stuttgart') # city()函数返回一个Observer，它的经度，纬度和海拔都是给定城市的。
>>> print(stuttgart.lon)
9:10:50.8
>>> print(stuttgart.lat)
48:46:37.6
```

12.2.10 Other Constants

- PyEphem为几个主要的星图集时代提供了常量：

  `B1900` `B1950` `J2000`

- PyEphem提供四个距离的参考长度，全部以米为单位：

  `ephem.meters_per_au` `ephem.earth_radius` `ephem.moon_radius` `ephem.sun_radius`

- PyEphem提供了每秒以米为单位的光速：

  `ephem.c`

---

13. pysofa

该项目旨在为某些程序提供一个python界面由史密森天体物理观测站（SAO）开发。 其中一个主要目标是通过XPA与来自python shell的ds9进行通信协议。 它为XPA库和子集提供了python包装基于XPA模块的ds9的python模块。

```
>>> import pysao
# run new instance of ds9
>>> ds9 = pysao.ds9()
```

```
>>> import numpy
>>> im = numpy.reshape(numpy.arange(100), (10, 10))
# display 2-d array
>>> ds9.view(im)

>>> import pyfits
>>> f = pyfits.open('test.fits')
# display first extension of fits file
>>> ds9.view(f[0])

# access with XPA method.
>>> ds9.set('file test.fits')
>>> ds9.get('file')

# list available xpa commands
>>> ds9.xpa_help()

# help on the specific xpa command
>>> ds9.xpa_help("tile")
```

14. spectrum

Spctrum包含使用基于傅立叶变换，参数方法或特征值分析的方法估计功率谱密度的工具：

- 傅立叶方法基于相关图，周期图和韦尔奇估计。 标准的削减窗口（汉恩，汉明，布莱克曼）和更奇特的窗口（DPSS，泰勒，...）。
- 参数方法基于Yule-Walker，BURG，MA和ARMA，协方差和修正的协方差方法。
- 基于特征分析（例如，MUSIC）和最小方差分析的非参数方法也被实现。
- Multitapering也可用

目标受众多元化。 尽管信号的功率谱的使用在电气工程（例如无线电通信，雷达）中是基本的，但它具有广泛的应用范围，从宇宙学（例如，在2016年检测引力波）到音乐（模式检测）或生物学 （质谱）。

14.1

14.2 Overview of available PSD methods

14.3 Tutorials

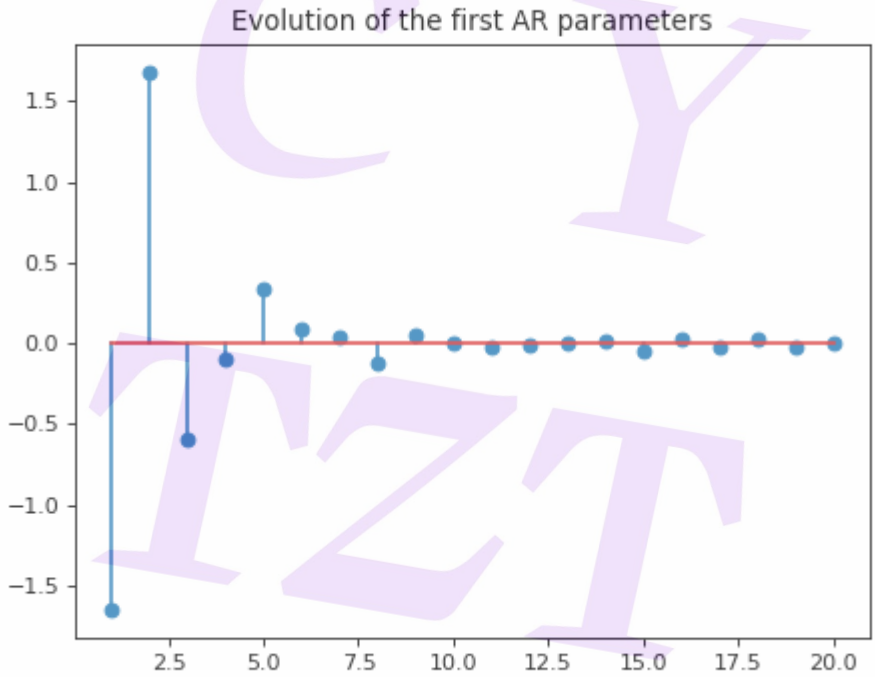
14.3.1 Yule Walker example(尤尔沃克)

以下示例说明了`aryule()`函数的用法，该函数允许您估计一组数据的自回归系数。

```
import scipy.signal
from spectrum import aryule

a = [1, -2.2137, 2.9403, -2.1697, 0.9606] # 定义一个AR滤波器系数列表
y = scipy.signal.lfilter([1], a, randn(1, 1024)) #与他们创建一些有噪音的数据
ar, variance, coeff_reflection = aryule(y[0], 20) #这个数组将成为测试Yule-Walker函数的数据，即aryule()。 我们的目标是估计y的AR系数。 由于我们不知道自回归估计的顺序，我们首先将顺序设置为20。
```

1、通过查看`coeff_reflection`输出，看起来对于阶数> 4，AR系数相当小（参见下图）。 从情节来看，选择一个指令似乎是一个合理的选择



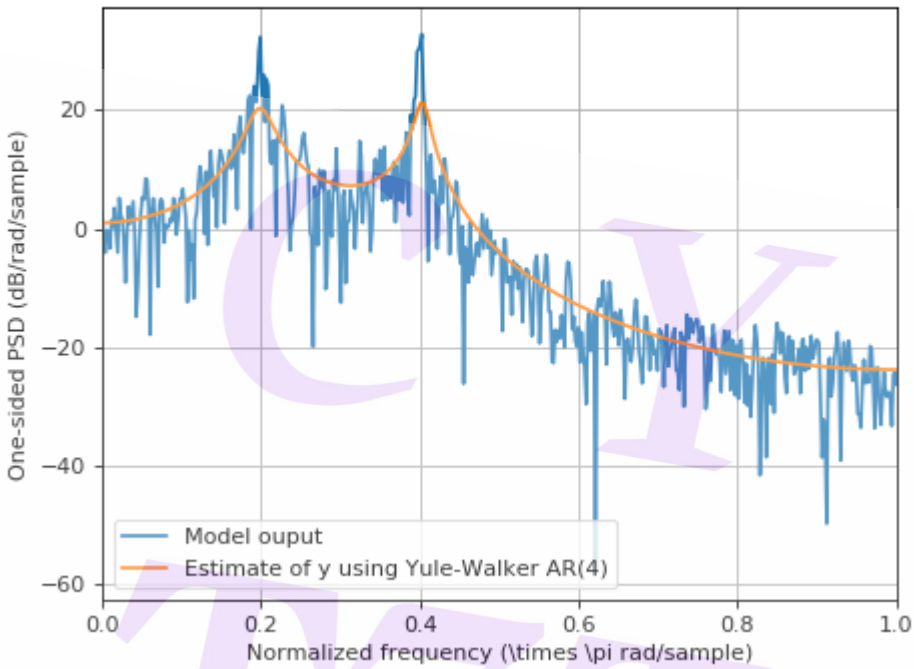
2、可以使用以下公式从ar值绘制PSD：

```
from pylab import log10, linspace, plot, xlabel, ylabel, legend, randn, pi
import scipy.signal
from spectrum import aryule, Periodogram, arma2psd
# Create a AR model
a = [1, -2.2137, 2.9403, -2.1697, 0.9606]
# create some data based on these AR parameters
y = scipy.signal.lfilter([1], a, randn(1, 1024))
# if we know only the data, we estimate the PSD using Periodogram
p = Periodogram(y[0], sampling=2)  # y is a list of list hence the y[0]
p.plot(label='Model ouput')

# now, let us try to estimate the original AR parameters
AR, P, k = aryule(y[0], 4)
PSD = arma2psd(AR, NFFT=512)
PSD = PSD[len(PSD):len(PSD)//2:-1]
plot(linspace(0, 1, len(PSD)), 10*log10(abs(PSD)*2./(2.*pi)),

    label='Estimate of y using Yule-Walker AR(4)')
```

```
xlabel(r'Normalized frequency (\times \pi rad/sample)')
ylabel('One-sided PSD (dB/rad/sample)')
legend()
```



14.3.2. PBURG

*Pburg*函数: 功能:利用Burg方法进行功率谱估计。

此处为另一种基于`arburg()`估计AR模型的方法。

> 这个例子的灵感来自Marple书中的一个例子。

```
from pylab import log10, pi, plot, xlabel, randn
import scipy.signal
from spectrum import arma2psd, arburg

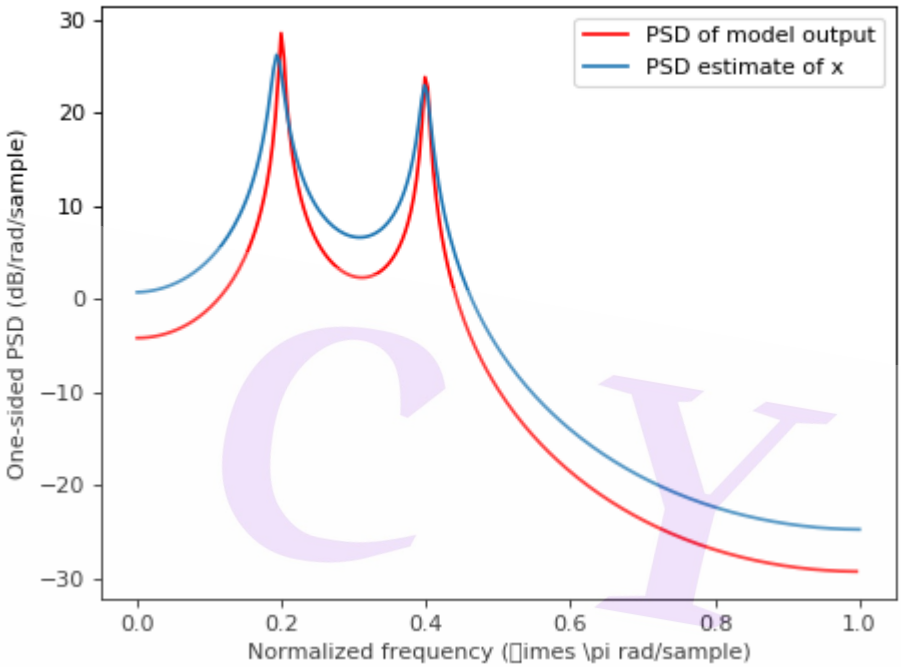
# Define AR filter coefficients
a = [1, -2.2137, 2.9403, -2.1697, 0.9606];

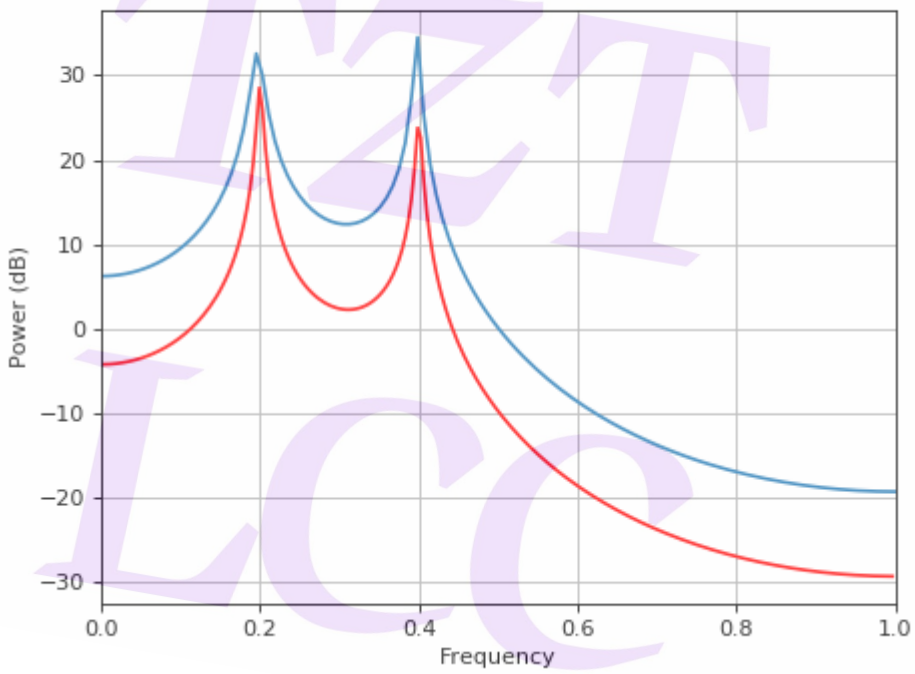
[w,H] = scipy.signal.freqz(1, a, 256)
Hp = plot(w/pi, 20*log10(2*abs(H)/(2.*pi)),'r')

x = scipy.signal.lfilter([1], a, randn(256))
AR, rho, ref = arburg(x, 4)

PSD = arma2psd(AR, rho=rho, NFFT=512)
PSD = PSD[len(PSD):len(PSD)//2:-1]

plot(linspace(0, 1, len(PSD)), 10*log10(abs(PSD)*2./(2.*pi)))
xlabel('Normalized frequency (\times \pi rad/sample)')
```





14.3.3 Variance outputs

`arburg()`函数返回AR参数，但也可以估计方差。

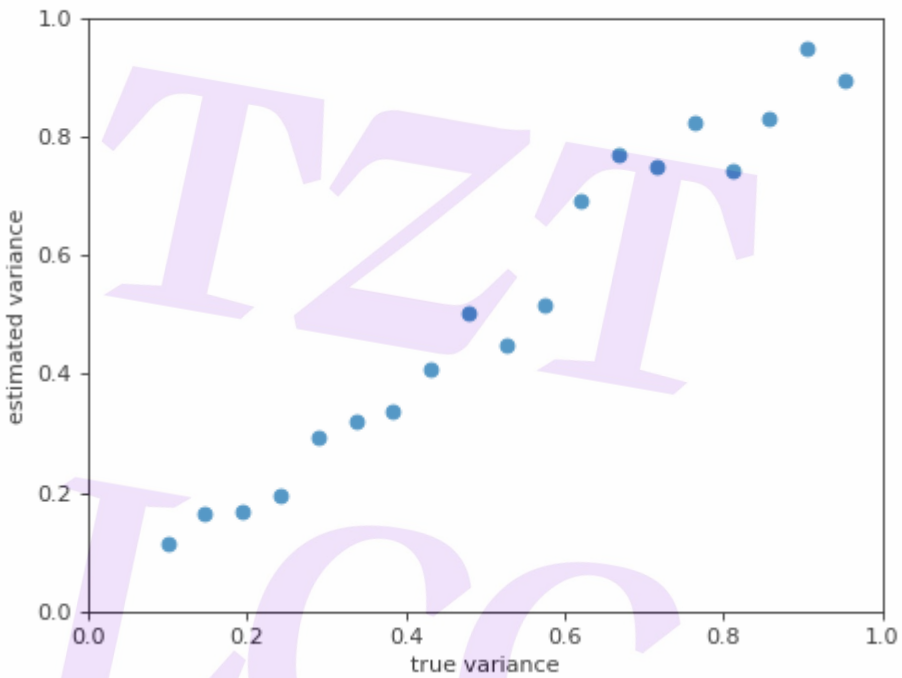
以下示例绘制了估计方差（使用arburg函数）与不同方差值的真实方差。 换句话说，我们绘制了方差可以被估计的准确度。

```
from pylab import plot, xlabel, ylabel, plot, axis, linspace, randn
import scipy.signal
from spectrum import arburg
```

```
# Define AR filter coefficients
a = [1, -2.2137, 2.9403, -2.1697, 0.9606];

# for different variance,
true_variance = linspace(0.1, 1, 20)
estimated_variance = []
for tv in true_variance:
    x = scipy.signal.lfilter([1], a, tv**0.5 * randn(1,256))
    AR, v, k = arburg(x[0], 4) # we estimate the AR parameter and variance
    estimated_variance.append(v)

plot(true_variance, estimated_variance, 'o')
xlabel('true variance')
ylabel('estimated variance')
plot([0,0],[1,1])
axis([0,1,0,1])
```



14.3.4 Windowing

在频谱分析中，通常将输入数据乘以逐渐变细的窗口。

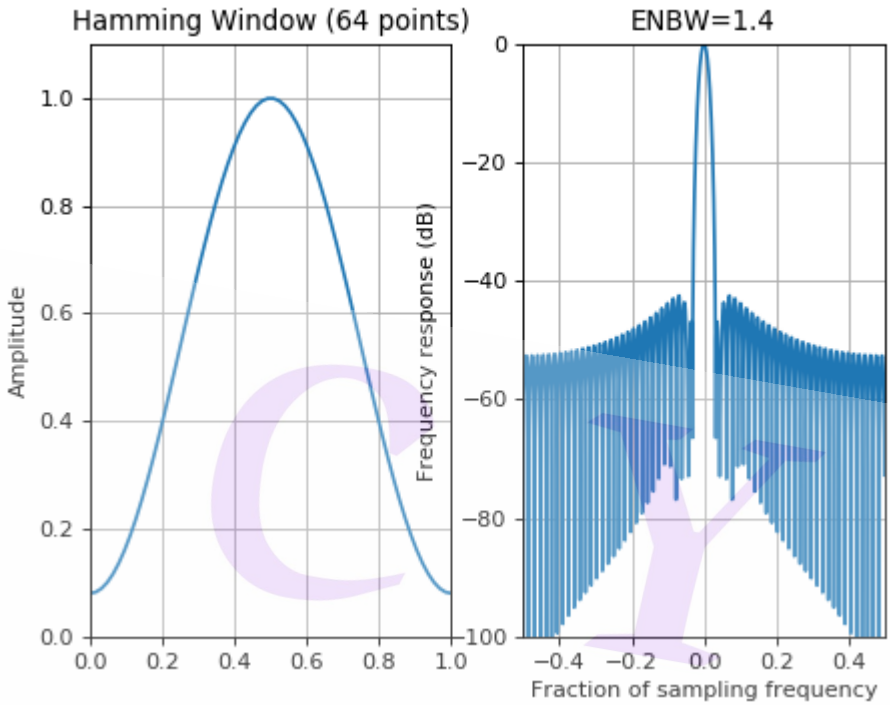
许多窗口在窗口模块中实现并可用，以及在时间和频率域中绘制窗口的实用程序。 一些已经实现的窗口是：

函数名称	函数功能
spectrum.window.window_bartlett(N)	Bartlett窗口（numpy.bartlett的包装）也被称为Fejer
spectrum.window.window_blackman(N[, alpha])	Blackman窗口
spectrum.window.window_gaussian(N[, alpha])	Gaussian窗口
spectrum.window.window_hamming(N)	Hann 窗口
spectrum.window.window_hann(N)	Hann 窗口 (or Hanning).
spectrum.window.window_kaiser(N[, beta, method])	Kaiser 窗口
spectrum.window.window_lanczos(N)	Lanczos窗口也被称为sinc窗口。
spectrum.window.window_nuttall(N)	Nuttall逐渐变细的窗口
spectrum.window.window_tukey(N[, r])	Tukey渐变窗口（或余弦渐变窗口）

请参阅 window 模块以获取完整的窗口列表。 还要注意波形提供了额外的 waveform 。

14.3.4.1 Window object

```
# 有一个类窗口可以简化渐变窗口的操作。
from spectrum.window import Window
N = 64                      # N是所需窗口的长度
w = Window(N, 'hamming')    # “hamming”是名称
w.plot_time_freq()
```



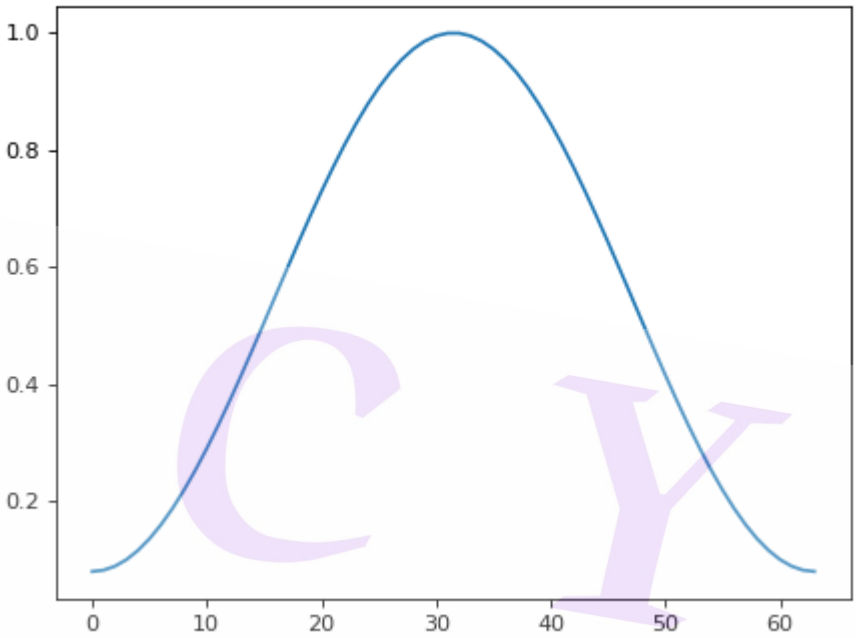
> 有很多不同的窗口，其中一些需要参数。

14.3.4.2 Simple window function call

可以探索模块以获取窗口功能。 例如，如果您查找 `Hamming` 窗口，则应该找到一个名为 `window_hamming()` 的函数。 你可以看看它如下：

```
from spectrum.window import window_hamming
from pylab import plot

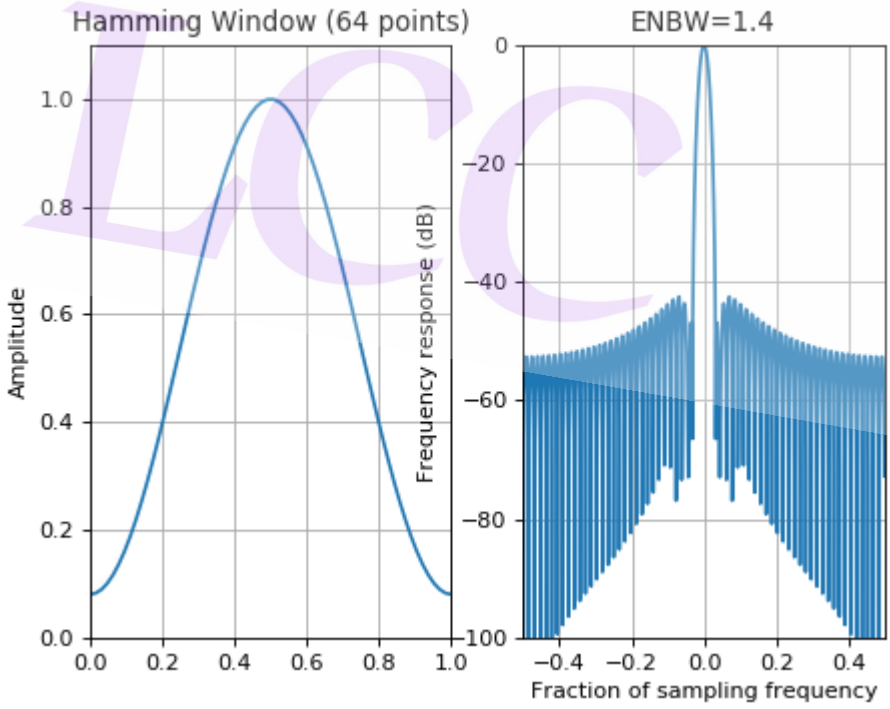
N = 64
w = window_hamming(N)
plot(w)
```



14.3.4.3 Window Visualisation

使用 `window_visu()` 快速查看窗口形状及其频率行为

```
from spectrum.window import window_visu
N = 64
window_visu(N, 'hamming')
```

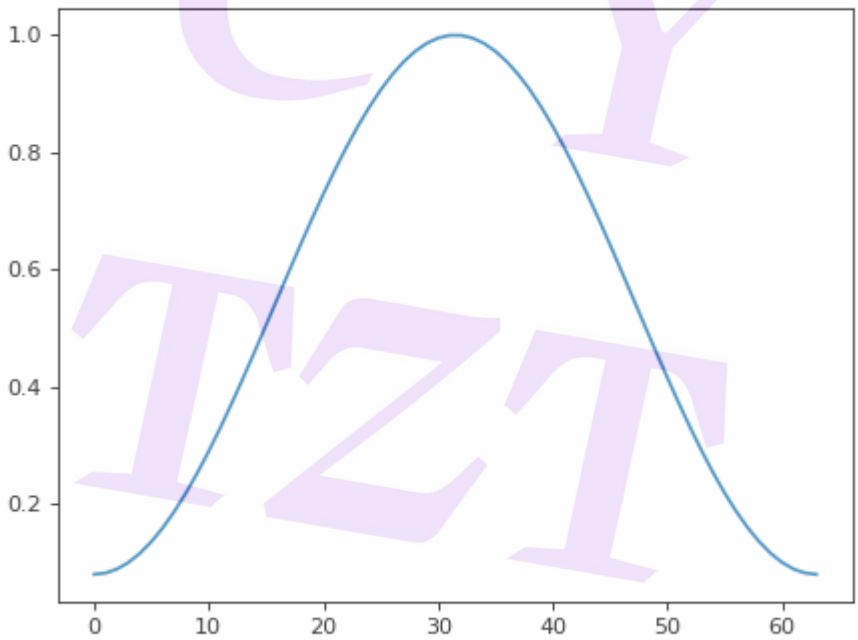


14.3.4.4 Window Factory

不想使用对象方法，则可能需要使用名为 `create_window()` 的 `Factory` 函数（Window类依赖于此函数）。 以前的汉明窗口可以使用：

```
from spectrum.window import create_window
from pylab import plot

N = 64
w = create_window(N, 'hamming')
plot(w)
```



14.3.5 All PSD methods

这个例子用于生成正面图像。 它显示了如何使用Spectrum中可以找到的不同PSD类。

```
import spectrum
from spectrum.datasets import marple_data
from pylab import legend, ylim
norm = True
sides = 'centerdc'

# MA method
p = spectrum.pma(marple_data, 15, 30, NFFT=4096)
p(); p.plot(label='MA (15, 30)', norm=norm, sides=sides)

# ARMA method
p = spectrum.parma(marple_data, 15, 15, 30, NFFT=4096)
p(); p.plot(label='ARMA(15,15)', norm=norm, sides=sides)
```

```
# yulewalker
p = spectrum.pyule(marple_data, 15, norm='biased', NFFT=4096)
p(); p.plot(label='YuleWalker(15)', norm=norm, sides=sides)

#burg method
p = spectrum.pburg(marple_data, order=15, NFFT=4096)
p(); p.plot(label='Burg(15)', norm=norm, sides=sides)

#covar method
p = spectrum.pcovar(marple_data, 15, NFFT=4096)
p(); p.plot(label='Covar(15)', norm=norm, sides=sides)

#modcovar method
p = spectrum.pmodcovar(marple_data, 15, NFFT=4096)
p(); p.plot(label='Modcovar(15)', norm=norm, sides=sides)

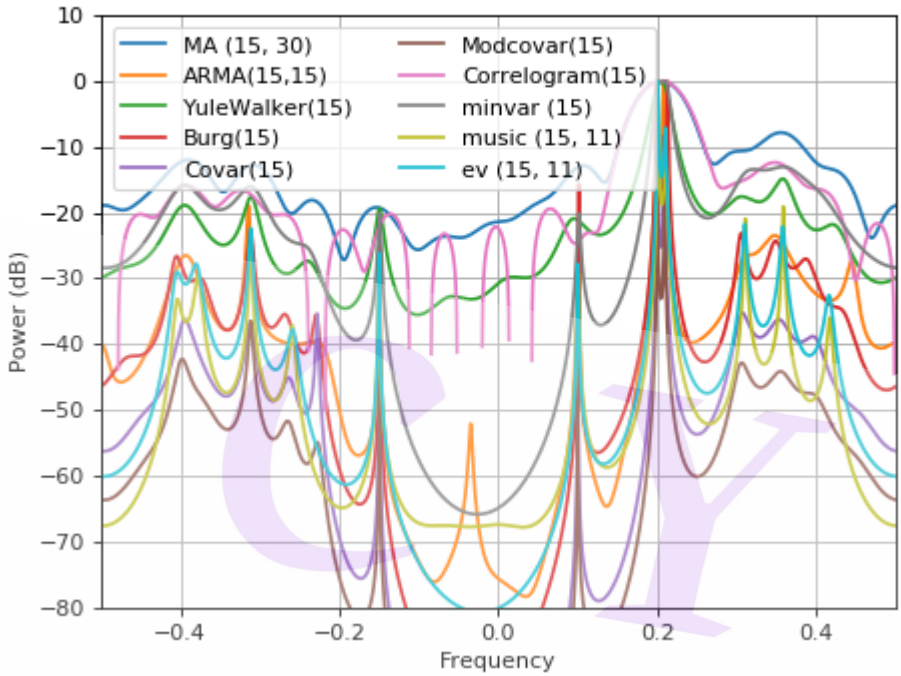
# correlagram
p = spectrum.pcorrelogram(marple_data, lag=15, NFFT=4096)
p(); p.plot(label='Correlogram(15)', norm=norm, sides=sides)

#minvar
p = spectrum.pminvar(marple_data, 15, NFFT=4096)
p(); p.plot(label='minvar (15)', norm=norm, sides=sides)

#music
p = spectrum.pmusic(marple_data, 15, 11, NFFT=4096)
p(); p.plot(label='music (15, 11)', norm=norm, sides=sides)

#ev
p = spectrum.pev(marple_data, 15, 11, NFFT=4096)
p(); p.plot(label='ev (15, 11)', norm=norm, sides=sides)

legend(loc='upper left', prop={'size':10}, ncol=2)
ylim([-80,10])
```



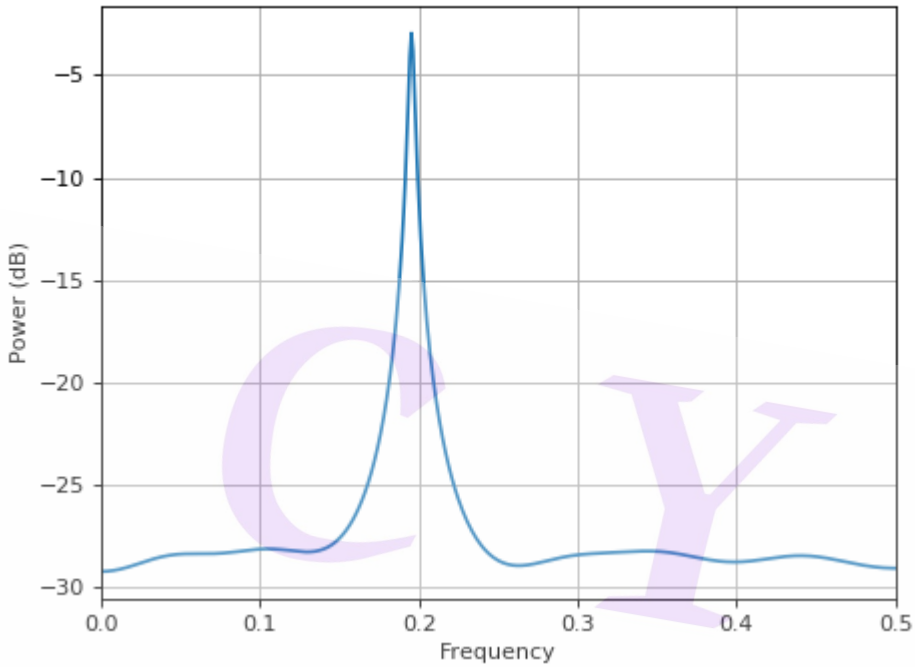
14.3.6 What is the Spectrum object ?

以下示例显示如何使用Spectrum。

```
from spectrum import Spectrum, data_cosine, speriodogram, minvar
p = Spectrum(data_cosine(), sampling=1024) #创建一个Spectrum实例（第一个参数是时间序列/数据）

p.N          # 一些信息被存储并且可以稍后被检索
p.sampling
p.psd        # 关于PSD估计方法的信息
psd = speriodogram(p.data) # 它返回一个警告信息(告诉你PSD尚未被计算）可以独立计算它，并手动设置psd属性
p.method = minvar # 或者可以将一个函数关联到method属性
p(15, NFFT=4096)  # 用适当的可选参数调用函数

'''在这两种情况下，PSD现在都保存在psd属性中。
   当然，如果您已经知道要使用的方法，那么直接调用相应的类就简单多了，如前面的部分和示例所示：'''
p =  pminvar(data_cosine(), 15)
p()
p.plot()
```



14.4 Reference

14.4.1 Fourier Methods

14.4.1.1 Power Spectrum Density based on Fourier Spectrum

`default_NFFT` = *4096* // 用于计算FFT的默认样本数

`randn`(*d0*, *d1*, ..., *dn*)

> For random samples from
>
> $$N(\mu, \sigma^2)$$
>
> , use:
>
> `sigma * np.random.randn(...) + mu`

```
>>> np.random.randn()
2.1923875335537315 #random

# Two-by-four array of samples from N(3, 6.25)
>>> 2.5 * np.random.randn(2, 4) + 3
array([[-4.49401501,  4.00950034, -1.81814867,  7.29718677],  #random
       [ 0.39924804,  4.68456316,  4.99394529,  4.84057254]]) #random
```

`spectrum_set_level`(*level*) 该模块提供了周期图 (classics, daniell, bartlett)

模块名称	模块功能
class Periodogram(data[, sampling, window, NFFT, ...])	The Periodogram class provides an interface to periodogram PSDs
DaniellPeriodogram(data, P[, NFFT, detrend, ...])	Return Daniell's periodogram.
speriodogram(x[, NFFT, detrend, sampling, ...])	Simple periodogram, but matrices accepted.
WelchPeriodogram(data[, NFFT, sampling])	Simple periodogram wrapper of numpy.psd function.
speriodogram(x[, NFFT, detrend, sampling, ...])	Simple periodogram, but matrices accepted.
`CORRELOGRAMPSD`(*X, Y=None, lag=-1, window='hamming', norm='unbiased', NFFT=4096, window_params={}, correlation_method='xcorr'*)	PSD estimate using correlogram method.
class `pcorrelogram`(*data, sampling=1.0, lag=-1, window='hamming', NFFT=None, scale_by_freq=True, detrend=None*)	The Correlogram class provides an interface to `CORRELOGRAMPSD()`.

14.4.1.2 Tapering Windows

模块名称	模块功能
Window(N[, name, norm])	Window tapering object
window_visu([N, name])	A Window visualisation tool
create_window(N[, name])	Returns the N-point window given a valid name
window_hann(N)	Hann window (or Hanning).
window_hamming(N)	Hamming window
window_bartlett(N)	Bartlett window (wrapping of numpy.bartlett) also known as Fejer
window_bartlett_hann(N)	Bartlett-Hann window
window_blackman(N[, alpha])	Blackman window
window_blackman_harris(N)	Blackman Harris window
window_blackman_nuttall(N)	Blackman Nuttall window
window_bohman(N)	Bohman tapering window
window_chebwin(N[, attenuation])	Cheb window
window_cosine(N)	Cosine tapering window also known as sine window.
window_flattop(N[, mode, precision])	Flat-top tapering window
window_gaussian(N[, alpha])	Gaussian window
window_hamming(N)	Hamming window
window_hann(N)	Hann window (or Hanning).
window_kaiser(N[, beta, method])	Kaiser window
window_lanczos(N)	Lanczos window also known as sinc window.
window_nuttall(N)	Nuttall tapering window
window_parzen(N)	Parsen tapering window (also known as de la Valle-Poussin)
window_taylor(N[, nbar, sll])	Taylor tapering window
window_tukey(N[, r])	Tukey tapering window (or cosine-tapered window)

14.4.2 Multitapering

该模块提供了用于谱估计的多锥化方法。 锥形窗口的计算在计算上是昂贵的，C模块用于计算锥形窗口（参见 `spectrum.mtm.dpss()` ）。

估计本身可以使用 `spectrum.mtm.pmtm()` 函数执行

模块名称	模块功能
class `MultiTapering` (*data*, *NW=None*, *k=None*, *NFFT=None*, *e=None*, *v=None*, *method='adapt'*, *scale_by_freq=True*, *sampling=1*)	有关详细信息，请参阅 `pmtm()`
`dpss` (*N*, *NW=None*, *k=None*)	计算离散长螺旋球体序列也称为平移序列，以及相应的特征值。
`pmtm` (*x*, *NW=None*, *k=None*, *NFFT=None*, *e=None*, *v=None*, *method='adapt'*, *show=False*)	多重谱估计

14.4.3 Parametric methods

14.4.3.1 Power Spectrum Density based on Parametric Methods

14.4.3.1.1 ARMA and MA estimates (yule-walker)

ARMA和MA计量，ARMA和MA PSD计量。

ARMA model and Power Spectral Densities.

`arma_estimate` 自回归和移动平均估计量。 `ma` 移动平均估计量。 `arma2psd` 根据ARMA值计算功率谱密度。 *class* `parma` 使用ARMA估计器创建PSD的类。 *class* `pma` 使用MA估计器创建PSD的类。.

14.4.3.1.2 AR estimate based on Burg algorithm

BURG方法的AR模型估计。

`arburg(X, order[, criteria])` Burg算法估计复杂的自回归参数。

class `pburg(data, order[, criteria, NFFT, ...])` 基于Burg算法创建PSD的类

14.4.3.1.3 AR estimate based on YuleWalker

Yule Walker方法来估计AR值。

`aryule` (X, order[, norm, allow_singularity]) 使用Yule-Walker方法计算AR系数

class ``pyule`` (data, order[, norm, NFFT, sampling, ...]) 基于Yule Walker方法创建PSD的类

14.4.3.2 Criteria

该模块提供了自动选择参数PSD估计或伪频谱估计（例如音乐）的顺序的标准。

14.4.4 Other Power Spectral Density estimates

14.4.4.1 Covariance method

> arcovar(x, order)

简单且快速实现协方差AR估计。

> arcovar_marple(x, order)

使用协方差法估计AR模型参数。

> class pcovar(data, order, NFFT=None, sampling=1.0, scale_by_freq=False)

基于协方差算法创建PSD的类

> arcovar_marple(x, order)

使用协方差法估计AR模型参数

14.4.4.2 Eigen-analysis methods

> eigen(X, P, NSIG=None, method='music', threshold=None, NFFT=4096, criteria='aic', verbose=False)

使用特征向量法（EV或Music）的伪谱，该功能计算音乐或特征值（EV）噪声子空间频率估计器。

> class pev(data, IP, NSIG=None, NFFT=None, sampling=1.0, scale_by_freq=False, threshold=None, criteria='aic', verbose=False)

使用ARMA估计器创建PSD的类。

14.4.4.3 Minimum Variance estimator(最小方差谱估计)

pminvar(data, order[, NFFT, sampling, ...]) 基于最小方差谱估计来创建PSD的类

> minvar(X, order, sampling=1.0, NFFT=4096)

最小方差谱估计（MV）

$$P_{MV}(f) = \frac{T}{e^H(f) R_p^{-1} e(f)}$$

> class pminvar(data, order, NFFT=None, sampling=1.0, scale_by_freq=False)

基于最小方差谱估计来创建PSD的类

14.4.4.4 Modified Covariance method

与modcovar方法有关的模块

> modcovar(x, order)

简单且快速实现协方差AR估计

> modcovar_marple(X, IP)

求解修正的协方差最小二乘法正则方程的快速算法

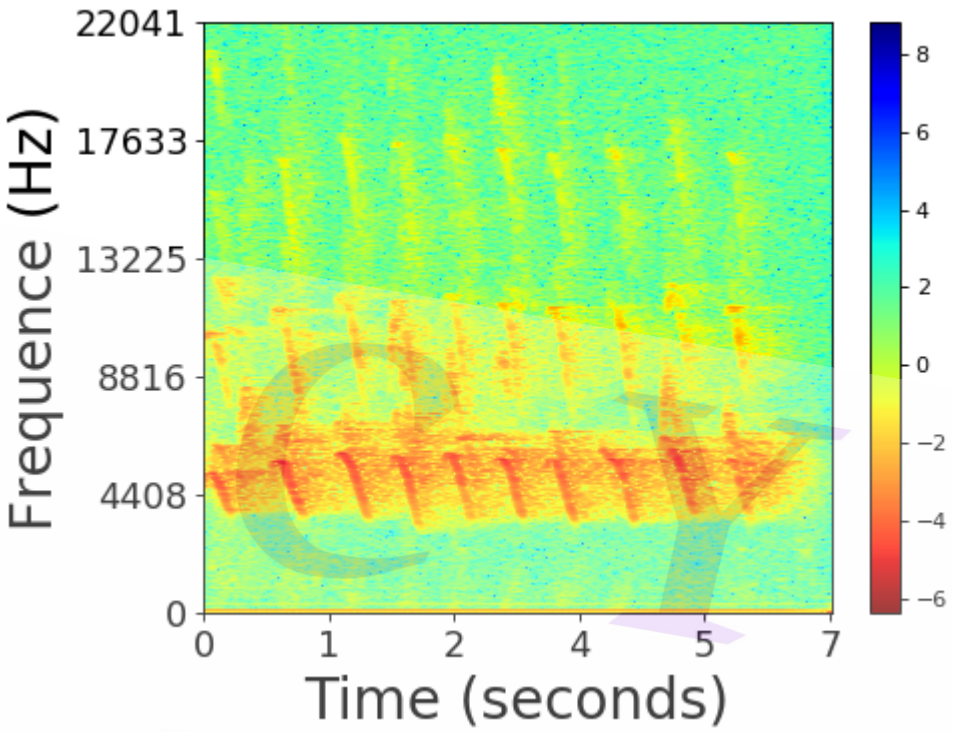
> class pmodcovar(data, order, NFFT=None, sampling=1.0, scale_by_freq=False)

基于修改的协方差算法创建PSD的类

14.4.4.5 Spectrogram

> class Spectrogram(signal, ws=128, W=4096, sampling=1, channel=1)

谱图的简单例子



> `periodogram`()
>
> `plot`(*filename=None, vmin=None, vmax=None, cmap='jet_r'*)
>
> `pmtm`()

---

15. pywcsgrid2

pywcsgrid2是一个python模块，与matplotlib一起使用来显示天文数字图像。 它提供了一个适用于显示拟合图像的自定义Axes类（从mpl的原始Axes类派生）。 其主要功能是在合适的天空坐标中绘制刻度线，标记线和网格。

- 提供基于`mpl_toolkits.axisartist`模块的定制Matplotlib轴类
- 与`mpl_toolkits.axes_grid1`模块轻松集成
- 有关抽样，请参阅图库

15.1 pywcsgrid2.axes.wcs

15.1.1 *class* `pywcsgrid2.axes_wcs.GridHelperWcsBase`

函数名称
`set_ticklabel1_type`
`set_ticklabel2_type`
`set_ticklabel_type`
`update_wcsgrid_params`

15.1.2*class* `pywcsgrid2.axes_wcs.AxesWcs`

函数名称
`add_beam_size`
`add_compass`
`add_inner_title`
`add_size_bar`
`set_default_label`
`set_default_path_effects`
`set_display_coord_system`
`set_label_type`
`set_ticklabel1_type`
`set_ticklabel2_type`
`set_ticklabel_type`
`swap_tick_coord`
`update_wcsgrid_params`

16. Aperture Photometry 孔径测光

16.1 Introduction 简介

- 定义好的 `Aperture Class` 有:

  Circular / Elliptical / Rectangular + Aperture / Annulus = 6种排列组合

```
from photutils import CircularAperture
```

- 上面每一种类都有天球坐标的对应变种：名字前加上 Sky，也是6种

16.2 Creating Aperture Object 创建对象

- 每一个 Aperture 对象都有位置、大小参数
  - `position`：像素坐标，需为 `(x, y)` 元组，或元组的列表，或 `Nx2` 的数组，也可以是 Quantity 对象
  - Circle - `r`：孔径的半径，单位为像素
  - Elliptical - a, b：半长 / 短轴
  - Elliptical - θ：半长轴倾斜的角度，从x正半轴方向逆时针计

```
positions = [(30., 30.), (40., 40.)]
apertures = CircularAperture(positions, r=3.)
```

- 创建Sky Aperture 对象时，position 参数需要输入 `SkyCoord` 对象，r 参数也需要加上角度单位

```
positions = SkyCoord(l=[1.2, 2.3] * u.deg, b=[0.1, 0.2] * u.deg, frame='galactic')
apertures = SkyCircularAperture(positions, r=4. * u.arcsec)
```

注意这些变种类只是在确定中心位置时用了天球坐标，孔径大小只是简单地将角度转化为像素，并未考虑投影效果（否则的话aperture就不再是原来的形状了）

16.3 Performing AP 实施孔径测光

16.3.1 `aperture_photometry()` 函数

- 输入参数
  - `data`：
    - **2D** array-like，Quantity，ImageHDU，HDUList
    - 需是经过背景扣除 (background-subtracted) 的数据！
    - 如果 data 本身不带单位，可以通过 unit 参数传入；如果是 HDU，会自动从 `BUNIT` 引入
  - `apertures`：须是 Aperture 或 SkyAperture 对象
  - `error`：
    - array-like，Quantity
    - 须和 data 一样形状！
    - The pixel-wise(单像素) Gaussian 1-sigma errors of the input `data`
  - `method`：如何计算 aperture 边界与像素交叉的部分，有三种方法：
    - exact：默认，精度最高但算得最慢
    - center：由中心是否被包括，来决定整个像素点是否被包括；精度最低但算得最快
    - subpixel：把整个像素分成若干个小像素，再用 center 方法决定每个小像素是否被包括；需设置 `subpixels` 参数来确定小像素的个数，默认是5
- 返回参数：`QTable`，其实就是带单位的 Table，即表内数据是 Quantity
  - `'id'`：The source ID.
  - `'xcenter'`，`'ycenter'`：孔径中心的像素坐标
  - `'celestial_center'`：孔径中心的天球坐标，只有输入 SkyAperture 对象时才返回

    - `'aperture_sum'`：孔内个像素点数值求和
    - `'aperture_sum_err'`: 对应的不确定度，只有当 error 参数不是 None 时才返回
- 要在不同的位置使用不同的 aperture，可以向 aperture_photometry() 传入 list of aperture objects，这样会返回多列 sum 结果

  通常这些 column name 是 `aperture_sum_i`，i = 0, 1, 2, ...

```
radii = [3., 4., 5.]
apertures = [CircularAperture(positions, r=r) for r in radii]   # 巧用列表推导式
phot_table = aperture_photometry(data, apertures)
```

16.4 Background Subtraction 背景消除

16.4.1 Global

详情见 “Background Estimation” `photutils.background`，如果 `bkg` 是代表背景的二维数组，在输入 data 参数时把背景减去即可：

```
phot_table = aperture_photometry(data - bkg, apertures)
```

16.4.2 Local ☆

- 通过套在测光孔径外的圆环来计算局部的背景：

```
apertures = CircularAperture(positions, r=3)
annulus_apertures = CircularAnnulus(positions, r_in=6., r_out=8.)
```

- 但不能直接用两个 sum 的数据相减，因为孔与环的面积不一样，故需要除以环的面积得到背景平均值，面积可用 `area()` 方法得到：

```
bkg_mean = phot_table['aperture_sum_1'] / annulus_apertures.area()
```

- 最后用背景平均值乘上测光孔面积，即需要见去的背景值：

```
bkg_sum = bkg_mean * apertures.area()
final_sum = phot_table['aperture_sum_0'] - bkg_sum
phot_table['residual_aperture_sum'] = final_sum
# 后面两步是通过列表操作完成的
```

16.4.3 Pixel Masking

- 用 `mask` 参数来忽略 / 排除某些像素的数值，比如说坏数据或是需要被遮盖的极亮光源
- mask 需要输入与 data 形状一样的布尔类型数组

16.5 Error Estimation 误差估计

- `'aperture_sum_err'` 的算法：

$$\Delta F = \sqrt{\sum_{i \in A} \sigma_{tot,i}^2}$$

- ΔF 是被圈起来的 source 的误差，A 是 non-masked pixels，$\sigma_{tot,i}$ 是输入的 error 数组

- 这种算法假定 `error` 输入的是包括了所有的误差的数据，无论是点源造成的 Poisson noise 还是其他不相关的噪声，但有可能实际上它只包含了纯背景噪声，此时若需要包含 Poisson noise，要用 `calc_total_err()` 函数
  - 参数：
    - data：原始数据
    - bkg_error：纯背景噪声数组
    - effective_gain：Ratio of counts (e.g., electrons or photons) to the units of `data` used to calculate the Poisson error of the sources
  - 如果要带单位的话，全部参数输入的都得是 Quantity 对象
- `calc_total_err()` 的算法：

$$\sigma_{tot} = \sqrt{\sigma_b^2 + \frac{I}{g}}$$

- σ_b = bkg_error，I = data，g = effective_gain
- 和 SEtractor 不同的是，$I < 0$ 的像素点的 Poisson noise 不计入 σ_{tot}
- 关于参数单位的对应关系：If your input `data` are in units of ADU(analog-to-digital unit 模数单元), then `effective_gain` should represent **electrons/ADU**. If your input `data` are in units of electrons/s then `effective_gain` should be the **exposure time** or an exposure time map

```
>>> from photutils.utils import calc_total_error
>>> effective_gain = 500  # seconds, exposure time
>>> error = calc_total_error(data, bkg_error, effective_gain)
>>> phot_table = aperture_photometry(data - bkg, apertures, error=error)
```

16.6 Pixel Masking & Aperture Mask

16.7 AP using Sky Coordinates

- `aperture_photometry()` 函数会自动从 hdu 参数中提取 wcs 信息，然后对传入的 SkyAperture 进行天球坐标到像素坐标的转化：

```
hdu = datasets.load_spitzer_image()
# hdu 是包含了 data & header 的 HDU 对象!
catalog = datasets.load_spitzer_catalog()

positions = SkyCoord(catalog['l'], catalog['b'], frame='galactic')
# positions 是 SkyCoord 对象
apertures = SkyCircularAperture(positions, r=4.8 * u.arcsec)
phot_table = aperture_photometry(hdu, apertures)
```

- 在某些时候注意**转换单位**!

如想和官方数据做对比，一般官方数据的单位是 mJ，而有些fits文件的flux单位是 MJy/sr，需要转换为 mJy/pixel：ss

```
# 1.2 是像素宽度对应的张角
factor = (1.2 * u.arcsec) ** 2 / u.pixel
fluxes_catalog = catalog['f4_5']
converted_aperture_sum = (phot_table['aperture_sum'] *
                          factor).to(u.mJy / u.pixel)
```

17.Metpy包

MetPy是由UCAR开发的Python中的一个工具集合，用于读取、可视化和气象数据计算。 MetPy旨在与其他Python包，包括Numpy（数值计算包），Scipy和Matplotlib（绘图包）项目，并添加特定于气象的功能。

MetPy的目标是在现有的科学Python生态系统中提供GEMPAK（可能类似于NCL）的类功能，包括Numpy，Scipy和Matplotlib等项目。

MetPy的设计理念的一部分是使它的例程可以用于任何气象Python应用程序; 这意味着可以很容易地抽出LCL计算并使用它，或者用您自己的数据代码重新使用Skew-T。 MetPy还以拥有良好的文件记录和经过良好测试而自豪，因此持续的维护很容易管理。

在天文学用途中，该包主要运用于行星大气（八大行星）的研究；在大气科学中，该包主要用于对地球大气层和常规气象观测的研究。

主要功能：

- 以气象为重点的绘图（例如T-Inp，台站图）
- 气象数据计算（例如平流，露点，混合比等物理量计算）
- 读取常见的气象文件格式（例如NC文件、格点文件、GINI卫星图像，NEXRAD Level 2和3）
- 网格和插值工具
- 颜色表操作

如何安装：

```
pip install metpy
或
conda install -c conda-forge metpy
```

MetPy 使用之前需要如下的包:

- NumPy >= 1.10.0
- SciPy >= 0.14.0
- Matplotlib >= 1.4.0
- pint >= 0.8

17.0 The MetPy API

`metpy.constants`	**气象常量集合**
`metpy.units`	单位支持
`metpy.io`	读/写文件
`metpy.io.cdm`	模仿通用数据模型（CDM）的API的工具。
`metpy.calc`	气象计算模块
`metpy.plots`	气象绘图模块
`metpy.plots.ctables`	使用自定义颜色表
`metpy.gridding`	将不规则间隔数据内插到常规网格的工具（插值）

各个模块的底层函数代码查看路径：(Window系统为例)

C:\Python27\Lib\site-packages\metpy*.py

17.1 Constants常数模块

即气象学重要常数的集合。

> 包含地球常量、水汽常量、干空气常量等常量。

在使用此模块之前，需要导入该模块：

```
from metpy import constants
```

17.1.1Earth 地球物理常量

Name	Symbol	Short Name	Units	Description
earth_avg_radius	ReRe	Re	mm	Avg. radius of the Earth
earth_gravity	gg	g	m s−2m s−2	Avg. gravity acceleration on Earth
gravitational_constant	GG	G	m3kg−1s−2m3kg−1s−2	Gravitational constant
earth_avg_angular_vel	ΩΩ	omega	rad s−1rad s−1	Avg. angular velocity of Earth
earth_sfc_avg_dist_sun	dd	d	mm	Avg. distance of the Earth from the Sun
earth_solar_irradiance	SS	S	W m−2W m−2	Avg. solar irradiance of Earth
earth_max_declination	δδ	delta	degreesdegrees	Max. solar declination angle of Earth
earth_orbit_eccentricity	ee		NoneNone	Avg. eccentricity of Earth's orbit
earth_mass	meme	me	kgkg	Total mass of the Earth (approx)

17.1.2 Water

Name	Symbol	Short Name	Units	Description
water_molecular_weight水分子重量	MwMw	Mw	g mol−1g mol−1	Molecular weight of water
water_gas_constant 气体常数	RvRv	Rv	J (K kg) −1J (K kg)−1	Gas constant for water vapor
density_water 水的密度	ρlρl	rho_l	kg m−3kg m−3	Nominal density of liquid water at 0C
wv_specific_heat_press 水蒸汽恒压比热	CpvCpv	Cp_v	J (K kg) −1J (K kg)−1	Specific heat at constant pressure for water vapor
wv_specific_heat_vol 水蒸气恒定体积比热	CvvCvv	Cv_v	J (K kg) −1J (K kg)−1	Specific heat at constant volume for water vapor
water_specific_heat 0℃时液态水的比热	CplCpl	Cp_l	J (K kg) −1J (K kg)−1	Specific heat of liquid water at 0C
water_heat_vaporization 0℃时液态水的汽化潜热	LvLv	Lv	J kg−1J kg−1	Latent heat of vaporization for liquid water at 0C
water_heat_fusion 液态水的熔化潜热	LfLf	Lf	J kg−1J kg−1	Latent heat of fusion for liquid water at 0C
ice_specific_heat 0℃冰的比热	CpiCpi	Cp_i	J (K kg) −1J (K kg)−1	Specific heat of ice at 0C
density_ice 0℃冰的密度	ρiρi	rho_i	kg m−3kg m−3	Density of ice at 0C

17.1.3 Dry Air

Name	Symbol	Short Name	Units	Description
dry_air_molecular_weight	MdMd	Md	g / molg / mol	Nominal molecular weight of dry air at the surface of th Earth
dry_air_gas_constant	RdRd	Rd	J (K kg) −1J (K kg)−1	Gas constant for dry air at the surface of the Earth
dry_air_spec_heat_press	CpdCpd	Cp_d	J (K kg) −1J (K kg)−1	Specific heat at constant pressure for dry air
dry_air_spec_heat_vol	CvdCvd	Cv_d	J (K kg) −1J (K kg)−1	Specific heat at constant volume for dry air
dry_air_density_stp	ρdρd	rho_d	kg m−3kg m−3	Density of dry air at 0C and 1000mb

17.1.4 General Meteorology Constants

Name	Symbol	Short Name	Units	Description
pot_temp_ref_press	P0P0	P0	PaPa	Reference pressure for potential temperature
poisson_exponent	κκ	kappa	NoneNone	Exponent in Poisson's equation (Rd/Cp_d)
dry_adiabatic_lapse_rate 干绝热递减率	γdγd	gamma_d	K km−1K km−1	The dry adiabatic lapse rate
molecular_weight_ratio 水的分子量与干燥空气的分子量之比（比湿）	εε	epsilon	NoneNone	Ratio of molecular weight of water to that of dry air

17.2 units 单位模块

> This makes use of the `pint` library and sets up the default settings for good temperature support.

`pint.UnitRegistry` –整个包中使用的单元注册表。 任何在MetPy中使用单位都应该导入这个注册表并使用它来抓取单位。

17.2.1 Functions

`atleast_1d` **(*arrs)**	**将输入转换为至少具有一个维度的数组**
`atleast_2d` (*arrs)	将输入转换为至少有两个维度的数组
`check_units` (*units_by_pos, **units_by_name)	创建一个decorator来检查函数参数的单位。
`concatenate` (arrs[, axis])	Concatenate multiple values into a new unitized object.
`diff` (x, **kwargs)	沿着第n个维度计算差分
`masked_array` (data[, data_units])	Create a `numpy.ma.MaskedArray` with units attached.

17.2.2 Classes

`PintAxisInfo` **(units)**	**支持默认轴和刻度标签以及默认限制。**
`PintConverter` (registry)	在matplotlib的单位转换框架中实现对pint的支持。

17.2.3 Exceptions

`DimensionalityError` **(units1, units2[, dim1, ...])**	**Raised when trying to convert between incompatible units.**
`UndefinedUnitError` (unit_names)	Raised when the units are not defined in the unit registry.

17.3 io 模块

MetPy的IO模块包含读取文件的类。 这些类被写入以获取文件名（用于本地文件）或文件类对象; 这允许读取已经在内存中的文件（使用`io.StringIO`）或远程文件（使用`urlopen()`）。 还有一些类可以实现公共数据模型（CDM）中的概念。

> 当然，也可以使用`netcdf4-python`模块来进行nc文件的读取

17.3.1 Functions 函数

`get_upper_air_data` **(time, site_id[, source])**	**从网络上存档下载并解析高空观测数据。**
`is_precip_mode` (vcp_num)	确定NEXRAD雷达是否在降水模式下运行

17.3.2 Classes 类

`GiniFile` **(filename)**	**处理从NWS读取GINI格式卫星图像**
`Level2File` (filename)	处理读取NEXRAD Level 2数据
`Level3File` (filename)	处理读取大量NEXRAD 3级（NIDS）产品文件。

17.4 CDM 模块

本模块用于模仿通用数据模型（CDM）的API的工具。

CDM是用于表示各种数据的数据模型。 目标是成为不同数据集的简单通用界面。

> This API is a Python implementation in the spirit of the original Java interface in netCDF-Java.

17.4.1 Functions

函数	说明
`cf_to_proj` (var)	Convert a Variable with projection information to a Proj.4 Projection instance.

17.5 calc 计算模块

该模块包含各种气象计算。

> 低层函数代码查看位置：C:\Python27\Lib\site-packages\metpy\calc

17.5.1 Functions 函数

函数	说明
add_height_to_pressure (pressure, height)	计算高于另一个压力水平的特定高度的压力
add_pressure_to_height (height, pressure)	Calculate the height at a certain pressure above another height.
advection (scalar, wind, deltas)	Calculate the advection of a scalar field by the wind.
bulk_shear (pressure, u, v[, heights, ...])	Calculate bulk shear through a layer.
bunkers_storm_motion (pressure, u, v, heights)	Calculate the Bunkers right-mover and left-mover storm motions and sfc-6km mean flow.
cape_cin (pressure, temperature, dewpt, ...)	计算CAPE和CIN值
convergence_vorticity (u, v, dx, dy[, dim_order])	计算水平风的水平散度和垂直涡度
coriolis_parameter (latitude)	计算每个格点的科里奥利参数。
density (pressure, temperature, mixing[, ...])	Calculate density.
dewpoint (e)	根据蒸汽压力计算环境露点温度
dewpoint_rh (temperature, rh)	根据空气温度和相对湿度计算环境露点温度
divergence (u, v, dx, dy)	计算水平风的水平散度。
dry_lapse (pressure, temperature)	Calculate the temperature at a level assuming only dry processes.
dry_static_energy (heights, temperature)	Calculate the dry static energy of parcels. 计算静态能量
el (pressure, temperature, dewpt)	Calculate the equilibrium level.
equivalent_potential_temperature (pressure, ...)	Calculate equivalent potential temperature.
find_intersections (x, a, b[, direction])	Calculate the best estimate of intersection.
first_derivative (f, **kwargs)	计算格点上的一阶导数
friction_velocity (u, w[, v, perturbation, axis])	根据速度分量的时间序列计算摩擦速度
frontogenesis (thta, u, v, dx, dy[, dim_order])	计算温度场的二维运动学锋生
geopotential_to_height (geopot)	由位势高度计算几何高度

函数	说明
geostrophic_wind (heights, f, dx, dy)	计算从高度或位势给定的地转风
get_layer (pressure, *args, **kwargs)	Return an atmospheric layer from upper air data with the requested bottom and depth.
get_layer_heights (heights, depth, *args, ...)	Return an atmospheric layer from upper air data with the requested bottom and depth.
get_perturbation (ts[, axis])	根据时间序列的均值计算扰动。
get_wind_components (speed, wdir)	Calculate the U, V wind vector components from the speed and direction.
get_wind_dir (u, v)	计算u和v分量的风向
get_wind_speed (u, v)	计算u和v分量的风速。
h_convergence (u, v, dx, dy[, dim_order])	Calculate the horizontal divergence of the horizontal wind.
heat_index (temperature, rh[, mask_undefined])	根据当前温度和相对湿度计算热指数
height_to_geopotential (height)	计算给定高度的位势高度
height_to_pressure_std (height)	高度数据转换为压力（毫巴）
interp (x, xp, *args, **kwargs)	以指定轴上的任何形状插值数据。
interpolate_nans (x, y[, kind])	在y中插入NaN值。
isentropic_interpolation (theta_levels, ...)	将等压坐标中的数据插值到等熵坐标
kinematic_flux (vel, b[, perturbation, axis])	计算两个时间序列的通量。
laplacian (f, **kwargs)	计算网格点上的拉普拉斯算子
lat_lon_grid_spacing (longitude, latitude, ...)	Calculate the distance between grid points that are in a latitude/longitude format.
lcl (pressure, temperature, dewpt[, ...])	计算抬升凝结高度LCL
lfc (pressure, temperature, dewpt)	计算自由对流高度LFC
log_interp (x, xp, *args, **kwargs)	在指定轴上插值对数x尺度的数据。
mean_pressure_weighted (pressure, *args, **kwargs)	通过图层计算任意变量的压力加权平均值。

函数	说明
mixed_layer (p, *args, **kwargs)	Mix variable(s) over a layer, yielding a mass-weighted average.
mixed_parcel (p, temperature, dewpt[, ...])	Calculate the properties of a parcel mixed from a layer.
mixing_ratio (part_press, tot_press[, ...])	计算气体的混合比
mixing_ratio_from_relative_humidity (...)	根据相对湿度，温度和压力计算混合比
mixing_ratio_from_specific_humidity (...)	Calculate the mixing ratio from specific humidity.
moist_lapse (pressure, temperature)	Calculate the temperature at a level assuming liquid saturation processes.
moist_static_energy (heights, temperature, ...)	Calculate the moist static energy of parcels.
montgomery_streamfunction (height, temperature)	Compute the Montgomery Streamfunction on isentropic surfaces.
most_unstable_cape_cin (pressure, ...)	计算CAPE / CIN（最不稳定）
most_unstable_parcel (pressure, temperature, ...)	确定最不稳定层结高度
nearest_intersection_idx (a, b)	Determine the index of the point just before two lines with common x values.
parcel_profile (pressure, temperature, dewpt)	Calculate the profile a parcel takes through the atmosphere.
potential_temperature (pressure, temperature)	计算潜在温度
precipitable_water (dewpt, pressure[, ...])	通过探测深度计算可降水量
pressure_to_height_std (pressure)	Convert pressure data to heights using the U.S.
psychrometric_vapor_pressure_wet (...[, ...])	用湿球和干球温度计算蒸汽压力
reduce_point_density (points, radius[, priority])	Return a mask to reduce the density of points in irregularly-spaced data.
relative_humidity_from_dewpoint (temperature, ...)	计算相对湿度
relative_humidity_from_mixing_ratio (...)	根据混合比，温度和压力计算相对湿度。
relative_humidity_from_specific_humidity (...)	根据特定的湿度，温度和压力计算相对湿度。
relative_humidity_wet_psychrometric (...)	用湿球和干球温度计算相对湿度。

函数	说明
resample_nn_1d(a, centers)	Return one-dimensional nearest-neighbor indexes based on user-specified centers.
saturation_mixing_ratio(tot_press, temperature)	计算水蒸气的饱和混合比例
saturation_vapor_pressure(temperature)	计算饱和水蒸气（部分）压力
second_derivative(f, **kwargs)	计算网格上的的二阶导数
shearing_deformation(u, v, dx, dy)	计算水平风切变
shearing_stretching_deformation(u, v, dx, dy)	Calculate the horizontal shearing and stretching deformation of the horizontal wind.
sigma_to_pressure(sigma, psfc, ptop)	从sigma值计算压力
significant_tornado(sbcape, ...)	计算重要龙卷风参数（固定层）
specific_humidity_from_mixing_ratio(mixing_ratio)	Calculate the specific humidity from the mixing ratio.
storm_relative_helicity(u, v, heights, depth)	计算风暴相对螺旋度。
stretching_deformation(u, v, dx, dy)	Calculate the stretching deformation of the horizontal wind.
supercell_composite(mucape, ...)	Calculate the supercell composite parameter.
surface_based_cape_cin(pressure, ...)	计算基于表面的CAPE和CIN。
thickness_hydrostatic(pressure, temperature, ...)	Calculate the thickness of a layer via the hypsometric equation.
thickness_hydrostatic_from_relative_humidity(...)	Calculate the thickness of a layer given pressure, temperature and relative humidity.
tke(u, v, w[, perturbation, axis])	计算湍流动能。
total_deformation(u, v, dx, dy)	计算水平风的水平总变形
v_vorticity(u, v, dx, dy[, dim_order])	计算水平风的垂直涡度
vapor_pressure(pressure, mixing)	计算水汽（部分）压力
virtual_potential_temperature(pressure, ...)	计算虚拟潜在温度
virtual_temperature(temperature, mixing[, ...])	计算虚温

函数	说明
vorticity(u, v, dx, dy)	Calculate the vertical vorticity of the horizontal wind.
windchill(temperature, speed[, ...])	计算风冷温度指数（WCTI）

使用范例：

```
from metpy import calc
# 引用calc库函数
metpy.calc.add_height_to_pressure(pressure, height)
#计算高于另一个压力水平的特定高度的压力
```

17.6 plots 绘图模块

绘制气象图形

注意：这个模块需要配合Matplotlib 等常用包使用！！！

17.6.1 Functions

add_metpy_logo(fig[, x, y, zorder, size])	**Add the MetPy logo to a figure.**
add_timestamp(ax[, time, x, y, ha, ...])	添加时间轴
add_unidata_logo(fig[, x, y, zorder, size])	Add the Unidata logo to a figure.（商标添加）

17.6.2 Classes

Hodograph([ax, component_range])	**Make a hodograph of wind data.制作一个风力数据的记录。**
SkewT([fig, rotation, subplot])	Make Skew-T log-P plots of data. 绘制温度对数压力T-lnP图
StationPlot(ax, x, y[, fontsize, spacing, ...])	制作一个标准的气象台站图
StationPlotLayout	make a layout to encapsulate plotting using StationPlot.

如何使用?

首先先要引用库函数：

```
from metpy.plots import add_metpy_logo, Hodograph, SkewT
```

然后可以参考上述链接，里面自带了很多函数，直接引用即可。

17.7 ctables 模块

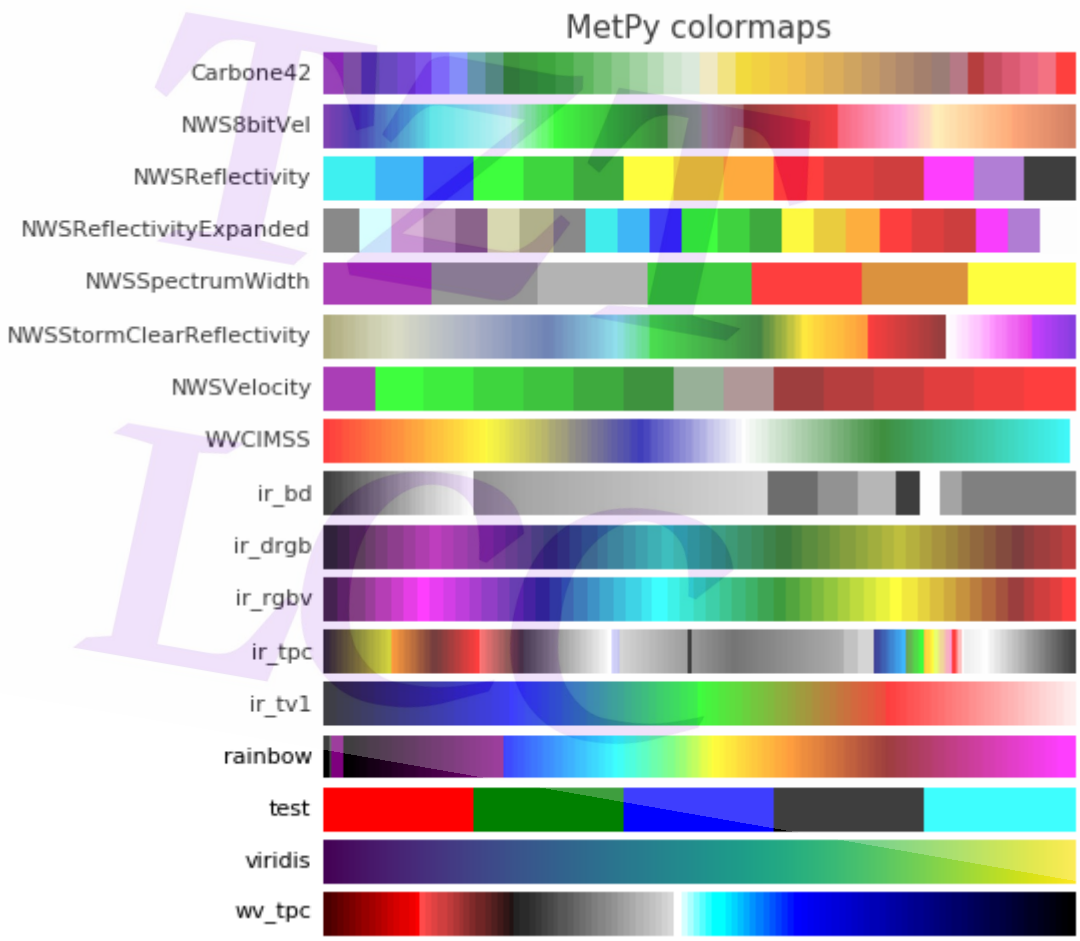
自定义颜色块

17.7.1 Functions

convert_gempak_table (infile, outfile)	**将GEMPAK颜色表转换为MetPy可以读取的颜色表。**
read_colortable (fobj)	Read colortable information from a file.

17.7.2 Classes

类	说明
ColortableRegistry	管理颜色表的集合。



官方已定义的颜色表

17.8 gridding 模块

提供用于将不规则间隔数据插入到规则网格中的工（站点数据插值到格点上）

17.8.1 Functions

`interpolate` **(x, y, z[, interp_type, hres, ...])**	**Interpolate given (x,y), observation (z) pairs to a grid based on given parameters. 高度上进行插值?**
`inverse_distance` (xp, yp, variable, grid_x, ...)	Generate an inverse distance weighting interpolation of the given points. 反距离加权插值
`natural_neighbor` (xp, yp, variable, grid_x, ...)	Generate a natural neighbor interpolation of the given points. 自然邻居插值

17.9 例子

官网上给出很多气象数据计算、绘图等例子

这里以绘制T-lnp图和对应的风玫瑰图为例简单讲解一下

17.9.1 Skew-T with Complex Layout

Combine a Skew-T and a hodograph using Matplotlib's *GridSpec* layout capability.

```
import matplotlib.gridspec as gridspec   #引用绘图包
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

import metpy.calc as mpcalc   #引用metpy包里面的各种模块：计算模块、绘图模块、单位转换模块
from metpy.cbook import get_test_data
from metpy.plots import add_metpy_logo, Hodograph, SkewT
from metpy.units import units
```

高空观测数据可以使用siphon package下载获得，但对于这个例子的数据则来自metpy包自带的样本数据。

```
col_names = ['pressure', 'height', 'temperature', 'dewpoint', 'direction', 'speed']

df = pd.read_fwf(get_test_data('may4_sounding.txt', as_file_obj=False),
                 skiprows=5, usecols=[0, 1, 2, 3, 6, 7], names=col_names)

df['u_wind'], df['v_wind'] = mpcalc.get_wind_components(df['speed'],
                                                        np.deg2rad(df['direction']))

# Drop any rows with all NaN values for T, Td, winds 对缺测值数据进行处理
df = df.dropna(subset=('temperature', 'dewpoint', 'direction', 'speed',
                       'u_wind', 'v_wind'), how='all').reset_index(drop=True)
```

We will pull the data out of the example dataset into individual variables and assign units.

```
p = df['pressure'].values * units.hPa

T = df['temperature'].values * units.degC
```

```
Td = df['dewpoint'].values * units.degC
wind_speed = df['speed'].values * units.knots
wind_dir = df['direction'].values * units.degrees
u, v = mpcalc.get_wind_components(wind_speed, wind_dir)
# Create a new figure. The dimensions here give a good aspect ratio
fig = plt.figure(figsize=(9, 9))
add_metpy_logo(fig, 630, 80, size='large')

# Grid for plots 绘制格点
gs = gridspec.GridSpec(3, 3)
skew = SkewT(fig, rotation=45, subplot=gs[:, :2])

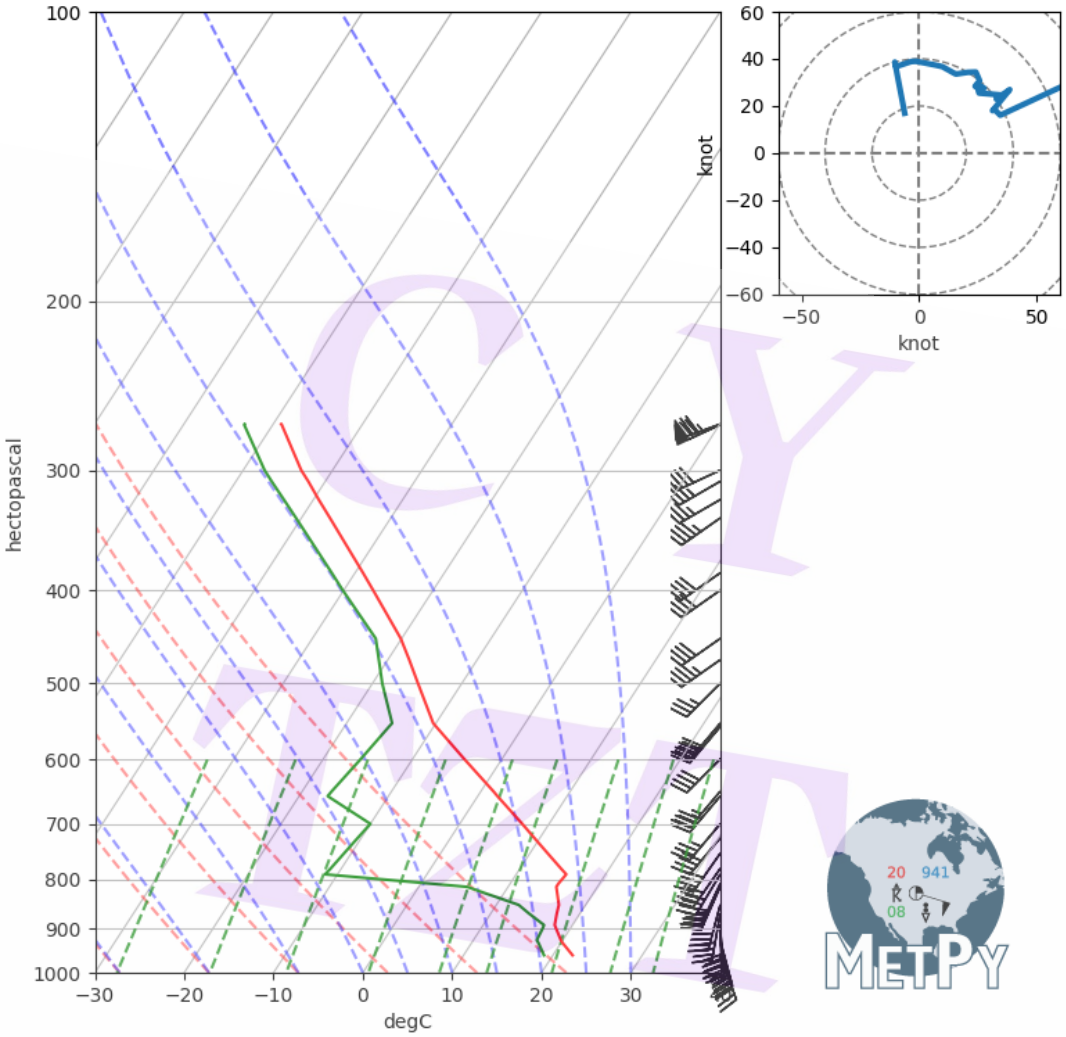
# Plot the data using normal plotting functions, in this case using
# log scaling in Y, as dictated by the typical meteorological plot
skew.plot(p, T, 'r')  #绘制大气温度曲线
skew.plot(p, Td, 'g')  #绘制大气露点温度曲线
skew.plot_barbs(p, u, v)  #绘制风羽
skew.ax.set_ylim(1000, 100)

# Add the relevant special lines
skew.plot_dry_adiabats()  #绘制底图干绝热线
skew.plot_moist_adiabats()  #绘制底图湿绝热线
skew.plot_mixing_lines()  #绘制底图等饱和比湿热线

# Good bounds for aspect ratio
skew.ax.set_xlim(-30, 40)

# Create a hodograph
ax = fig.add_subplot(gs[0, -1])
h = Hodograph(ax, component_range=60.)
h.add_grid(increment=20)
h.plot(u, v)

# Show the plot 出图
plt.show()
```


100
200
300
400
500
600
700
800
900
1000
hectopascal
−30
−20
−10
0
10
20
30
degC
60
40
20
0
−20
−40
−60
knot
−50
0
50
knot
20 941
08
METPY