km3astro

Release

Nov 02, 2017

Contents

1 Install **3**

2 Contents **5**
2.1 Coordinates 5
2.2 Example Gallery 6
2.3 API Reference 12

Python Module Index **17**

KM3Astro is a collection astronomy utils, like coordinate transformations.

CHAPTER 1

Install

This is developed on python 3.6. Lower versions (especially py2) might or might not work.

Install the dependencies

- numpy
- pandas
- astropy

In your python env, do

```
pip install km3astro
```

CHAPTER 2

Contents

2.1 Coordinates

2.1.1 Antares

see http://antares.in2p3.fr/internal/dokuwiki/doku.php?id=astro_coordinatetransformation_howto

2.1.2 Conventions + Definitions

Neutrino direction vs Source Direction

We usually talk about the direction a neutrino is *going to*. When determining which source it came from, we obviously need to invert the direction.

Neutrino direction, polar coordinates: *theta* and *phi*

Neutrino source: *zenith* and *azimuth*

Azimuth Definition

Our definition of the azimuth angle differs from the more common definition that is used in SLALIB, SeaTray's astro package, and astropy (used in km3astro).

```
true_azimuth = (90deg - azimuth) mod 360deg
```

Equatorial Coordinates

Equatorial coordinates are ICRS, i.e. J2000 Epoch, J2000 Equinox, Barycentric (barycenter of Solar system as coordinate origin).

If you'd rather have geocentric equatorial coordinates, use GCRS.

If you are looking for "J2000 Equatorial coordinates" you probably want ICRS and not FK5.

For more details on ICRS, see https://en.wikipedia.org/wiki/International_Celestial_Reference_System and maybe http://docs.astropy.org/en/stable/api/astropy.coordinates.ICRS.html#astropy.coordinates.ICRS .

UTM Grid

The UTM origin is noted at the head of the .detx v2 files: Dataformats#Detector_Description_.28.detx.29

A snapshot of these (July 2017) can be found at `km3astro.constants`.

2.1.3 Benchmarks

Antares `astro` benchmarks: http://antares.in2p3.fr/internal/dokuwiki/doku.php?id=benchmarks_astro

(Partial) replication with `km3astro`: *Benchmark Calculations for Coordinates Transformations*

2.2 Example Gallery

orphan

2.2.1 Sun in local coordinates

Show off some coordinate transformations.

```
# Author: Moritz Lotze <mlotze@km3net.de>
# License: BSD-3

from astropy.units import deg
import numpy as np
import pandas as pd

from km3astro.random import random_date, random_azimuth, random_zenith
from km3astro.coord import local_frame, Sun, source_to_neutrino_direction
```

generate some random events

```
n_evts = 1e4
zen = random_zenith(n=n_evts)
time = random_date(n=n_evts)
azi = random_azimuth(n=n_evts)
```

transform to horizontal coordinates

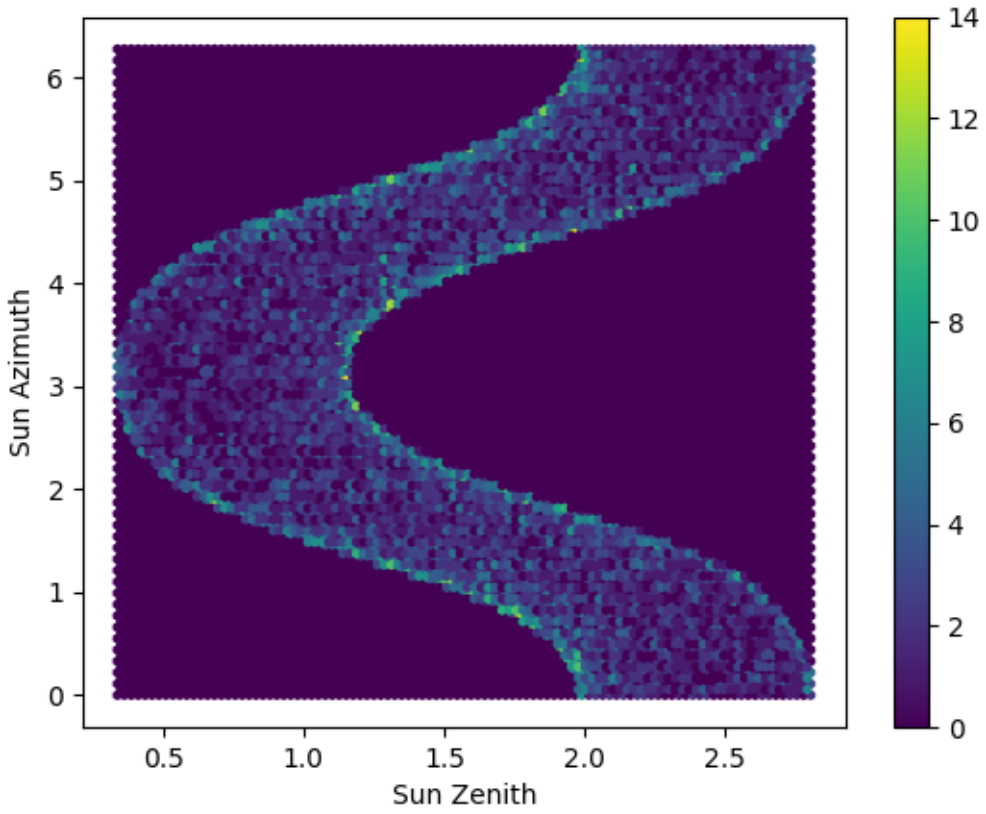
```
orca_frame = local_frame(time=time, location='orca')
sun = Sun(time)

sun_orca = sun.transform_to(orca_frame)

sun_azi = sun_orca.az.rad
sun_zen = (90 * deg - sun_orca.alt).rad

sun_phi, sun_theta = source_to_neutrino_direction(sun_azi, sun_zen)
```

```
sun_df = pd.DataFrame({
    'Sun Azimuth': sun_azi,
    'Sun Zenith': sun_zen,
    'Sun Cos Zenith': np.cos(sun_zen),
    'Sun Phi': sun_phi,
    'Sun Theta': sun_theta,
    'Sun Cos Theta': np.cos(sun_theta),
})
```

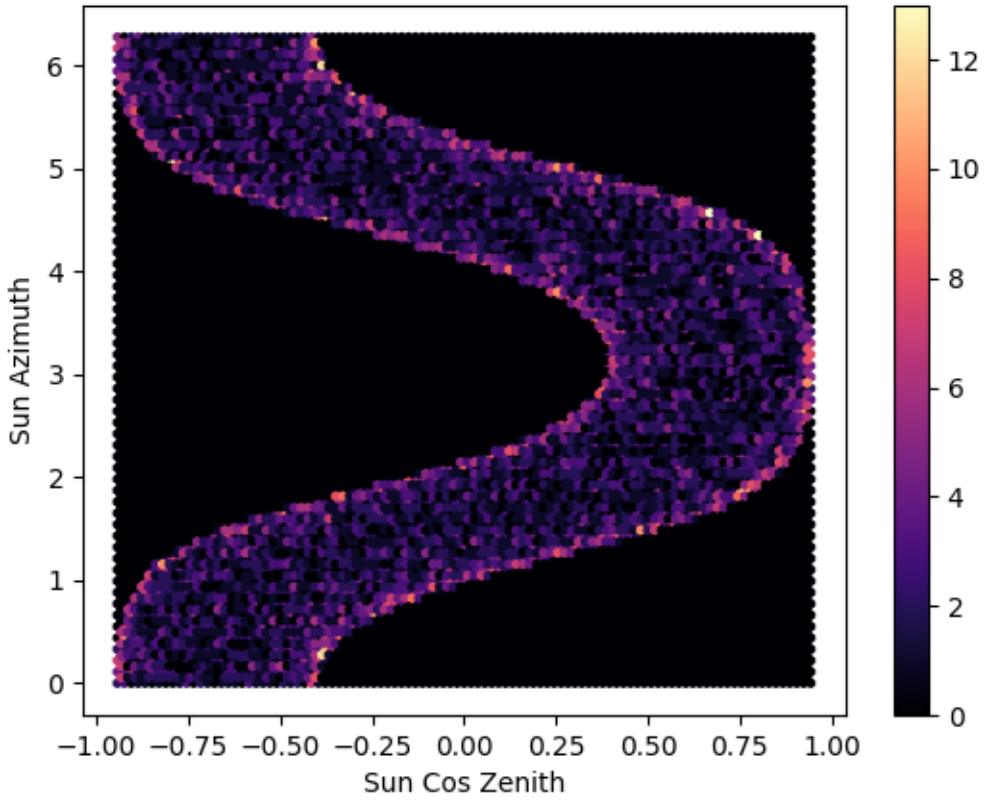
Out:

```
Downloading http://maia.usno.navy.mil/ser7/finals2000A.all [Done]
```

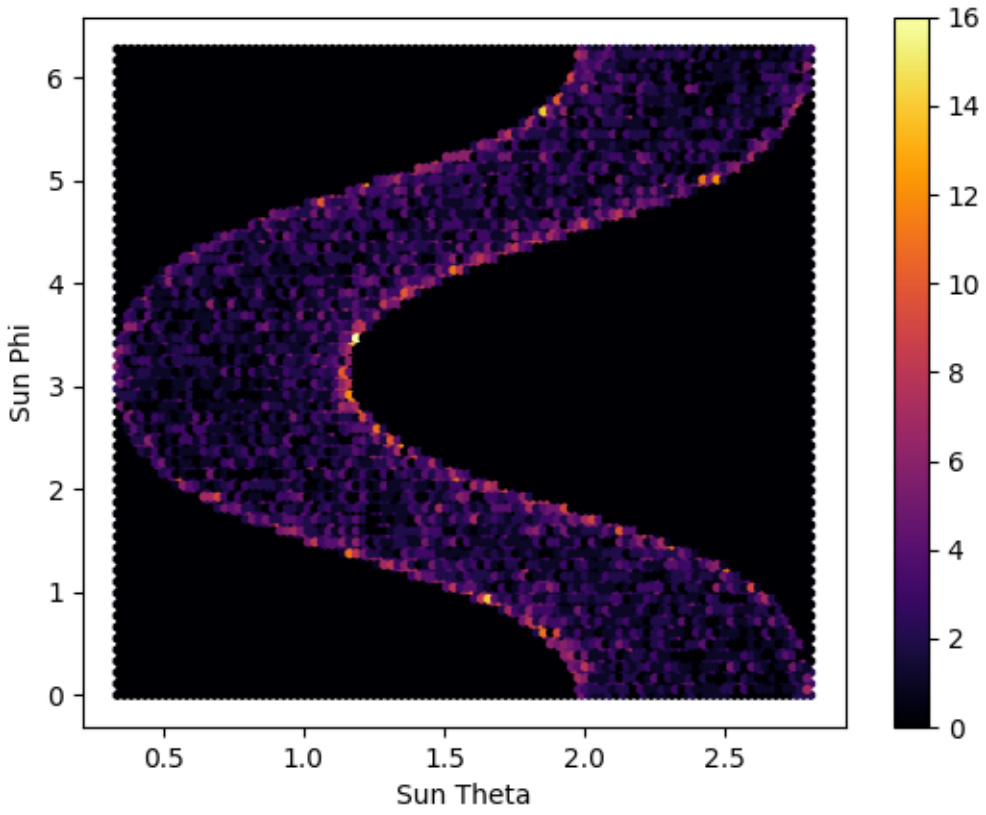
```
sun_df.plot.hexbin(
    'Sun Zenith',
    'Sun Azimuth',
    cmap='viridis')
```



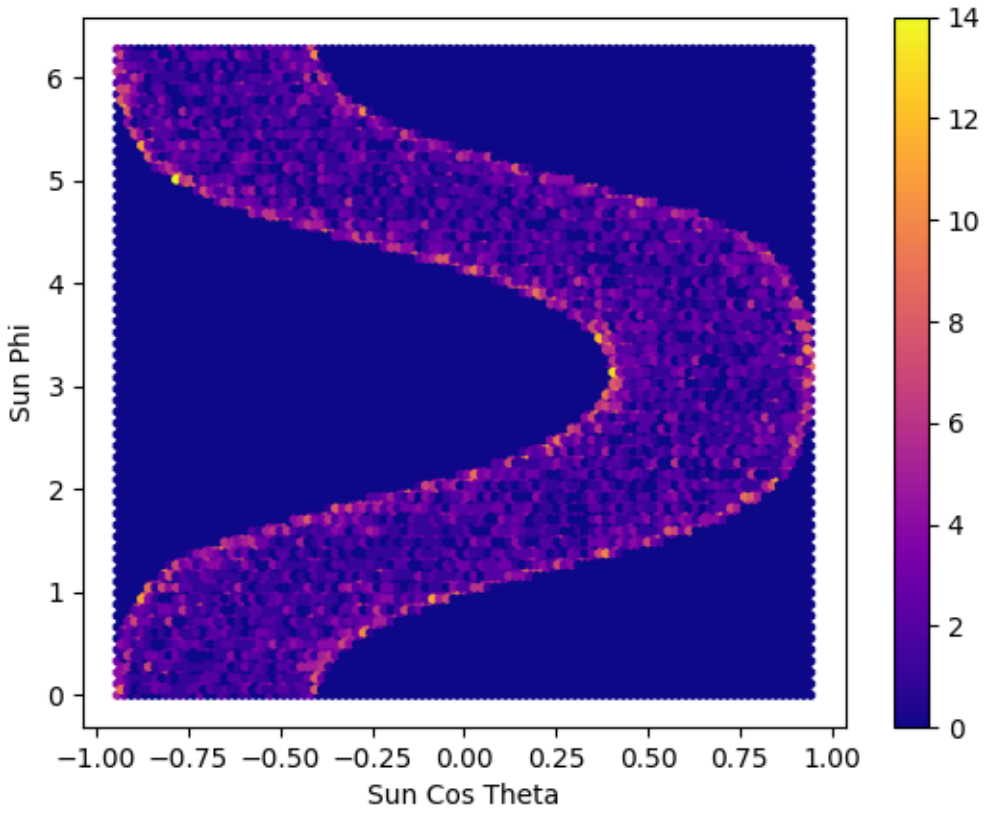

```
sun_df.plot.hexbin(
    'Sun Cos Zenith',
    'Sun Azimuth',
    cmap='magma')
```




```
sun_df.plot.hexbin(
    'Sun Theta',
    'Sun Phi',
    cmap='inferno')
```




```
sun_df.plot.hexbin(
    'Sun Cos Theta',
    'Sun Phi',
    cmap='plasma')
```



Total running time of the script: (0 minutes 14.478 seconds)

2.2.2 Benchmark Calculations for Coordinates Transformations

Run some benchmarks similar to *http://antares.in2p3.fr/internal/dokuwiki/doku.php?id=benchmarks_astro*

CAVEAT: IceCube and ANTARES/KM3NeT define "azimuth" differently:

`true_azimuth = (90 - azimuth) % 360!`

```
# Author: Moritz Lotze <mlotze@km3net.de>
# License: BSD-3

from collections import OrderedDict

import numpy as np
import pandas as pd

from km3astro.coord import (neutrino_to_source_direction,
                            local_event, local_frame)
from km3astro.sources import SIRIUS, CANOPUS, ARCTURUS, ANTARES      # noqa


time = pd.to_datetime([
    '2007-10-04 03:03:03.00',
    '2007-10-04 03:03:03.00',
```

```
    '2007-10-04 03:03:03.00',
    '2007-10-04 03:03:03.00',
    '2007-10-04 03:03:03.00',
])
theta_deg = np.array([
    135.00,
    11.97,
    22.97,
    33.97,
    85.23,
])
phi_deg = np.array([
    97.07,
    23.46,
    97.07,
    192.50,
    333.33,
])
theta = theta_deg * np.pi / 180
phi = phi_deg * np.pi / 180
azimuth, zenith = neutrino_to_source_direction(phi, theta)

zenith = np.pi - zenith
azimuth = np.pi + azimuth

evt = local_event(azimuth, time, zenith, location='antares')
equat = evt.fk5
#dec = equat.dec.degree
#ra = equat.ra.degree
dec = equat.dec
ra = equat.ra
gal = evt.galactic
l = gal.l
b = gal.b

data = pd.DataFrame(OrderedDict([
    ('time', time),
    ('theta [deg]', theta_deg),
    ('phi [deg]', phi_deg),
    ('theta', theta),
    ('phi', phi),
    ('zenith [deg]', zenith * 180 / np.pi),
    ('azimuth [deg]', azimuth * 180 / np.pi),
    ('zenith', zenith),
    ('azimuth', azimuth),
    ('declication', dec),
    ('right ascension', ra),
    ('gal_longitude', l),
    ('gal_latitude', b),
]))
print(data[:])
```

Out:

```
time  theta [deg]  phi [deg]     theta       phi  \
0 2007-10-04 03:03:03       135.00      97.07  2.356194  1.694191
1 2007-10-04 03:03:03        11.97      23.46  0.208916  0.409454
2 2007-10-04 03:03:03        22.97      97.07  0.400902  1.694191
```

```
3 2007-10-04 03:03:03        33.97     192.50  0.592888  3.359759
4 2007-10-04 03:03:03        85.23     333.33  1.487544  5.817706

   zenith [deg]  azimuth [deg]    zenith   azimuth  declication  \
0        135.00         457.07  2.356194  7.977376    -1.995039
1         11.97         383.46  0.208916  6.692640   -37.132066
2         22.97         457.07  0.400902  7.977376   -65.438763
3         33.97         192.50  0.592888  3.359759   -28.302860
4         85.23         333.33  1.487544  5.817706   -22.597218

   right ascension  gal_longitude  gal_latitude
0         69.254453     198.047421    -30.540847
1        230.429014     333.427330     16.686478
2        237.548107     320.133958     -8.726749
3        205.954480     316.717830     33.162808
4        139.019698     251.312426     18.035496
```

look at some sources in horizontal coordinates

```
frame = local_frame(
    time=pd.to_datetime(['2007-10-04 03:03:03.00', ]), location='antares')
sirius_local = SIRIUS.transform_to(frame)
sirius_alt = sirius_local.alt.degree
sirius_az = sirius_local.az.degree
sirius_zen = 90 - sirius_alt
print()
print(sirius_az)
print(sirius_zen)

canopus_local = CANOPUS.transform_to(frame)
canopus_alt = canopus_local.alt.degree
canopus_az = canopus_local.az.degree
canopus_zen = 90 - canopus_alt
print()
print(canopus_az)
print(canopus_zen)
```

Out:

```
[ 141.72305448]
[ 68.5514653]

[ 161.18692266]
[ 99.31872154]
```

Total running time of the script: (0 minutes 0.405 seconds)

2.3 API Reference

- *km3astro.constants*
- *km3astro.coord*
- *km3astro.time*

- *km3astro.time*
- *km3astro.sources*

2.3.1 km3astro.constants

Constants, like geographical positions.

—

2.3.2 km3astro.coord

Coordinate transformations.

Galactic: GC at (0, 0), gal. longitude, latitude (l, b)

Horizontal / altaz (km3): centered at detector position altitude, azimuth (altitude = 90deg - zenith)

EquatorialJ200 / FK5 / ICRS / GCRS (right ascension, declination)

> Equatorial is the same as FK5. FK5 is superseded by the ICRS, so use this instead. Note that FK5/ICRS are _barycentric_ implementations, so if you are looking for *geocentric* equatorial (i.e. for solar system bodies), use GCRS.

A note on maing conventions: `phi` and `theta` refer to neutrino directions, `azimuth` and `zenith` to source directions (i.e. the inversed neutrino direction). The former says where the neutrino points to, the latter says where it comes from.

Also radian is the default. Degree can be used, but generally the default is to assume radian.

get_location([location])	
neutrino_to_source_direction(phi, theta[, ...])	Flip the direction.
source_to_neutrino_direction(azimuth, zenith)	
Sun(time)	
local_frame(time[, location])	Get the (horizontal) coordinate frame of your detector.
local_event(azimuth, time, zenith[, radian, ...])	Create astropy events from detector coordinates.
sun_local(time[, loc])	
gc_in_local(time[, loc])	
orca_gc_dist(azimuth, time, zenith[, frame])	Return angular distance of event to GC.
orca_sun_dist(azimuth, time, zenith)	Return distance of event to sun, in detector coordinates.
sun_dist_random(zenith)	Generate random (time, azimuth) events and get distance to GC.
gc_dist_random(zenith[, frame])	Generate random (time, azimuth) events and get distance to GC.

km3astro.coord.get_location

km3astro.coord.**get_location**(*location='orca'*)

km3astro.coord.neutrino_to_source_direction

km3astro.coord.**neutrino_to_source_direction**(*phi*, *theta*, *radian=True*)
Flip the direction.

Parameters **phi, theta: neutrino direction**

radian: bool [default=True]

receive + return angles in radian? (if false, use degree)

km3astro.coord.source_to_neutrino_direction

km3astro.coord.**source_to_neutrino_direction**(*azimuth*, *zenith*, *radian=True*)

km3astro.coord.Sun

km3astro.coord.**Sun**(*time*)

km3astro.coord.local_frame

km3astro.coord.**local_frame**(*time*, *location='orca'*)
Get the (horizontal) coordinate frame of your detector.

km3astro.coord.local_event

km3astro.coord.**local_event**(*azimuth*, *time*, *zenith*, *radian=True*, *location='orca'*, ***kwargs*)
Create astropy events from detector coordinates.

km3astro.coord.sun_local

km3astro.coord.**sun_local**(*time*, *loc='orca'*)

km3astro.coord.gc_in_local

km3astro.coord.**gc_in_local**(*time*, *loc='orca'*)

km3astro.coord.orca_gc_dist

km3astro.coord.**orca_gc_dist**(*azimuth*, *time*, *zenith*, *frame='detector'*)
Return angular distance of event to GC.

Parameters **frame: str, [default: 'detector']**

valid are 'detector', 'galactic', 'icrs', 'gcrs'

km3astro.coord.orca_sun_dist

km3astro.coord.**orca_sun_dist**(*azimuth*, *time*, *zenith*)
Return distance of event to sun, in detector coordinates.

km3astro.coord.sun_dist_random

km3astro.coord.**sun_dist_random**(*zenith*)
Generate random (time, azimuth) events and get distance to GC.

km3astro.coord.gc_dist_random

km3astro.coord.**gc_dist_random**(*zenith*, *frame='detector'*)
Generate random (time, azimuth) events and get distance to GC.

2.3.3 km3astro.time

Time conversion utilities.

For random time samples, goto *km3flux.random*

np_to_astrotime(intime)

km3astro.time.np_to_astrotime

km3astro.time.**np_to_astrotime**(*intime*)

2.3.4 km3astro.time

Time conversion utilities.

For random time samples, goto *km3flux.random*

random_azimuth
random_zenith
second_from_interval
equidistant_from_interval
random_date

2.3.5 km3astro.sources

Time conversion utilities.

For random time samples, goto *km3flux.random*

SIRIUS
CANOPUS
ARCTURUS
ANTARES
RX_J1713
VELA_X
SAGITTARIUS_A_STAR
GALACTIC_CENTER

- genindex

Python Module Index

k

km3astro.constants, 13
km3astro.coord, 13
km3astro.time, 15

Index

G

gc_dist_random() (in module km3astro.coord), 15
gc_in_local() (in module km3astro.coord), 14
get_location() (in module km3astro.coord), 13

K

km3astro.constants (module), 13
km3astro.coord (module), 13
km3astro.time (module), 15

L

local_event() (in module km3astro.coord), 14
local_frame() (in module km3astro.coord), 14

N

neutrino_to_source_direction() (in module km3astro.coord), 14
np_to_astrotime() (in module km3astro.time), 15

O

orca_gc_dist() (in module km3astro.coord), 14
orca_sun_dist() (in module km3astro.coord), 14

S

source_to_neutrino_direction() (in module km3astro.coord), 14
Sun() (in module km3astro.coord), 14
sun_dist_random() (in module km3astro.coord), 15
sun_local() (in module km3astro.coord), 14