Curso básico de Python para estudantes de Física

Germán A. Racca

Universidade do Estado do Rio Grande do Norte
Faculdade de Ciências Exatas e Naturais
Departamento de Física
Mossoró - RN

30 de Junho de 2016

Tópicos

1. SciPy
   - Estatística
   - Integração numérica
2. Mais sobre Python
   - Compreensão de lista
   - Indexação de *arrays*
3. Astropy
   - Quantidades
   - Coordenadas
   - Modelagem e ajuste
   - Imagens FITS
   - Informação de catálogo
   - Pacote afiliado: Astroquery
4. Material do curso

Tópicos

Tópicos

Estatística

- Estatística descritiva:

```
>>> import numpy as np
>>> from scipy import stats

>>> amostra = np.random.randn(100)
>>> n, min_max, mean, var, skew, kurt = stats.describe(amostra)

>>> print "Numero de elementos:", n
Numero de elementos: 100

>>> print "Minimo:", min_max[0], "Maximo:", min_max[1]
Minimo: -2.46105281326 Maximo: 2.29497467683

>>> print "Media:", mean
Media: -0.0127357965463

>>> print "Variancia:", var
Variancia: 0.97951664897

>>> print "Obliquidade:", skew
Obliquidade: -0.211993018852

>>> print "Curtose:", kurt
Curtose: -0.240283872736
```

```
>>> stats.describe(amostra)
(100,
 (-2.3014411282665841, 2.4979711279843966),
 0.19241517466875757,
 1.0640842582744734,
 -0.15801757606178518,
 -0.3811230889237791)
```

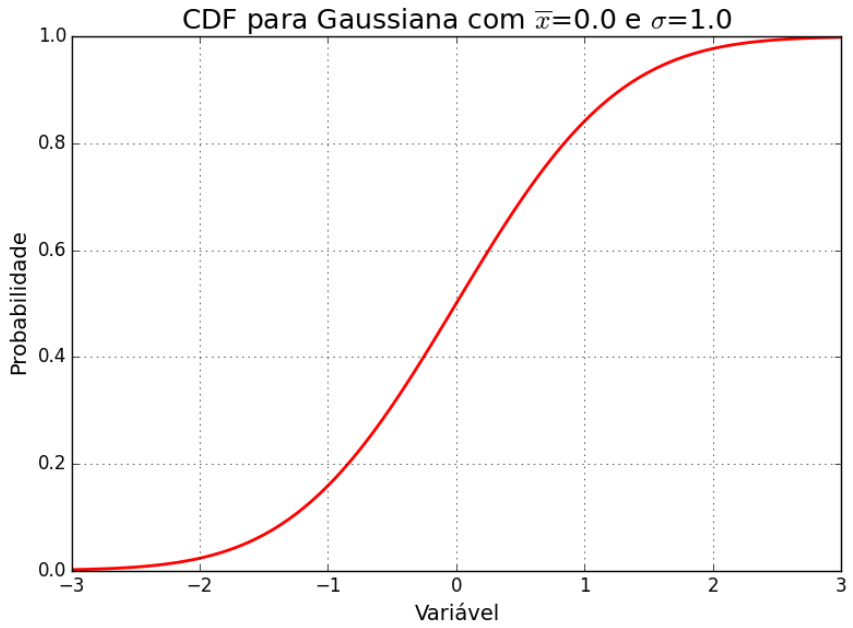
Estatística

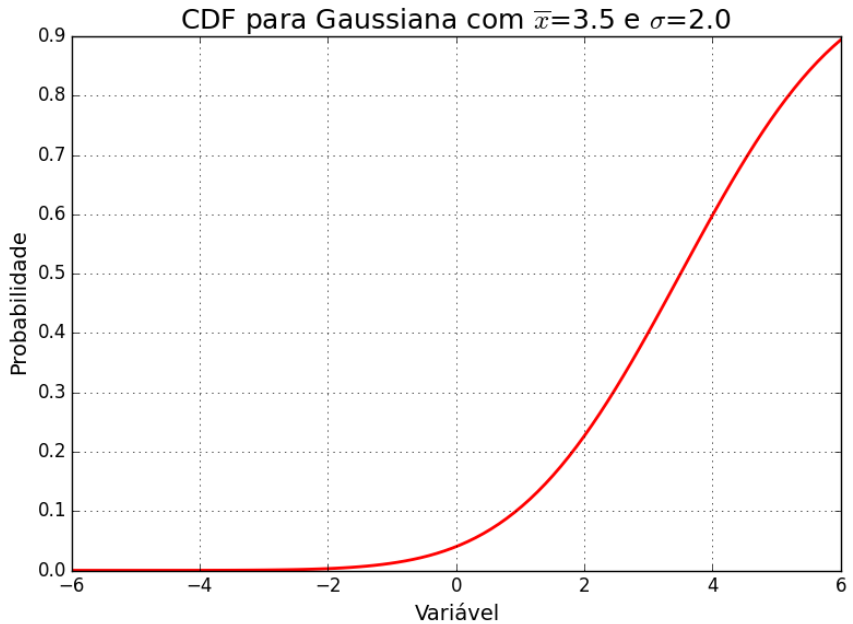
- Distribuições de probabilidade:

```
>>> import numpy as np
>>> from scipy import stats

# loc = x̄, scale = σ
>>> n = stats.norm(loc=3.5, scale=2.0)
>>> n.rvs()
4.578672796257988

# seleciona número aleatorio da Gaussiana
>>> stats.norm.rvs(loc=3.5, scale=2.0)
3.1156690832878935

# PDF (função densidade de probabilidade) para variável continua
>>> stats.norm.pdf([-0.1, 0.0, 0.1], loc=3.5, scale=2.0)
array([ 0.03947508,  0.04313866,  0.04702454])

# PMF (função massa de probabilidade) para variável discreta
>>> k = np.arange(5)
>>> n = 10
>>> p = 0.5
>>> stats.binom.pmf(k, n, p)
array([ 0.00097656,  0.00976563,  0.04394531,  0.1171875 ,  0.20507813])
```

Estatística

```
>>> import numpy as np
>>> from scipy import stats
>>> import matplotlib.pyplot as plt

# CDF (função acumulada de probabilidade)
# para uma distribuição Gaussiana
>>> def norm_cdf(media=0.0, sigma=1.0):
...     x = np.linspace(-3*sigma, 3*sigma, 100)
...     y = stats.norm.cdf(x, loc=media, scale=sigma)
...     plt.plot(x, y, "r-", lw=2)
...     plt.xlabel("Variavel", fontsize=14)
...     plt.ylabel("Probabilidade", fontsize=14)
...     texto = "CDF para Gaussiana com " + \
...     r"$\overline{x}$=" + str(media) + \
...     r" e $\sigma$=" + str(sigma)
...     plt.title(texto, fontsize=18)
...     plt.grid()
...     plt.tight_layout()
...     plt.show()

>>> norm_cdf()
>>> norm_cdf(3.5, 2.0)
```





Tópicos

Integração numérica

- A rotina mais genérica para integração de funções é `integrate.quad`:

```
>>> import numpy as np
>>> from scipy import integrate
>>> res, err = integrate.quad(np.sin, 0, np.pi/2)
>>> res
0.9999999999999999
>>> err
1.1102230246251564e-14
```

- EDOs da forma $\mathrm{dy/dt} = \mathrm{rhs}(\mathrm{y_1}, \mathrm{y_2}, ..., \mathrm{t_0})$ com `integrate.odeint`:

```
# dy/dt = -2y, t entre 0 e 4, y(t=0) = 1
>>> def deriv(y, t):
...     return -2*y
>>> t = np.linspace(0, 4, 10)
>>> yvec, info = integrate.odeint(deriv, 1, t, full_output=True)
>>> yvec.T
array([[  1.00000000e+00,   4.11112313e-01,   1.69013313e-01,
          6.94834400e-02,   2.85654998e-02,   1.17436289e-02,
          4.82795070e-03,   1.98483034e-03,   8.15986861e-04,
          3.35461690e-04]])
>>> info['nfe'][-1]
129
```

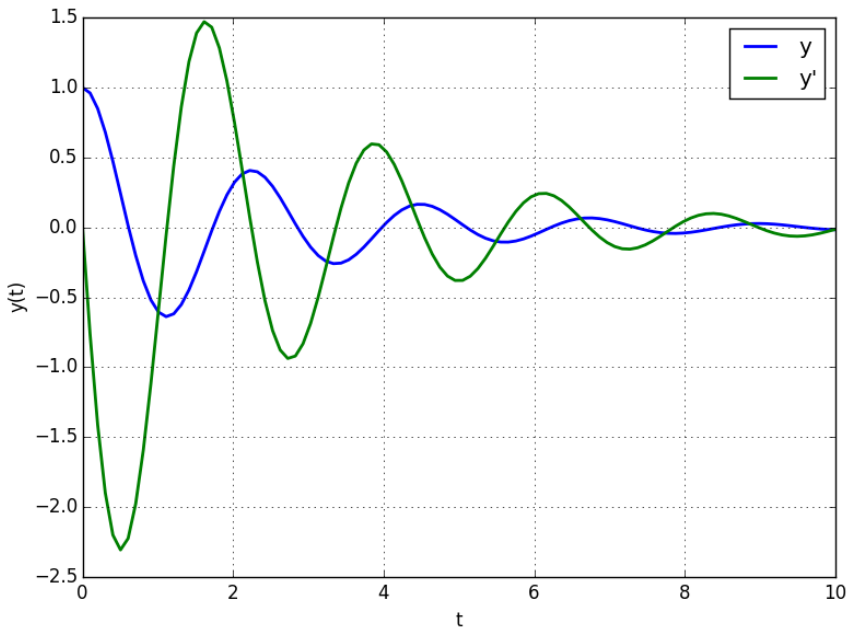
Integração numérica

- Resolvemos a equação do oscilador massa-mola amortecido:

$$\frac{\mathrm{d}^2\mathrm{y}}{\mathrm{dt}^2} + 2\epsilon\omega_0\frac{\mathrm{dy}}{\mathrm{dt}} + \omega_0^2\mathrm{y} = 0$$

$$\omega_0^2 = \mathrm{k/m}, \epsilon = \mathrm{c}/(2\mathrm{m}\omega_0) \Rightarrow \mathrm{nu} = 2\epsilon\omega_0 = \mathrm{c/m}, \mathrm{om} = \omega_0^2 = \mathrm{k/m}$$

```
>>> import numpy as np
>>> from scipy.integrate import odeint
>>> import matplotlib.pyplot as plt

>>> m, k, c = 0.5, 4, 0.4
>>> nu, om = c/m, k/m
# yvec = (y, y')
>>> def deriv(yvec, t, nu, om):
...     return (yvec[1], -nu*yvec[1] - om*yvec[0])
>>> t = np.linspace(0, 10, 100)
>>> yarr = odeint(deriv, (1, 0), t, args=(nu, om))

>>> plt.plot(t, yarr[:, 0], lw=2, label="y")
>>> plt.plot(t, yarr[:, 1], lw=2, label="y'")
>>> plt.xlabel("t")
>>> plt.ylabel("y(t)")
>>> plt.legend()
>>> plt.grid()
>>> plt.show()
```



Tópicos

Tópicos

Compreensão de lista

```
# Este loop...
>>> a = []
>>> for i in range(5):
...     a.append(i*10)
>>> a
[0, 10, 20, 30, 40]

# é equivalente a:
>>> a = [i*10 for i in range(5)]
>>> a
[0, 10, 20, 30, 40]

# Este loop com condição...
>>> a = []
>>> for i in range(5):
...     if i % 2 == 0:
...         a.append(i*10)
>>> a
[0, 20, 40]

# é equivalente a:
>>> a = [i*10 for i in range(5) if i % 2 == 0]
>>> a
[0, 20, 40]
```

Tópicos

Indexação de *arrays*

- Indexação booleana (máscaras):

```
>>> import numpy as np
>>> A = np.array([4, 7, 3, 4, 2, 8])
>>> A < 5
array([ True, False,  True,  True,  True, False], dtype=bool)
```

- Indexação sofisticada:

```
>>> C = np.array([123, 188, 190, 99, 77, 88, 100])
>>> A = np.array([4, 7, 2, 8, 6, 9, 5])
>>> R = C[A <= 5]
>>> R
array([123, 190, 100])

>>> C[C >= 100]
array([123, 188, 190, 100])
```

- Indexação com *array* de inteiros:

```
>>> C[[0, 2, 3, 1, 4, 1]]
array([123, 190,  99, 188,  77, 188])
```

Tópicos

Astropy

Astropy

- Pacote open source para Astronomia de desenvolvimento comunitário
- O núcleo do pacote contém várias classes e utilidades, dividido em sub-pacotes
- O projeto inclui "pacotes afiliados" (pacotes de Astronomia para Python)

Como importar Astropy:

```
>>> from astropy import sub_pacote

>>> from astropy.io import fits
>>> hdulist = fits.open('imagem.fits')
```

As vezes usamos apelidos:

```
>>> from astropy import units as u
>>> from astropy import coordinates as coord
```

Ou importamos a classe diretamente:

```
>>> from astropy.cosmology import WMAP7
>>> from astropy.table import Table
>>> from astropy.wcs import WCS
```

Astropy

Sub-pacote	Descrição
`astropy.constants`	Constantes físicas para Astronomia
`astropy.units`	Unidades e quantidades
`astropy.nddata`	Conjunto de dados multi-dimensionais
`astropy.table`	Tabelas de dados
`astropy.time`	Tempo e datas
`astropy.coordinates`	Sistemas de coordenadas astronômicos
`astropy.wcs`	Sistemas de coordenadas globais
`astropy.modeling`	Modelos e ajustes
`astropy.analytic_functions`	Funções analíticas
`astropy.io`	Leitura e escritura de dados
`astropy.cosmology`	Cálculos cosmológicos
`astropy.convolution`	Convolução e filtragem
`astropy.visualization`	Visualização de dados
`astropy.stats`	Ferramentas de estatística

Tópicos

Quantidades

- Una quantidade é a combinação de um valor e uma unidade:

```
>>> from astropy import units as u
>>> q = 42.0 * u.meter
>>> q
<Quantity 42.0 m>
>>> q.value
42.0
>>> q.unit
Unit("m")

>>> import numpy as np
>>> np.array([1., 2., 3.]) * u.m
<Quantity [ 1., 2., 3.] m>

>>> q = 3.0 * u.kilometer / (130.51 * u.meter / u.second)
>>> print q
0.0229867443108 km s / m
>>> q.decompose()
<Quantity 22.986744310780782 s>

>>> x = 1.0 * u.parsec
>>> print x.to(u.km)
3.08567758147e+13 km
```

Tópicos

Coordenadas

- Representação de coordenadas celestes e espaciais:

```
>>> from astropy import units as u
>>> from astropy.coordinates import SkyCoord
>>> c = SkyCoord(ra=10.625*u.degree, dec=41.2*u.degree, frame='icrs')
>>> c = SkyCoord(10.625, 41.2, frame='icrs', unit='deg')
>>> c = SkyCoord('00h42m30s', '+41d12m00s')
>>> 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))
>>> c
<SkyCoord (ICRS): (ra, dec) in deg
    (10.625, 41.2)>

>>> c = SkyCoord(ra=10.68458*u.degree, dec=41.26917*u.degree)
>>> c.ra
<Longitude 10.68458 deg>
>>> c.ra.hour
0.7123053333333335
>>> c.ra.hms
hms_tuple(h=0.0, m=42.0, s=44.299200000000525)
>>> c.dec.degree
41.26917
>>> c.dec.radian
0.7202828960652683
```

Coordenadas

- Transformação a um sistema de coordenadas diferente:

```
>>> c_icrs = SkyCoord(ra=10.68458*u.degree, dec=41.26917*u.degree, frame='icrs')
>>> c_icrs.galactic
<SkyCoord (Galactic): (l, b) in deg
    (121.17424181, -21.57288557)>

>>> c_fk5 = c_icrs.transform_to('fk5')
>>> 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'))
<SkyCoord (FK5: equinox=J1975.000): (ra, dec) in deg
    (10.34209135, 41.13232112)>
```

- Mudando a representação das coordenadas:

```
>>> c = SkyCoord(x=1, y=2, z=3, unit='kpc', representation='cartesian')
>>> c
<SkyCoord (ICRS): (x, y, z) in kpc
    (1.0, 2.0, 3.0)>
>>> c.x, c.y, c.z
(<Quantity 1.0 kpc>, <Quantity 2.0 kpc>, <Quantity 3.0 kpc>)
```

Coordenadas

- Assignando distâncias às coordenadas:

```
>>> from astropy.coordinates import Distance
>>> c = SkyCoord(ra=10.68458*u.degree, dec=41.26917*u.degree, distance=770*u.kpc)
>>> c.cartesian.x
<Quantity 568.7128654235232 kpc>
>>> c.cartesian.y
<Quantity 107.3008974042025 kpc>
>>> c.cartesian.z
<Quantity 507.88994291875713 kpc>

>>> c1 = SkyCoord(ra=10*u.degree, dec=9*u.degree, distance=10*u.pc)
>>> c2 = SkyCoord(ra=11*u.degree, dec=10*u.degree, distance=11.5*u.pc)
>>> c1.separation_3d(c2)
<Distance 1.5228602415117989 pc>
```

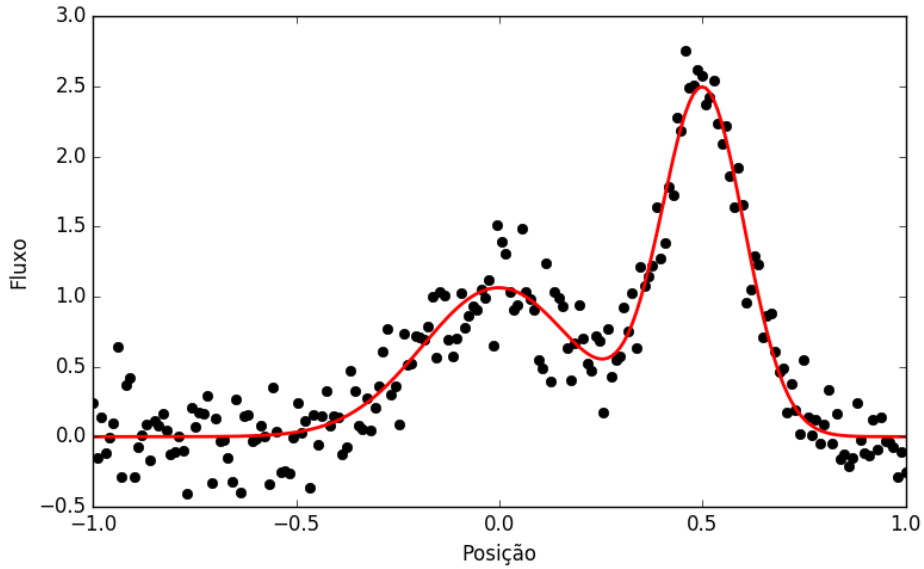
- Separação angular e busca em catálogo:

```
>>> c1 = SkyCoord(ra=10*u.degree, dec=9*u.degree, frame='icrs')
>>> c2 = SkyCoord(ra=11*u.degree, dec=10*u.degree, frame='fk5')
>>> c1.separation(c2)
<Angle 1.4045335865905868 deg>

>>> SkyCoord.from_name("M42")
<SkyCoord (ICRS): (ra, dec) in deg
    (83.82208, -5.39111)>
```

Tópicos

Modelagem e ajuste

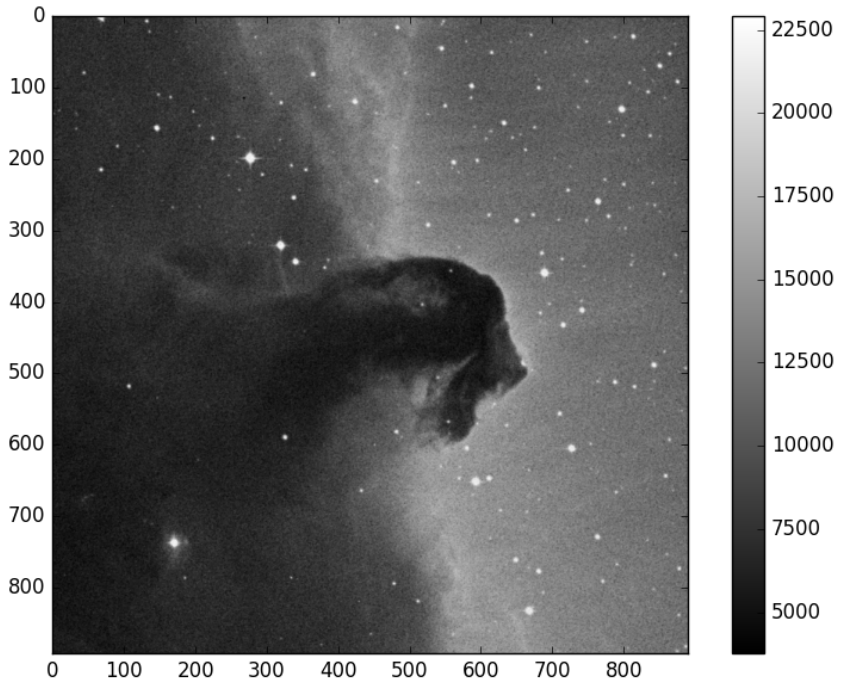
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from astropy.modeling import models, fitting

>>> g1 = models.Gaussian1D(1, 0, 0.2)
>>> g2 = models.Gaussian1D(2.5, 0.5, 0.1)
>>> x = np.linspace(-1, 1, 200)
>>> y = g1(x) + g2(x) + np.random.normal(0., 0.2, x.size)

>>> gg_init = models.Gaussian1D(1, 0, 0.1) + models.Gaussian1D(2, 0.5, 0.1)
>>> fitter = fitting.SLSQPLSQFitter()
>>> gg_fit = fitter(gg_init, x, y)
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 8.33
            Iterations: 15
            Function evaluations: 145
            Gradient evaluations: 15

>>> plt.figure(figsize=(8,5))
>>> plt.plot(x, y, 'ko')
>>> plt.plot(x, gg_fit(x), 'r-', lw=2)
>>> plt.xlabel(u'Posição')
>>> plt.ylabel('Fluxo')
>>> plt.show()
```



Tópicos

Imagens FITS

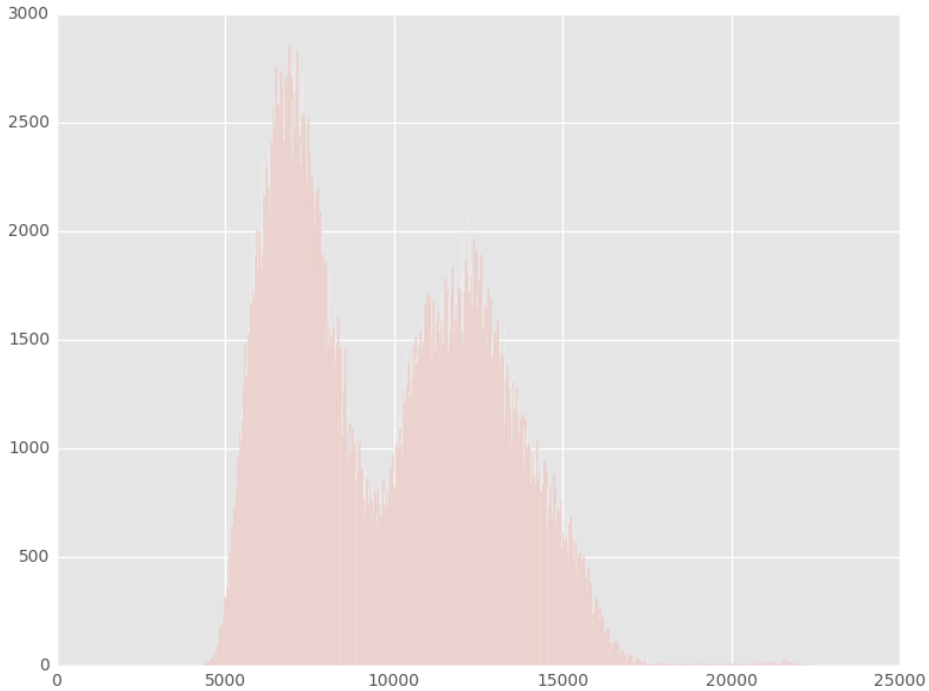
```
>>> import numpy as np
>>> from astropy.io import fits
>>> import matplotlib.pyplot as plt
>>> hdu_list = fits.open("HorseHead.fits")
>>> hdu_list.info()
Filename: HorseHead.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     161   (891, 893)   int16
1    er.mask     TableHDU        25   1600R x 4C   [F6.2, F6.2, F6.2, F6.2]

>>> image_data = hdu_list[0].data
>>> print image_data.shape
(893, 891)
>>> plt.imshow(image_data, cmap='gray')
>>> plt.colorbar()
>>> plt.show()
>>> print "Minimo:", image_data.min()
Minimo: 3759
>>> print "Maximo:", image_data.max()
Maximo: 22918
>>> print "Media:", image_data.mean()
Media: 9831.48167629
>>> print "Sigma:", image_data.std()
Sigma: 3032.3927542
```



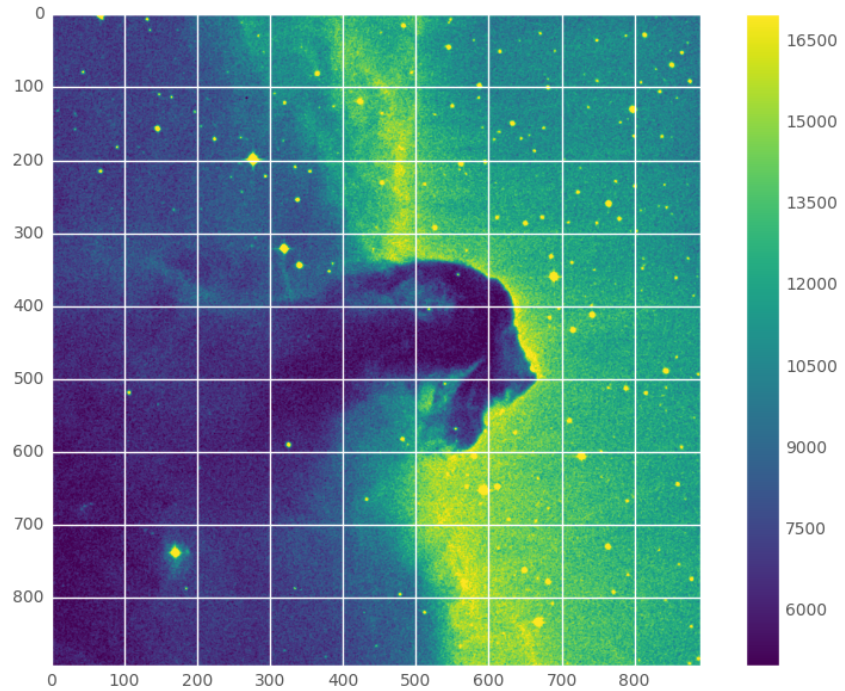
Imagens FITS

- Estilos para gráficos: https://tonysyu.github.io/raw_content/matplotlib-style-gallery/gallery.html
- Mapas de cores: http://matplotlib.org/users/colormaps.html

```
>>> plt.style.use('ggplot')
>>> plt.hist(image_data.flatten(), 1000)
>>> plt.show()
```




```
>>> plt.imshow(image_data, cmap='viridis',
...                vmin=5.e3, vmax=1.7e4)
>>> plt.colorbar()
>>> plt.show()
```



Tópicos

Informação de catálogo

```
>>> from astropy.io import ascii
>>> tbl = ascii.read("Young-Objects-Compilation.csv", header_start=1, data_start=2)
```

```
>>> print tbl.colnames
['Name',
 'Designation',
 'RA',
 'Dec',
 'Jmag',
 'J_unc',
 'Hmag',
 'H_unc',
 'Kmag',
 'K_unc',
 'W1',
 ...
 'pi (mas)',
 'pi_unc',
 'radial velocity (km/s)',
 'rv_unc',
 'Astrometry Refs',
 'Discovery Refs',
 'Group/Age',
 'Note']
```

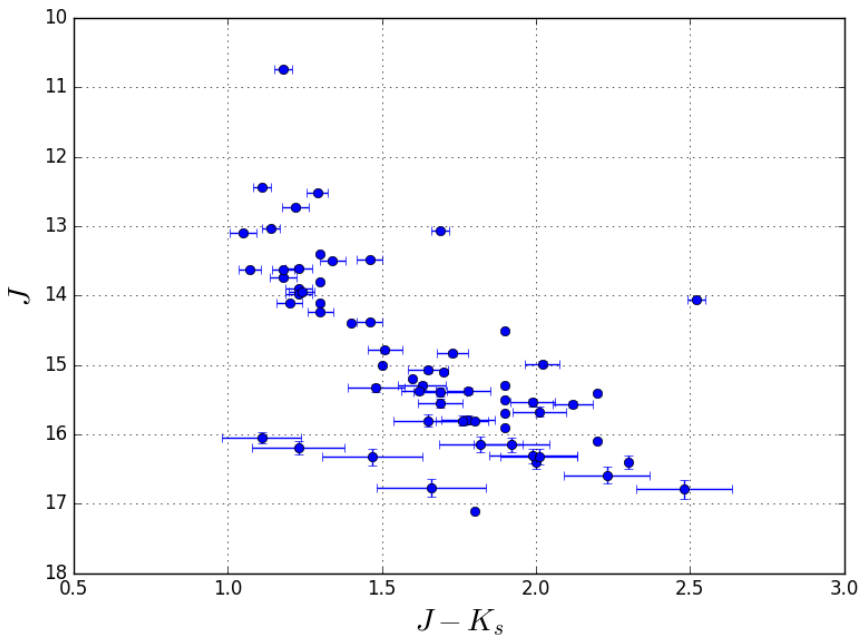
```
>>> print tbl['RA']
    RA
---------
  1.01201
  6.92489
  8.23267
  9.42942
 11.33929
       --
       --
       --
      ...
303.46467
       --
       --
332.05679
333.43715
342.47273
       --
350.72079
Length = 64 rows
```

```
>>> print tbl['RA'].filled(np.nan)
    RA
---------
  1.01201
  6.92489
  8.23267
  9.42942
 11.33929
      nan
      nan
      nan
      ...
303.46467
      nan
      nan
332.05679
333.43715
342.47273
      nan
350.72079
Length = 64 rows
```

Informação de catálogo

- Diagrama cor-magnitude: (J - K) vs. J

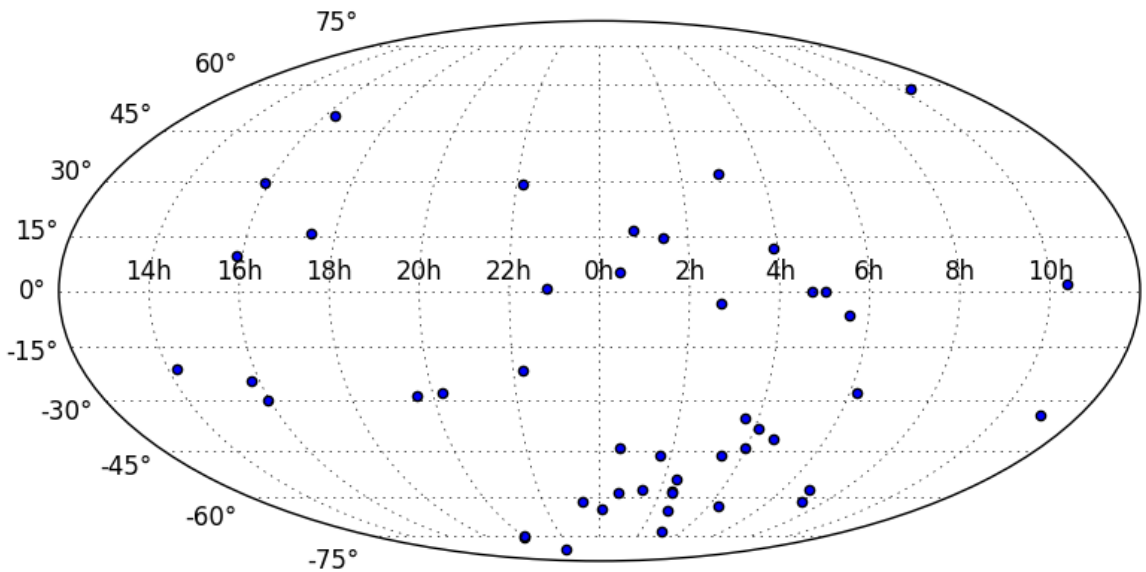
```
>>> import numpy as np
>>> from astropy.io import ascii
>>> import matplotlib.pyplot as plt

>>> tbl = ascii.read("Young-Objects-Compilation.csv", header_start=1, data_start=2)
>>> x = tbl['Jmag'] - tbl['Kmag']
>>> y = tbl['Jmag']
>>> sx = np.sqrt(tbl['J_unc']**2 + tbl['K_unc']**2)
>>> sy = tbl['J_unc']
>>> plt.errorbar(x, y, xerr=sx, yerr=sy, fmt='bo')
>>> plt.ylim(reversed(plt.ylim()))
>>> plt.xlabel("$J-K_s$", fontsize=20)
>>> plt.ylabel("$J$", fontsize=20)
>>> plt.grid()
>>> plt.tight_layout()
>>> plt.show()
```



Informação de catálogo

- Projeção do céu bi-dimensional: Mollweide

```
>>> import astropy.units as u
>>> import astropy.coordinates as coord
>>> ra = coord.Angle(tbl['RA'].filled(np.nan)*u.degree)
>>> ra = ra.wrap_at(180*u.degree)
>>> dec = coord.Angle(tbl['Dec'].filled(np.nan)*u.degree)

>>> fig = plt.figure()
>>> ax = fig.add_subplot(111, projection='mollweide')
>>> ax.scatter(ra.radian, dec.radian)
>>> ax.set_xticklabels(['14h','16h','18h','20h','22h',
                        '0h','2h','4h','6h','8h','10h'])
>>> ax.grid()
>>> plt.show()
```



Tópicos

Pacote afiliado: Astroquery

- Conjunto de ferramentas para consultar catálogos astronômicos online
  http://astroquery.readthedocs.io

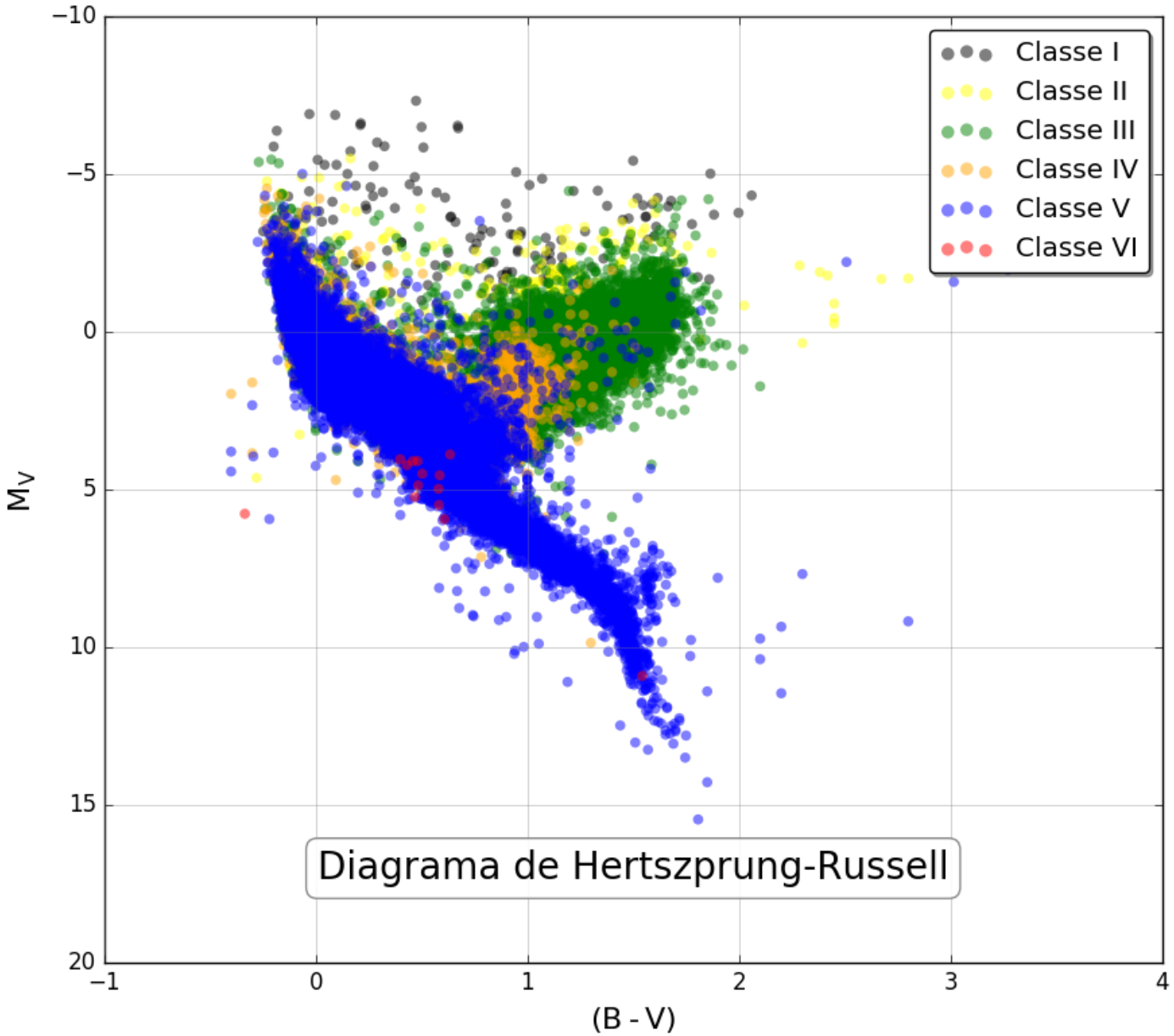
Diagrama de Hertszprung-Russell usando o catálogo XHIP do Vizier:

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from astroquery.vizier import Vizier

>>> v = Vizier(columns=['Lc', 'B-V', 'VMag'],
...            column_filters={'Lc': '!=', 'B-V': '!=', 'VMag': '!='},
...            row_limit=-1)

>>> result = v.query_constraints(catalog='V/137D')  # XHIP catalog
>>> lc = result[0]['Lc'].data.data
>>> bv = result[0]['B-V'].data.data
>>> mv = result[0]['VMag'].data.data

>>> bvlist = [bv[lc == i] for i in np.arange(1, 7)]
>>> mvlist = [mv[lc == i] for i in np.arange(1, 7)]
>>> lclist = ['Classe I', 'Classe II', 'Classe III',
...           'Classe IV', 'Classe V', 'Classe VI']
>>> colors = ['black', 'yellow', 'green', 'orange', 'blue', 'red']
```

Pacote afiliado: Astroquery

```
>>> fig = plt.figure(figsize=(9, 8))
>>> ax = fig.add_subplot(111)

>>> for x, y, l, c in zip(bvlist, mvlist, lclist, colors):
...     ax.scatter(x, y, c=c, s=30, label=l, alpha=0.5, edgecolors='none')

>>> ax.set_xlabel(r'$\mathregular{(B-V)}$', fontsize=16)
>>> ax.set_ylabel(r'$\mathregular{M_V}$', fontsize=16)
>>> ax.set_xlim(-1, 4)
>>> ax.invert_yaxis()
>>> ax.grid(ls='-', c='gray', alpha=0.5)
>>> ax.legend(markerscale=1.25, fancybox=True, shadow=True)

>>> bbox_props = dict(boxstyle='round', fc='w', ec='0.5', alpha=0.9)
>>> ax.text(1.5, 17, 'Diagrama de Hertszprung-Russell', ha='center',
...         va='center', size=20, bbox=bbox_props)

>>> plt.tight_layout()
>>> plt.show()
```

Pacote afiliado: Astroquery



Tópicos

Material do curso

Documentação

- Python
  https://docs.python.org/2/tutorial/index.html
- NumPy
  https://docs.scipy.org/doc/numpy-dev/user/index.html
  https://docs.scipy.org/doc/numpy-dev/reference/index.html
- Matplotlib
  http://matplotlib.org/users/beginner.html
- SciPy
  http://docs.scipy.org/doc/scipy/reference/index.html
- Astropy
  http://docs.astropy.org/en/stable/index.html