Bright Sky Model

We want to create a better model for the moon.

In [1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [26]:
```
import os, glob, sys, fnmatch
import numpy as np
import pandas as pd
import seaborn as sns
import pandas as pd
from astropy.io import fits

import astropy.table
import astropy.units as u
from astropy.time import Time
import corner
import seaborn as sns

import sklearn.linear_model
import specsim.atmosphere
from scipy import interpolate
import statsmodels.api as sm
from lmfit import models, Parameters, Parameter, Model
from scipy.optimize import curve_fit

import speclite
import astropy.units as u

from matplotlib.font_manager import FontProperties
font = FontProperties()
font.set_family('serif')
font.set_size('small')
```

Define Data

First, look at bright photometric data. Bright is defined as 2.5 x mean dark level, which is 2.79 $10^{-17} erg/cm^2/s/A$

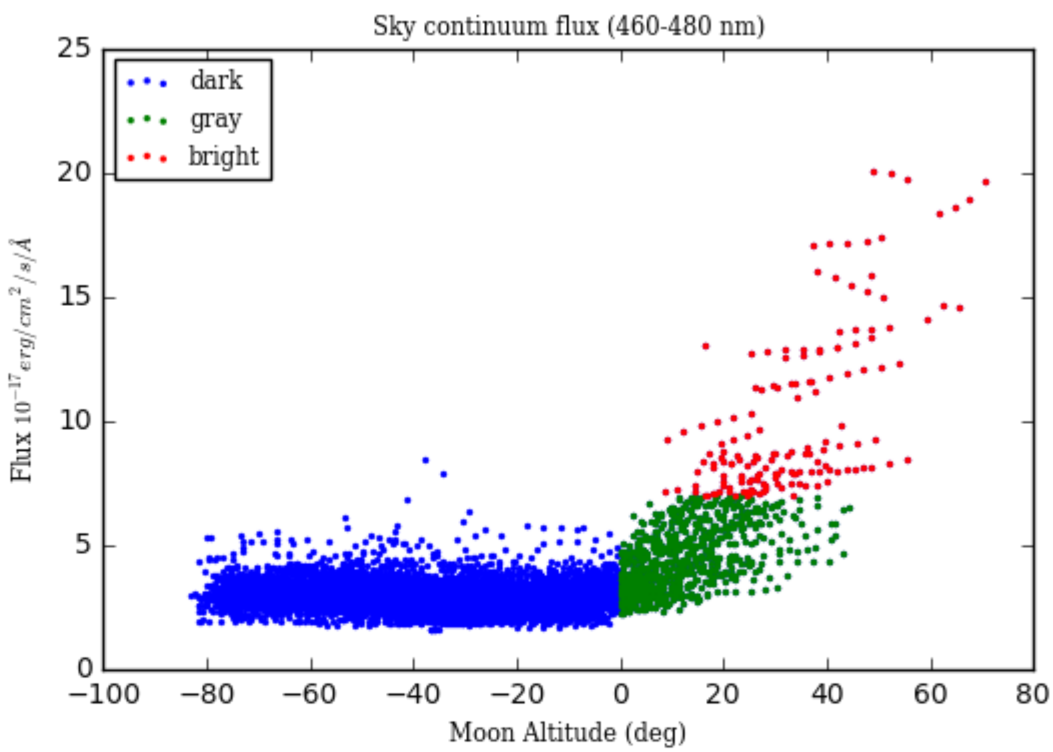
In [3]:
```
data = astropy.table.Table.read('data/good_data.fits')
```

In [4]:
```
def make_data_dict(data = data, plot = False):
    gray_level = 2.5*2.79
    gray = np.where((data['MOON_ALT']>0)&(data['SKY_VALUE']<gray_leve
    bright = np.where((data['MOON_ALT']>0)&(data['SKY_VALUE']>gray_le

    if plot:
        plt.figure()
        plt.scatter(data['MOON_ALT'], data['SKY_VALUE'], s = 2, colo
        plt.scatter(data['MOON_ALT'][gray], data['SKY_VALUE'][gray],
        plt.scatter(data['MOON_ALT'][bright], data['SKY_VALUE'][brigl
        plt.legend(loc='upper left', prop=font)
        plt.xlabel("Moon Altitude (deg)", fontproperties = font)
        plt.ylabel("Flux $10^{-17} erg/cm^{2}/s/\AA$", fontpropertie:
        plt.title("Sky continuum flux (460-480 nm)", fontproperties =

    return {'good': data, 'gray': data[gray], 'bright': data[bright]

data_dict = make_data_dict(plot=True)
```



In [5]:
```
# Add in clouds/photometricity
cloud_data = np.load('clouds/phot_rec.npy')
def get_cloud_data(line):
    df = pd.DataFrame(cloud_data)
    clouds = df[(df['STARTTAI']<= line['TAI-BEG']) & (df['ENDTAI']>li
    if len(clouds) == 0:
        clouds = df[(df['STARTTAI']< line['TAI-END']) & (df['ENDTAI'
    if len(clouds) == 0:
        clouds = 0.5

    return clouds
```

In [6]:
```
clouds = [get_cloud_data(line) for line in data]
data['PHOTO'] = astropy.table.Column(np.hstack(clouds).astype(np.floa
data_dict = make_data_dict()
```

In [7]:
```
def photometric_cut(data = data):
    phot_data = data[data['PHOTO'] == 1]
    phot_data_dict = make_data_dict(data = phot_data)
    return phot_data_dict
phot_data_dict = photometric_cut()
```

How good is a linear regression model?

These results are only from the photometric bright nights. The linear regression fit is pretty good. It helps me identify components important to the moon model

In [8]:
```
def linear_regression(data_name = 'bright', data_dict = data_dict):
    data = data_dict[data_name]
    #features = ['MOON_ILL','MOON_ALT','MOON_SEP','AIRMASS','AZ','SU
    features = ['MOON_ILL','MOON_ALT','MOON_SEP','AIRMASS','SUN_MOON_
                'ECL_LON','ECL_LAT','GAL_LAT','SEASON','HOUR','MOON_
                'PRESSURE','SUN_ALT','DUSTB','MOON_LAT','MOON_LON','
                'WINDS25M','GUSTS','HUMIDITY','MOON_D']

    X = []
    for feat in features:
        X.append(data[feat])
    X = np.column_stack(X)

    y = data['SKY_VALUE']
    print(X.shape, y.shape)

    sm.OLS.exog_names = features
    results = sm.OLS(y, X).fit()

    params = results.params
    model = np.dot(X, params)

    print(results.summary())


    xmin, xmax = np.percentile(model, (1, 99))
    ymin, ymax = np.percentile(y - model, (1, 99))
    plt.hist2d(model, y-model,bins=(50,50), cmap=plt.cm.jet)
    plt.colorbar()
    plt.xlim(xmin, xmax)
    plt.ylim(ymin, ymax)
    plt.xlabel("Model flux",fontproperties=font)
    plt.ylabel("Data - Model (residuals)",fontproperties=font)
    plt.title("Residual plot for Linear Regression Model")
linear_regression(data_name='bright', data_dict = phot_data_dict)
```

```
(86, 22) (86,)
                            OLS Regression Results
==================================================================
============
Dep. Variable:                      y   R-squared:
0.997
Model:                            OLS   Adj. R-squared:
0.997
Method:                 Least Squares   F-statistic:
1137.
Date:                Wed, 24 Jan 2018   Prob (F-statistic):
1.51e-74
Time:                        15:17:34   Log-Likelihood:
-62.224
No. Observations:                  86   AIC:
168.4
Df Residuals:                      64   BIC:
222.4
```

Now, we want to create a better more physical model

In [11]:

```
wavelength = np.linspace(3550, 9350, (9350-3550)*10)*u.Angstrom
sky_unit = 1e-17 * u.erg / (u.cm**2 * u.s * u.Angstrom * u.arcsec**2
area = 1 * u.arcsec ** 2
```

In [12]:

```
#moon spectrum
moon_s = np.loadtxt('/Users/parkerf/Research/SkyModel/DESI_Sky/solar
moon_s = interpolate.interp1d(moon_s[:,1], moon_s[:,2], bounds_error=
moon_spectrum = moon_s(wavelength)* u.erg/ (u.cm**2 * u.s * u.Angstr

#extinction curve
ext_c = np.loadtxt('/Users/parkerf/Research/SkyModel/DESI_Sky/Zenith
ext_c = interpolate.interp1d(ext_c[:,0], ext_c[:,1], bounds_error=Fa
extinction_coefficient = ext_c(wavelength)
```

In [18]:

```
moon_zenith = 90-data['MOON_ALT']
data['MOON_ZENITH'] = astropy.table.Column(moon_zenith.astype(np.flo
# Compute the pointing zenith angle in degrees.
obs_zenith = 90 - data['ALT']
data['OBS_ZENITH'] = astropy.table.Column(obs_zenith.astype(np.float

mphase = np.arccos(2 * data['MOON_ILL'] - 1) / np.pi
data['MPHASE'] = astropy.table.Column(mphase.astype(np.float32))

helio_lon = data['ECL_LON'] - data['SUN_LON']
data['HELIO_LON'] = astropy.table.Column(np.array(helio_lon).astype(
```

Calculate Values for Moon Brightness

In [14]:

```
# Calculate the V-band extinction of the moon spectrum.
_vband = speclite.filters.load_filter('bessell-V')
V = _vband.get_ab_magnitude(moon_spectrum, wavelength)
extinction = 10 ** (-extinction_coefficient / 2.5)
Vstar = _vband.get_ab_magnitude(
    moon_spectrum * extinction, wavelength)
_vband_extinction = Vstar - V
```

In [19]:
```
#Calculate the wavelength-dependent extinction of moonlight
# scattered once into the observed field of view.
idx = np.where((wavelength>4600*u.Angstrom)&(wavelength<4800*u.Angst
ext = np.array(extinction_coefficient)

scattering_airmass = np.array((
    1 - 0.96 * np.sin(data['MOON_ZENITH']) ** 2) ** (-0.5))
data['scatt_airmass'] = astropy.table.Column(scattering_airmass.asty
```

In [20]:
```
RAWV = []
MEAN_EXT = []
for item in data:
    extinction = (10 ** ((-ext * item['scatt_airmass']) / 2.5) *
        (1 - 10 ** (-ext * item['AIRMASS'] / 2.5)))
    surface_brightness = moon_spectrum * extinction

    # Renormalized the extincted spectrum to the correct V-band magn
    raw_V = _vband.get_ab_magnitude(
        surface_brightness, wavelength) * u.mag
    RAWV.append(raw_V.value)

    idx = np.where((wavelength>4400*u.Angstrom)&(wavelength<4800*u.A
    MEAN_EXT.append(np.mean(extinction[idx]))

data['raw_V'] = astropy.table.Column(np.array(RAWV).astype(np.float3
data['mean_ext'] = astropy.table.Column(np.array(MEAN_EXT).astype(np
```

Calculate values for Zodiacal light

In [23]:
```
leinert_lookup = '/Users/parkerf/Research/SkyModel/BOSS_Sky/Model/fi
ZodiLookup = np.load(leinert_lookup)
S10 = 1.28*10**(-9) #erg/cm2/s/sr/A  @ 500nm
sr = 4.25*10**(10) #arcsec^2/sr
fiber_area = np.pi
```

In [24]:
```
hdu = fits.open('ftp://ftp.stsci.edu/cdbs/current_calspec/sun_refere
sun_spectrum = hdu[1].data
hdu.close()
sun_wave = sun_spectrum['WAVELENGTH']/10. #nm
sun_flux = sun_spectrum['FLUX'] #erg/s/cm2/A
sun_spectrum = interpolate.interp1d(sun_wave, sun_flux, bounds_error
#relative to 500nm
nm_500 = [np.abs(500.-w) for w in sun_wave]
nm_id = np.argmin(nm_500)
relative_sun_flux = sun_flux/sun_flux[nm_id]
```

In [25]:
```
ZODI = []
for line in data:
    lamb = np.argmin([np.abs(np.abs(line['HELIO_LON'])-lon) for lon i
    beta = np.argmin([np.abs(np.abs(line['ECL_LAT'])-lat) for lat in
    BASE_ZODI = ZodiLookup[2][lamb][beta]

    IO = BASE_ZODI*S10*(fiber_area/sr)
    flux = relative_sun_flux*IO*10**17
    zodi_spect = interpolate.interp1d(sun_wave*10, flux, bounds_erro
    ext = 10 ** (-extinction_coefficient * line['AIRMASS'] / 2.5)
    zodi_flux = zodi_spect(wavelength)*ext*u.erg/ (u.cm**2 * u.s * u
    idx = np.where((wavelength>4600*u.Angstrom)&(wavelength<4800*u.An
    ZODI.append(np.mean(zodi_flux[idx]).value)
data['ZODI'] = astropy.table.Column(np.array(ZODI).astype(np.float32
```

In [27]:
```
data_dict = make_data_dict()
phot_data_dict = photometric_cut()
this_data = np.random.choice(phot_data_dict['bright'],1)
```

In [32]:
```
def bright_sky_model(X, A, B, C, D, E, a1, a2, a3, a4, a5, a6, a7, a8
    moon_ill, moon_sep, moon_alt, airmass, Xmoon, raw_V, mean_ext, zo

    #The following is from the specsim.atmosphere model with constan
    abs_alpha = 180. * moon_ill
    m = -a1 + a2 * abs_alpha + a3 * abs_alpha ** 4

    # Calculate the illuminance of the moon outside the atmosphere i
    # foot-candles (eqn. 8).
    Istar = 10 ** (-0.4 * (m + a4))

    # Calculate the scattering function (eqn.21).
    rho = moon_sep #separation_angle.to(u.deg).value
    f_scatter = (10 ** a5 * (a6 + np.cos(np.deg2rad(rho)) ** 2) +
                 10 ** (a7 - rho / a8))


    # Calculate the V-band moon surface brightness in nanoLamberts.
    B_moon = (f_scatter * Istar *
        10 ** (-0.4 * _vband_extinction * Xmoon) *
        (1 - 10 ** (-0.4 * (_vband_extinction * airmass))))

    # Convert from nanoLamberts to to mag / arcsec**2 using eqn.19 o
    # Garstang, "Model for Artificial Night-Sky Illumination",
    # PASP, vol. 98, Mar. 1986, p. 364 (http://dx.doi.org/10.1086/13
    _scattered_V =  ((20.7233 - np.log(B_moon / 34.08)) / 0.92104 *
            u.mag / (u.arcsec ** 2))

    idx = np.where((wavelength>4600*u.Angstrom)&(wavelength<4800*u.An
    surface_brightness = np.mean(moon_spectrum[idx]) * mean_ext

    area = 1 * u.arcsec ** 2

    surface_brightness *= 10 ** (
        -(_scattered_V * area - raw_V * u.mag) / (2.5 * u.mag))

    moon =  (surface_brightness*10**17).value
    sky = A*zodi + moon #+ B*pressure + C*dust +D*airtemp + E*dewpoi
    return moon
```

In [33]:
```
my_data = phot_data_dict['bright']
p0 = 1, 1, 1,1,1, 12.73, 0.025, 4*10**(-9), 16.57, 5.36, 1.06, 6.15,
xx = (my_data['MOON_ILL'], my_data['MOON_SEP'],my_data['MOON_ALT'], m
      my_data['scatt_airmass'], my_data['raw_V'], my_data['mean_ext'
      my_data['PRESSURE'],my_data['DUSTB'],my_data['AIRTEMP'],my_data

popt, pcov = curve_fit(bright_sky_model, xx, my_data['SKY_VALUE'],p0
```

```
/Users/parkerf/anaconda3/lib/python3.5/site-packages/ipykernel/__m
ain__.py:10: RuntimeWarning: overflow encountered in power
/Users/parkerf/anaconda3/lib/python3.5/site-packages/ipykernel/__m
ain__.py:20: RuntimeWarning: invalid value encountered in multiply
/Users/parkerf/anaconda3/lib/python3.5/site-packages/scipy/optimiz
e/minpack.py:715: OptimizeWarning: Covariance of the parameters co
uld not be estimated
  category=OptimizeWarning)
```

In [30]:
```
print(p0)
print(popt)
```

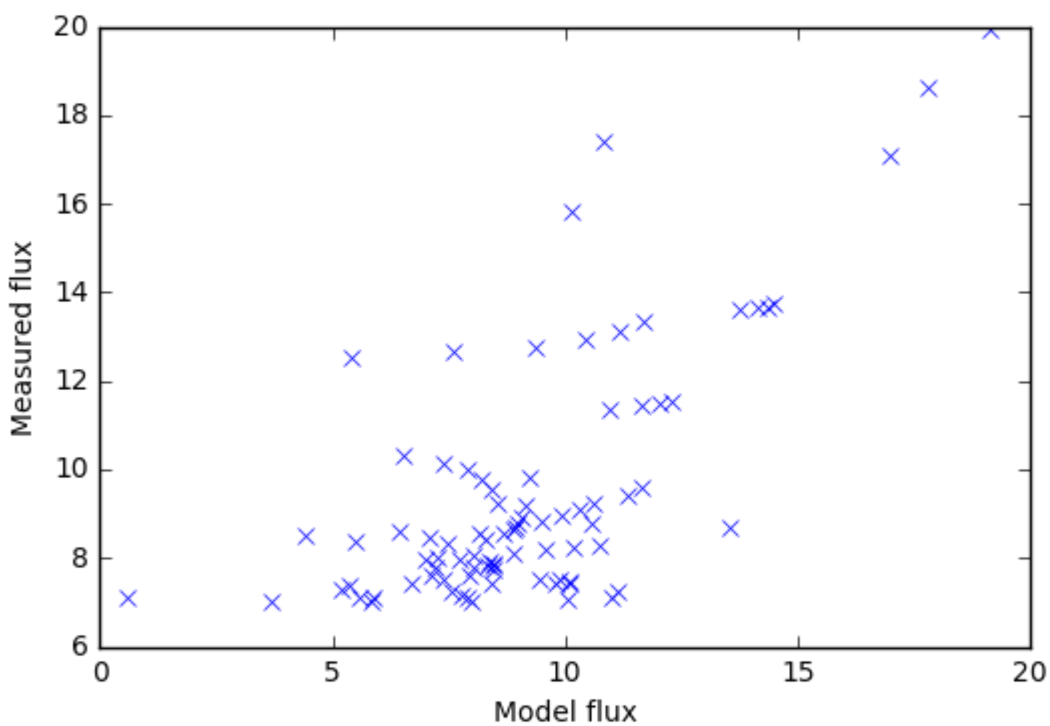
```
(1, 1, 1, 1, 1, 12.73, 0.025, 4e-09, 16.57, 5.36, 1.06, 6.15, 40.0
)
[  1.00000000e+00   1.00000000e+00   1.00000000e+00   1.00000000e+
00
   1.00000000e+00  -2.37544188e+02  -4.84908534e-02   1.20063607e-
08
  -2.28078738e+02   6.21961954e+00   6.58817916e-01   5.63404504e+
00
   1.29500833e+00]
```

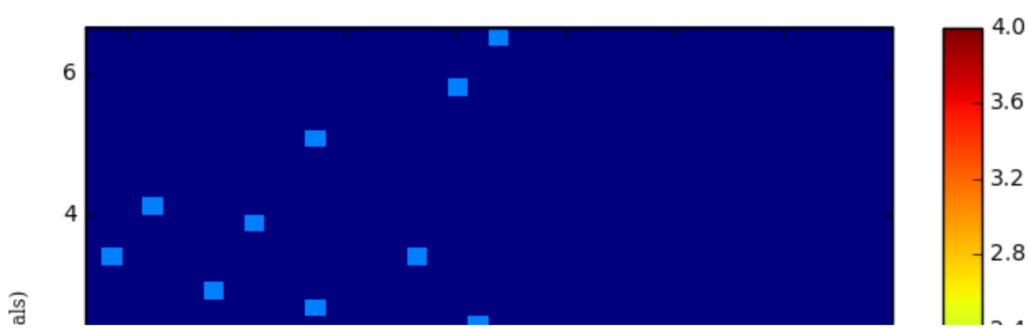
In [36]:
```
model = bright_sky_model(xx, *popt)
res = my_data['SKY_VALUE']-model

plt.figure()
plt.plot(model, my_data['SKY_VALUE'],'x')
plt.xlabel("Model flux")
plt.ylabel("Measured flux")
fig, ax = plt.subplots(1, figsize=(8,6))

xmin, xmax = np.percentile(model, (1, 99))
ymin, ymax = np.percentile(res, (1, 99))
plt.hist2d(model, res,bins=(50,50), cmap=plt.cm.jet)
plt.colorbar()
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)
plt.xlabel("Model flux",fontproperties=font)
plt.ylabel("Data - Model (residuals)",fontproperties=font)
```

Out[36]: <matplotlib.text.Text at 0x12137cac8>





In [ ]: