Reproducing the Third Figure from Lunch Talk: Comparing Young stars at Galactic Center using (Using Fall-2020 Version of SPISEA from fork)

Here, I will be attempting to reproduce the third figure from my lunch talk on 3/4/2021.

For the first two cells, I will be establishing the functions used to read out the data file containing step K' band luminosity function. The original code for the function in this cell can be found in jlu_python/jlu/papers/lu_gc_imf.py on the Moving Universe Lab's computers.

In [1]:

```python
# After spending a bit of time figuring out how pickling files and loading work
# I have finally done it.

# These come from
import pickle
import sys

class dat:
    datin = "Container of Data"

def load_klf_by_radius(rmin=0, rmax=30, mask_for_log=False):
    pickleFile = ('%sklf_r_%.1f_%.1f.dat' %
                  ('/Users/jlu/work/gc/imf/klf/current/', rmin, rmax))
    _in = open(pickleFile, 'rb')
    d = dat()
    d.Kp = pickle.load(_in,encoding='bytes')
    d.N = pickle.load(_in, encoding='bytes')
    d.eN = pickle.load(_in, encoding='bytes')

    d.N_ext = pickle.load(_in, encoding='bytes')
    d.eN_ext = pickle.load(_in, encoding='bytes')

    d.N_ext_cmp_sp = pickle.load(_in, encoding='bytes')
    d.eN_ext_cmp_sp = pickle.load(_in, encoding='bytes')

    d.N_ext_cmp_sp_im = pickle.load(_in, encoding='bytes')
    d.eN_ext_cmp_sp_im = pickle.load(_in, encoding='bytes')

    d.KLF = pickle.load(_in, encoding='bytes')
    d.eKLF = pickle.load(_in, encoding='bytes')

    d.KLF_ext = pickle.load(_in,
                            encoding='bytes')
    d.eKLF_ext = pickle.load(_in,
                             encoding='bytes')

    d.KLF_ext_cmp_sp = pickle.load(_in, encoding='bytes')
    d.eKLF_ext_cmp_sp = pickle.load(_in, encoding='bytes')

    d.KLF_ext_cmp_sp_im = pickle.load(_in,
                                      encoding='bytes')
    d.eKLF_ext_cmp_sp_im = pickle.load(_in,
                                       encoding='bytes')

    d.N_noWR = pickle.load(_in, encoding='bytes')
    d.eN_noWR = pickle.load(_in, encoding='bytes')

    d.N_ext_noWR = pickle.load(_in, encoding='bytes')
    d.eN_ext_noWR = pickle.load(_in, encoding='bytes')

    d.N_ext_cmp_sp_noWR = pickle.load(_in,
                                      encoding='bytes')
    d.eN_ext_cmp_sp_noWR = pickle.load(_in,
                                       encoding='bytes')

    d.N_ext_cmp_sp_im_noWR = pickle.load(_in,
                                         encoding='bytes')
    d.eN_ext_cmp_sp_im_noWR = pickle.load(_in,
                                          encoding='bytes')

    d.KLF_noWR = pickle.load(_in, encoding='bytes')
    d.eKLF_noWR = pickle.load(_in, encoding='bytes')

    d.KLF_ext_noWR = pickle.load(_in, encoding='bytes')
    d.eKLF_ext_noWR = pickle.load(_in, encoding='bytes')

    d.KLF_ext_cmp_sp_noWR = pickle.load(_in,
                                        encoding='bytes')
    d.eKLF_ext_cmp_sp_noWR = pickle.load(_in,
                                         encoding='bytes')

    d.KLF_ext_cmp_sp_im_noWR = pickle.load(_in,
                                           encoding='bytes')
    d.eKLF_ext_cmp_sp_im_noWR = pickle.load(_in,
                                            encoding='bytes')

    d.comp_spec_ext = pickle.load(_in, encoding='bytes')
    d.comp_imag_ext = pickle.load(_in, encoding='bytes')

    if mask_for_log:
        # Repair for zeros since we are plotting in semi-log-y
        d.eN = np.ma.masked_where(d.N <= 0, d.eN)
        d.N = np.ma.masked_where(d.N <= 0, d.N)

        d.eN_ext = np.ma.masked_where(d.N_ext <= 0, d.eN_ext)
        d.N_ext = np.ma.masked_where(d.N_ext <= 0, d.N_ext)

        d.eN_ext_cmp_sp = np.ma.masked_where(d.N_ext_cmp_sp <=
                                             0, d.eN_ext_cmp_sp)
        d.N_ext_cmp_sp = np.ma.masked_where(d.N_ext_cmp_sp <=
                                            0, d.N_ext_cmp_sp)

        d.eN_ext_cmp_sp_im = np.ma.masked_where(d.N_ext_cmp_sp_im <=
                                                0, d.eN_ext_cmp_sp_im)
        d.N_ext_cmp_sp_im = np.ma.masked_where(d.N_ext_cmp_sp_im <=
                                               0, d.N_ext_cmp_sp_im)

        d.eKLF = np.ma.masked_where(d.KLF <= 0, d.eKLF)
        d.KLF = np.ma.masked_where(d.KLF <= 0, d.KLF)

        d.eKLF_ext = np.ma.masked_where(d.KLF_ext <= 0,
                                        d.eKLF_ext)
        d.KLF_ext = np.ma.masked_where(d.KLF_ext <= 0,
                                       d.KLF_ext)

        d.eKLF_ext_cmp_sp = np.ma.masked_where(d.KLF_ext_cmp_sp <= 0,
                                               d.eKLF_ext_cmp_sp)
        d.KLF_ext_cmp_sp = np.ma.masked_where(d.KLF_ext_cmp_sp <= 0,
                                              d.KLF_ext_cmp_sp)

        d.eKLF_ext_cmp_sp_im = (np.ma.
                                masked_where(d.KLF_ext_cmp_sp_im <=
                                             0, d.eKLF_ext_cmp_sp_im))
        d.KLF_ext_cmp_sp_im = (np.ma.
                               masked_where(d.KLF_ext_cmp_sp_im <=
                                            0, d.KLF_ext_cmp_sp_im))


    _in.close()

    return d
```

The following line loads data from /Users/jlu/work/gc/imf/klf/current/klf_r_0.0_30.0.dat. Recall I use the noWR option as the corresponding figure

In [2]:

```
import numpy as np
result = load_klf_by_radius()
result.KLF_ext_cmp_sp_im_noWR
magBin = result.Kp[1] - result.Kp[0]
idx = np.where(result.Kp < 16)[0]
```

Now, I create the BPASS isochrone for that specific age. $10^{6.78}$ years of age (around 6 million years). Out of that, I will create a cluster using an IMF of $\alpha = 1.7$. (Note that, when I create my BPASS cluster, I assume that there are no stars with 200 $M_\odot$; that is physically difficult to have for a Population I system, which have relatively high metallicity.)

In [3]:

```
from spisea import synthetic
BPASS_iso = synthetic.Isochrone_Binary(6.78, 2.7,
                                       8000, 0.0, filters=['nirc2,Kp'])
```

```
Changing to logg=5.00 for T=113729 logg=5.40
Changing to logg=3.50 for T= 27428 logg=3.36
Changing to logg=3.50 for T= 28471 logg=3.47
Changing to T= 50000 for T=171676 logg=5.85
Changing to logg=5.00 for T=171676 logg=5.85
Changing to logg=3.50 for T= 27428 logg=3.36
Changing to logg=3.50 for T= 27428 logg=3.36
Changing to logg=3.50 for T= 26250 logg=3.24
Changing to logg=3.50 for T= 26382 logg=3.26
Changing to logg=3.50 for T= 26250 logg=3.24
Changing to T= 50000 for T=171933 logg=5.86
Changing to logg=5.00 for T=171933 logg=5.86
Changing to logg=3.50 for T= 27428 logg=3.36
Changing to T= 50000 for T=132850 logg=5.43
Changing to logg=5.00 for T=132850 logg=5.43
Changing to T= 50000 for T=172910 logg=5.87
Changing to logg=5.00 for T=172910 logg=5.87
Changing to T= 50000 for T=144521 logg=5.57
Changing to logg=5.00 for T=144521 logg=5.57
Changing to logg=4.00 for T= 34533 logg=3.99
```

In [4]:

```
from spisea import imf, ifmr
import numpy as np
custom_IMF = imf.imf.IMF_broken_powerlaw(np.array([1, 200]),
                                          np.array([-1.7]),
                                          multiplicity=
                                          (imf.multiplicity.
                                           MultiplicityResolvedDK()))
import time
t1 = time.time()
BPASS_Cluster = synthetic.Binary_Cluster(BPASS_iso, custom_IMF,
                                          170000,
                                          ifmr=ifmr.IFMR_Spera15())
t2 = time.time()
t_time = t2 - t1
```

```
/opt/anaconda3/envs/astroconda/lib/python3.7/site-packages/astropy/table/column.p
y:1020: RuntimeWarning: invalid value encountered in greater_equal
  result = getattr(super(), op)(other)
```

In [5]:

```
area = 116.098   # arcsec^2
# The scaling factor we use to turn number of stars in a magnitude bin
# to area (arcsecond^2) density of stars in a magnitude bin.
scaleFactorBPS = ((170000/
                  BPASS_Cluster.star_systems['systemMass'].sum()) /
                  area)
print(scaleFactorBPS)
```

```
0.013839288597784945
```

In the next cell, I make sure that I am NOT including WR stars in my plots and that I am not including stars with magnitudes greater than cutoff (K' = 15.5)

In [6]:

```
totl =(BPASS_Cluster.star_systems['m_nirc2_Kp']
       [np.where((~BPASS_Cluster.star_systems['isWR']) &
                 (BPASS_Cluster.star_systems['m_nirc2_Kp'] <=15.5))[0]])
```

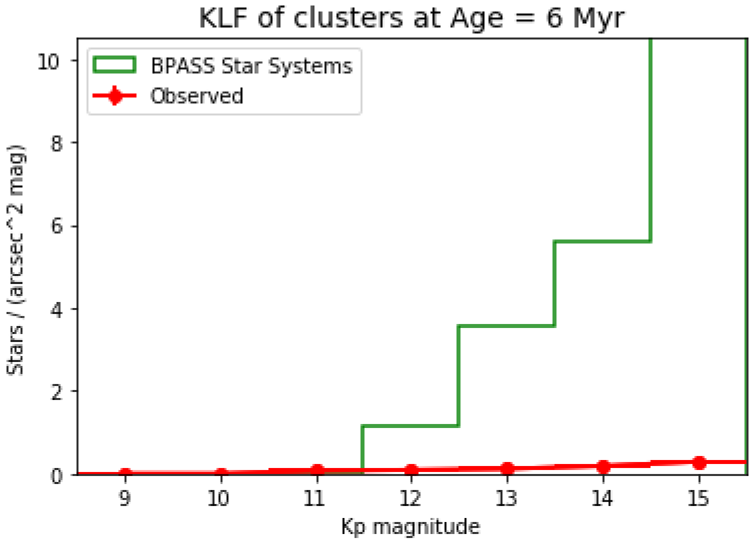
Below, I produce my first plot that demonstrates the problem with BPASS's IMF. I also set up bins and BPASS weights, which I will use in the next figure.

In [7]:

```python
import matplotlib.pyplot as py
# Primary star fluxes
klf_mag_bins = np.arange(9.0, 17, 1.0)
binsKp = klf_mag_bins
binEdges = binsKp[0:-1] + (binsKp[1:] - binsKp[0:-1]) / 2.0

weightsBPS = np.array([1.0 for x in totl])
weightsBPS *= scaleFactorBPS
(n1, b1, p1) = py.hist(totl, bins=binEdges, weights=weightsBPS,
                       histtype='step', color='green',
                       label='BPASS Star Systems', align='mid',
                       linewidth=1.5)
py.errorbar(result.Kp[idx],
            result.KLF_ext_cmp_sp_im_noWR[idx],
            fmt='ro-', xerr=magBin/2.0, linewidth=2)
py.errorbar(result.Kp[idx],result.KLF_ext_cmp_sp_im_noWR[idx],
            fmt='ro-', yerr=result.eKLF_ext_cmp_sp_im_noWR[idx],
            linewidth=2,
            label='Observed')
py.legend(loc='upper left', numpoints=1)
py.ylim(0, 10.5)
py.xlim(8.5, 15.5)
py.xlabel('Kp magnitude')
py.ylabel('Stars / (arcsec^2 mag)')
py.title('KLF of clusters at Age = %d Myr' % (10**(6.78 - 6)),
         fontsize=14)
```

Out[7]:

```
Text(0.5, 1.0, 'KLF of clusters at Age = 6 Myr')
```



Now I create the MIST v1 cluster and then make the final figure of the talk.

In [8]:

```
MIST_iso = synthetic.IsochronePhot(6.78, 2.7, 8000, 0.0,
                                   min_mass = 1.0,
                                   max_mass = 150,
                                   filters=['nirc2,Kp'])
MIST_Cluster = synthetic.ResolvedCluster(MIST_iso, custom_IMF,
                                         170000,
                                         ifmr=ifmr.IFMR_Spera15())
```

```
Found 24 stars out of mass range
Found 610 companions out of stellar mass range
```

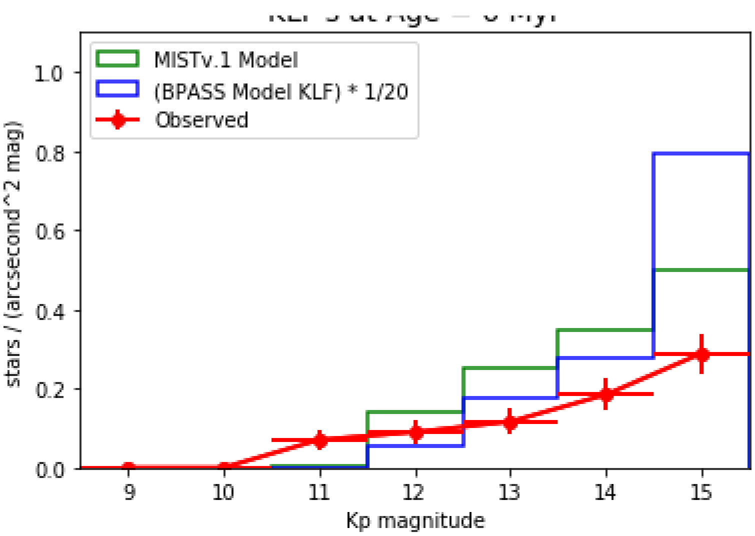
In [9]:

```python
import matplotlib.pyplot as py
# Primary star fluxes
klf_mag_bins = np.arange(9.0, 17, 1.0)
binsKp = klf_mag_bins
binEdges = binsKp[0:-1] + (binsKp[1:] - binsKp[0:-1]) / 2.0
area = 116.098  # arcsec^2
# Setting up framework so that I can scale the number of stars in mass bin
# into density: number of stars per square arcsecond.

mist_scale = ((17000/ MIST_Cluster.star_systems['systemMass'].sum()) /
              area)
totl2 = (MIST_Cluster.star_systems['m_nirc2_Kp']
         [np.where(MIST_Cluster.star_systems['m_nirc2_Kp'] <=
                   15.5)[0]])
weightsMST = np.array([1.0 for x in totl2])
weightsMST *= mist_scale
# Binning
binnd_tot1 = np.digitize(totl, binEdges)
binnd_tot2 = np.digitize(totl2, binEdges)
count1 = [0 for x in range(len(binEdges))]
count2 = [0 for x in range(len(binEdges))]

# binning each of these stars into where they should be
for x in range(len(binnd_tot1)):
    if not (binnd_tot1[x] >= len(count1) or binnd_tot1[x] <= 0):
        count1[binnd_tot1[x]-1] += 1

for x in range(len(binnd_tot2)):
    if not (binnd_tot2[x] > len(count2) or binnd_tot2[x] <= 0):
        count2[binnd_tot2[x]-1] += 1

# Applying weights on star counts in each bin
binnd_tot2 = weightsMST * binnd_tot2

py.hist(totl2, bins=binEdges, histtype='step',
        weights=weightsMST, color='green', label='MISTv.1 Model',
        align='mid', linewidth=1.5)
py.hist(totl, bins=binEdges, histtype='step',
        weights=weightsBPS/20, color='blue',
        label='(BPASS Model KLF) * 1/20',
        align='mid', linewidth=1.5)
py.errorbar(result.Kp[idx], result.KLF_ext_cmp_sp_im_noWR[idx],
            fmt='ro-', xerr=magBin/2.0, capsize=0, linewidth=2)
py.errorbar(result.Kp[idx],result.KLF_ext_cmp_sp_im_noWR[idx],
            fmt='ro-', yerr=result.eKLF_ext_cmp_sp_im_noWR[idx],
            linewidth=2,
       label='Observed')
py.ylim(0, 1.1)
py.xlim(8.5, 15.5)
py.xlabel('Kp magnitude')
py.ylabel("stars / (arcsecond^2 mag)")
py.title("KLF's at Age = %d Myr" % (10**(6.78 - 6)), fontsize=14)
py.legend(loc='upper left', numpoints=1)
```

Out[9]:

```
<matplotlib.legend.Legend at 0x7fc1033e6ba8>
```

KLF's at Age = 6 Myr


KLF 3 at Age = 6 Myr
MISTv.1 Model
(BPASS Model KLF) * 1/20
Observed
stars / (arcsecond^2 mag)
0.0
0.2
0.4
0.6
0.8
1.0
9
10
11
12
13
14
15
Kp magnitude