In [1]:
```
import numpy as np
import scipy as sp
import pandas as pds
import matplotlib
from matplotlib import pyplot as plt
%matplotlib inline
```

collectblsanal is the cross match between the detection and the injected signals

collectblsanal_kois is the cross match between the detection and the real KOIs (regardless it is false positive, EB or transit planet at this stage)

detect_distance -1 means the period distance is larger than 1%.

In [2]:
```
data = pds.read_csv("collectblsanal.csv", index_col=False)
#data = pds.read_csv("collectblsanal_kois.csv", index_col=False, na_values=' nan ')
data = data.dropna()
```

In [3]:
```
data.head()
```

Out[3]:

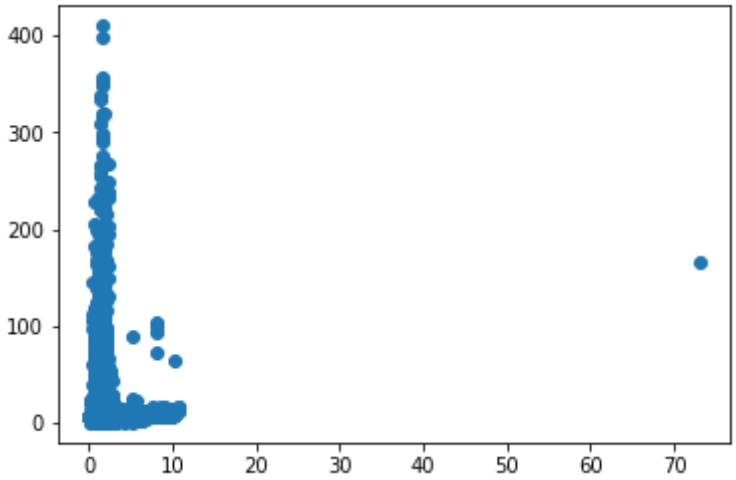
	kepid	period_inj	depth	epoch	MES_expect	MES_scaled	ntransits	BLS_P
0	11651634	12.199	417.0	65.4292	6.5484	1.005293	93.9127	1.3341
1	11651634	12.199	417.0	65.4292	6.5484	1.005293	93.9127	4.2020
2	11651634	12.199	417.0	65.4292	6.5484	1.005293	93.9127	2.9976
3	11651634	12.199	417.0	65.4292	6.5484	1.005293	93.9127	3.7081
4	11651634	12.199	417.0	65.4292	6.5484	1.005293	93.9127	2.6707

5 rows × 28 columns

We have many signals with high detection significance in BLS but not injection, see figure below (this is because the BLS is detecting the KOI signal in the light curve). This is plotting everything, without considering if the detection period is matched or not.

In [4]:
```
plt.scatter(data.MES_scaled, data.BLS_SignaltoPinknoise)
```

Out[4]: <matplotlib.collections.PathCollection at 0x7ff560f5fa90>



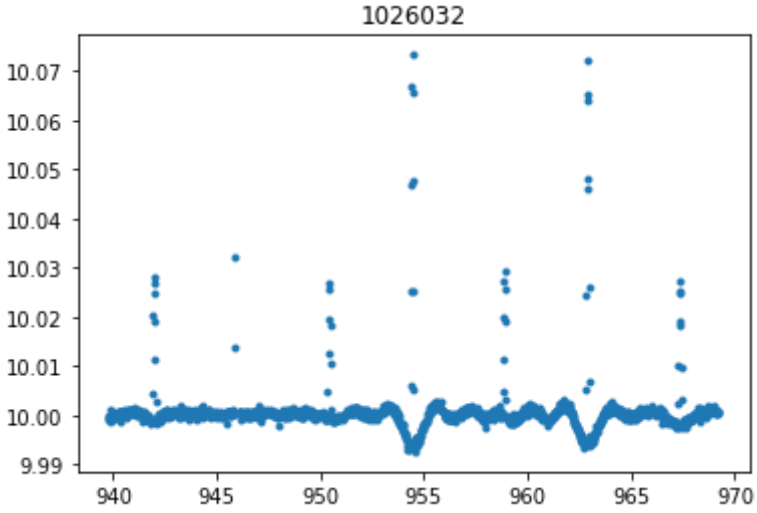
we use a threshold of 5 times bewtween the two signal to noise estimations to identify some of these signals and examine them (note if you switch to the KOI files, the total number of descripancy detections are smaller).

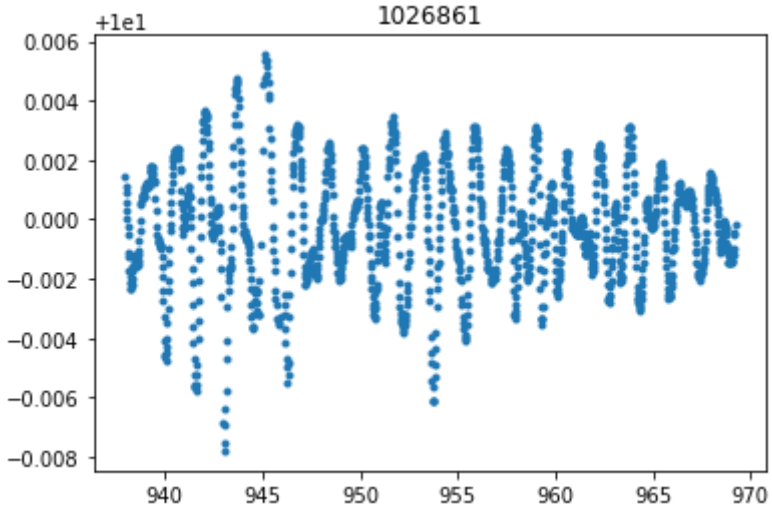
In [5]:
```
select = (data.BLS_SignaltoPinknoise>5*data.MES_scaled) & (data.BLS_Signa
ltoPinknoise>10)
print len(np.unique(np.array(data[select].kepid)))
```

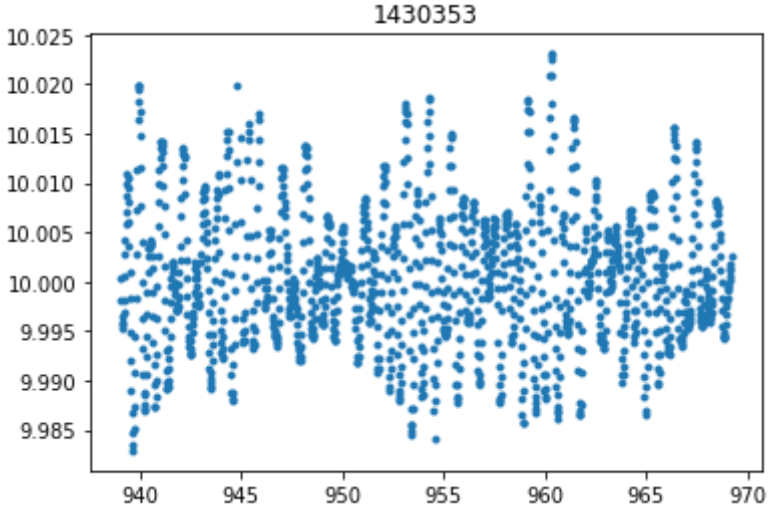
693

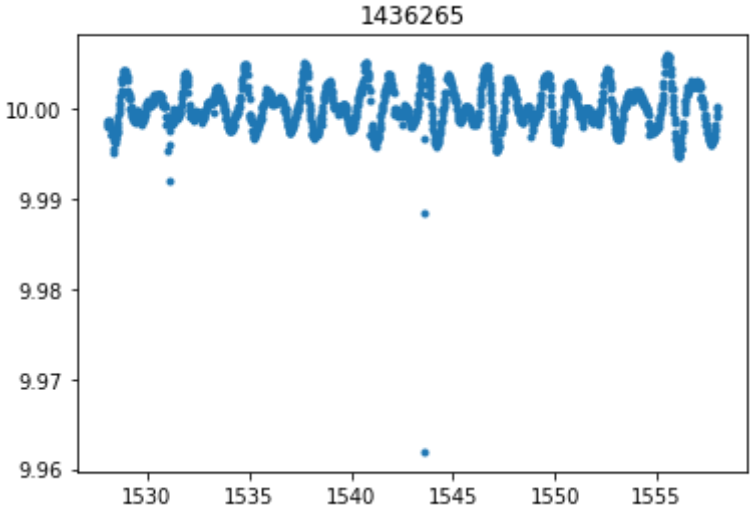
stars like 1026032, 1722276 are EBs/False positives in original KOIs, stars like 1026861, 1430353, 1436265, 1849555, 1865567, 1867078, ... are stellar variability signals that our machine learning algorithm should identify as non-transit. The rest are likely false detections on systematic effects, which hopefully our machine learning algorithm will also pick out.

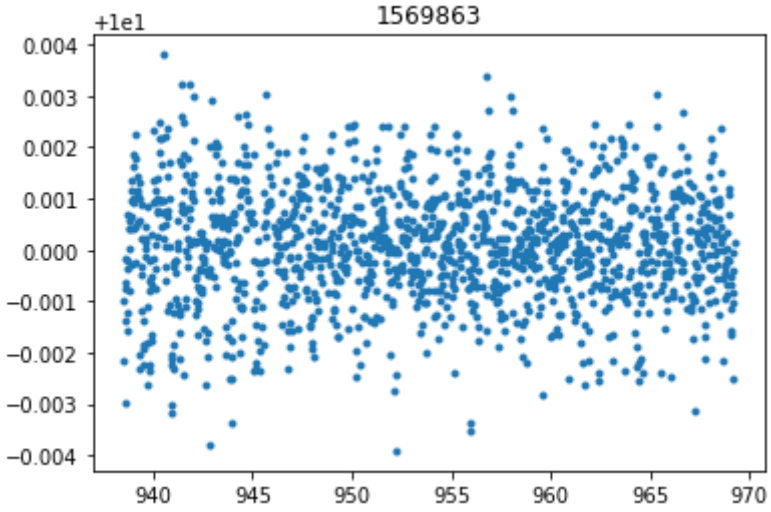
In [6]:
```
count = 0
for kepid in np.unique(np.array(data[select].kepid)):
    if count<0:
        count+=1
        continue
    infile = "../simulation/primaryinj1/kplr%.9d-0_prim_ltf.lc" % int(kep
id)
    lc = np.loadtxt(infile)
    plt.title("%s" % kepid)
    plt.plot(lc[:,0], lc[:,2], '.')
    plt.show()
    count+=1
    if count>10:
        break
```

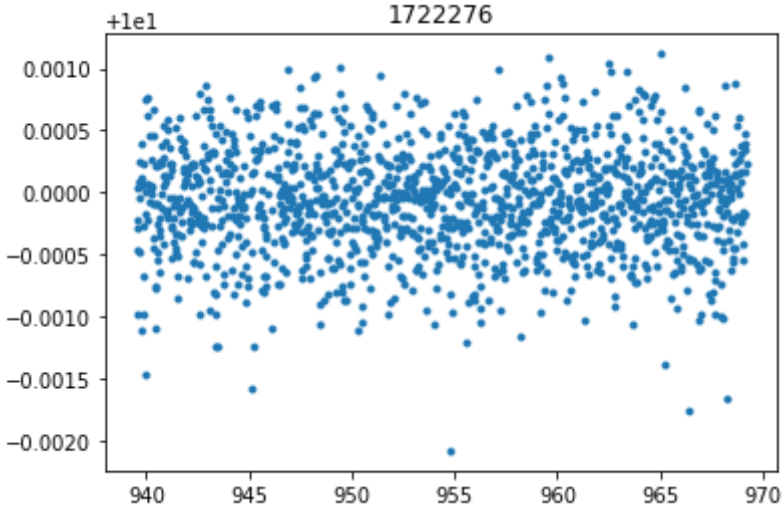

1026032
10.07
10.06
10.05
10.04
10.03
10.02
10.01
10.00
9.99
940
945
950
955
960
965
970

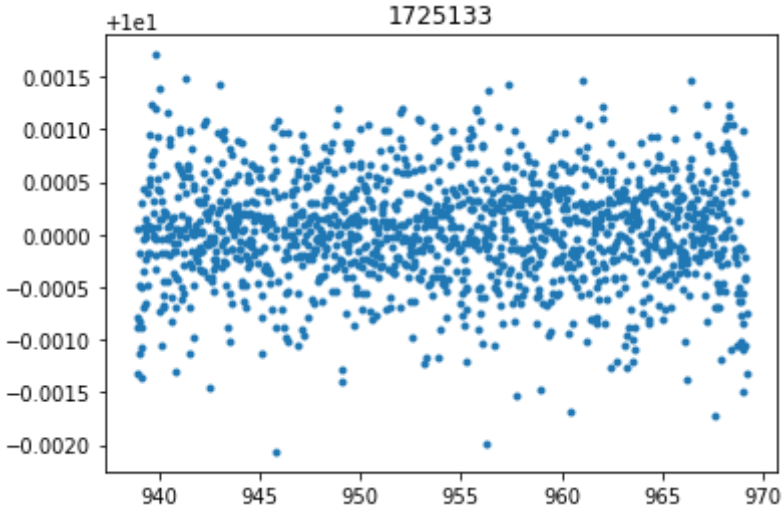

1026861
+1e1
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
−0.008
940
945
950
955
960
965
970

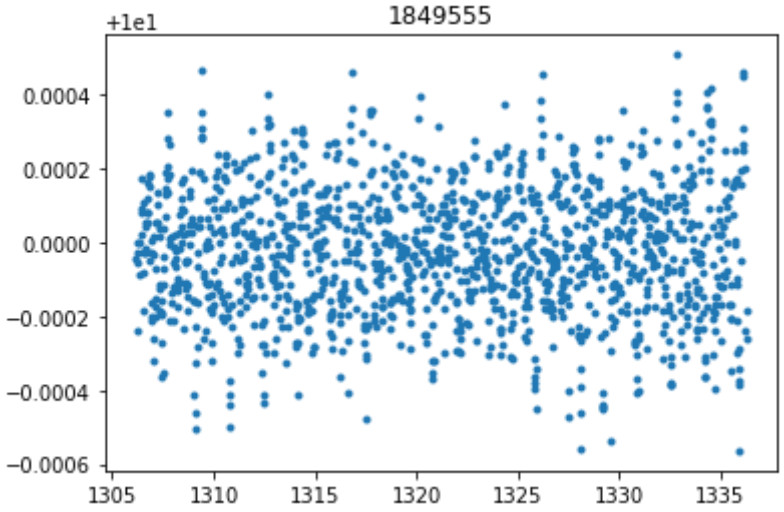

1430353
10.025
10.020
10.015
10.010
10.005
10.000
9.995
9.990
9.985
940
945
950
955
960
965
970

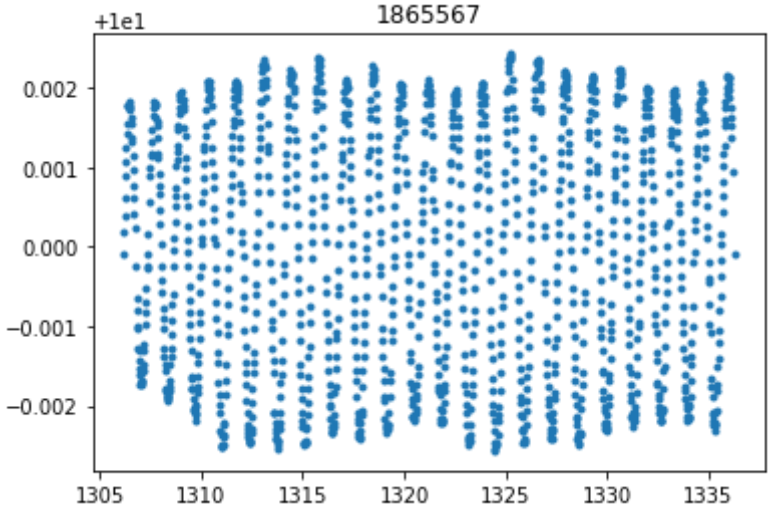

1436265
10.00
9.99
9.98
9.97
9.96
1530
1535
1540
1545
1550
1555

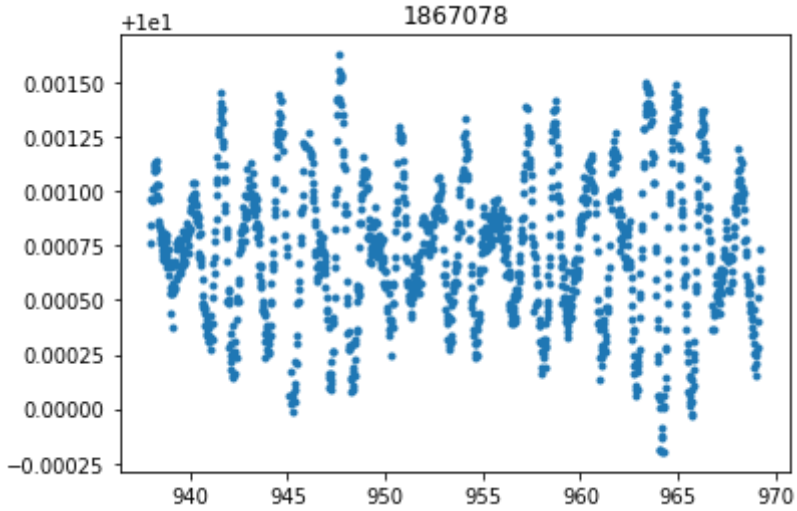

1569863
+1e1
0.004
0.003
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
940
945
950
955
960
965
970

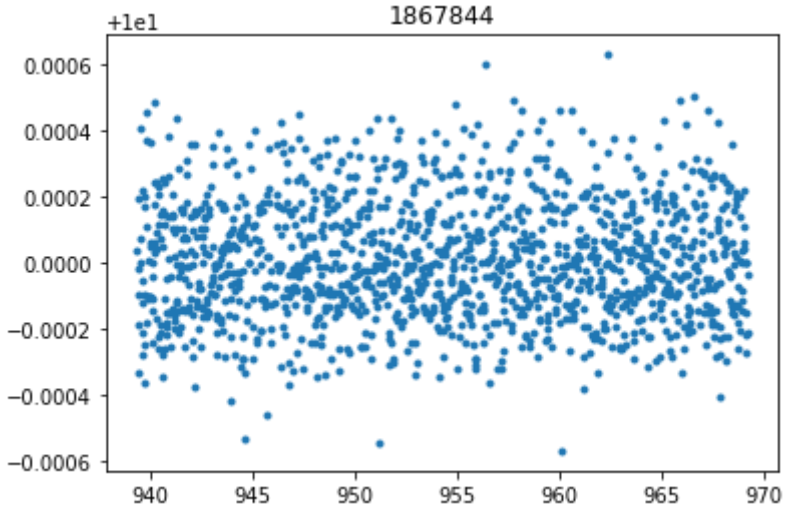

1722276
+1e1
0.0010
0.0005
0.0000
−0.0005
−0.0010
−0.0015
−0.0020
940
945
950
955
960
965
970


1725133
+1e1
0.0015
0.0010
0.0005
0.0000
−0.0005
−0.0010
−0.0015
−0.0020
940
945
950
955
960
965
970


1849555
+1e1
0.0004
0.0002
0.0000
−0.0002
−0.0004
−0.0006
1305
1310
1315
1320
1325
1330
1335

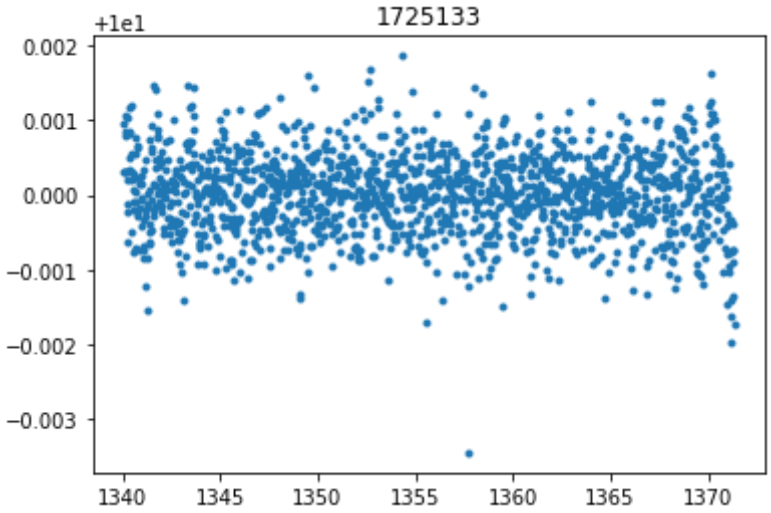

1865567
+1e1
0.002
0.001
0.000
−0.001
−0.002
1305
1310
1315
1320
1325
1330
1335





A little tool to plot the LCs.

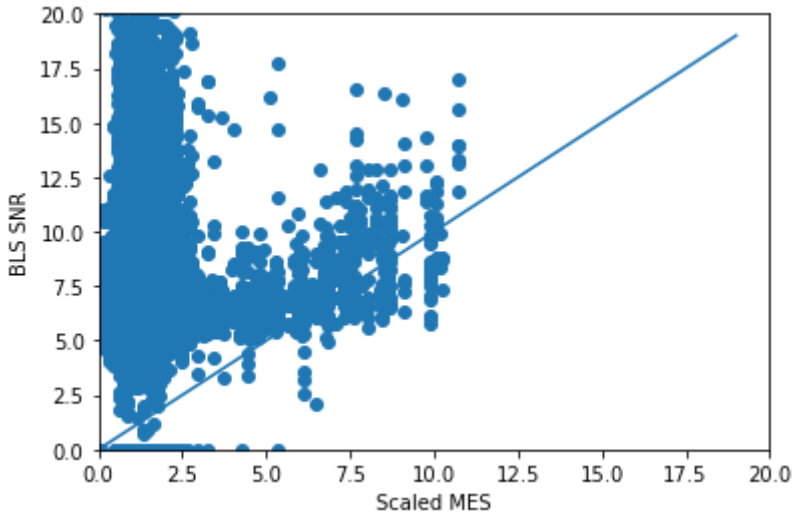
In [7]:
```
kepid = 1725133
infile = "../simulation/primaryinj1/kplr%.9d-3_prim_ltf.lc" % int(kepid)
lc = np.loadtxt(infile)
plt.title("%s" % kepid)
plt.plot(lc[:,0], lc[:,2], '.')
plt.show()
```



A zoomed in look at the comparison between scaled MES and BLS SNR. The arm that lie near the x=y line is the actual detections of the injected signals.

In [8]:
```
plt.scatter(data.MES_scaled, data.BLS_SignaltoPinknoise)
plt.ylim([0,20])
plt.xlim([0,20])
plt.plot(np.arange(20))
plt.xlabel("Scaled MES")
plt.ylabel("BLS SNR")
```

Out[8]: <matplotlib.text.Text at 0x7ff55da1f050>

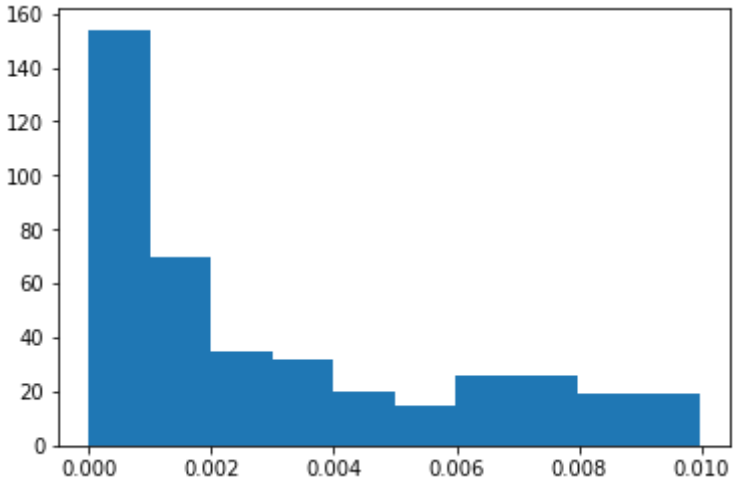


Examine the period distance.

In [9]:
```
detected = data.detect_distance>0
print len(data.detect_distance[detected])
plt.hist(data.detect_distance[detected])
```

```
416
```

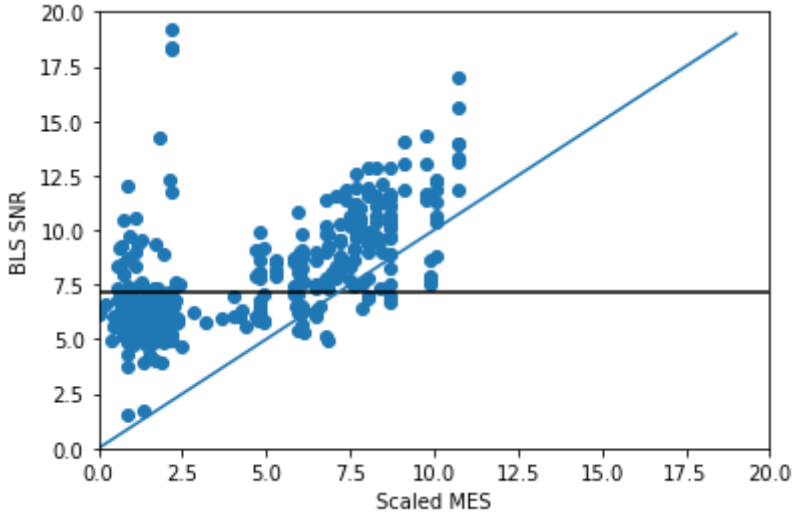
Out[9]:
```
(array([ 154.,   70.,   35.,   32.,   20.,   15.,   26.,   26.,   19.,
         19.]),
 array([  1.20269787e-05,   1.00715568e-03,   2.00228439e-03,
          2.99741309e-03,   3.99254179e-03,   4.98767050e-03,
          5.98279920e-03,   6.97792790e-03,   7.97305661e-03,
          8.96818531e-03,   9.96331401e-03]),
 <a list of 10 Patch objects>)
```



Let's first loosely say everything with period error smaller than 1% is a detection, and reexamine the scaled MES versus BLS plot again, now the vertical feature is removed for the overall catalog.

In [10]:
```
plt.scatter(data.MES_scaled[detected], data.BLS_SignaltoPinknoise[detecte
d])
plt.ylim([0,20])
plt.xlim([0,20])
plt.plot(np.arange(20))
plt.hlines(7.2, 0, 20)
plt.xlabel("Scaled MES")
plt.ylabel("BLS SNR")
```

Out[10]: <matplotlib.text.Text at 0x7ff55dd19a50>



Things to DO:

(1) Merged the two tables, if one peak is detected because of a KOI signal, replace the corresponding line in collectblsanal.csv with the KOI line, notedown this is from KOI in a different column.

(2) create other feature columns for all the peaks.

(3) Generate True/False sample columns

(3a) True sample: signals detected because of injected signals (play with the ScaledSNR cut, period distance) False sample: signals detected because of neither injected/KOI signals signals detected because of KOI signals are removed from this set

(3b) True sample: signals detected because of injected/KOI signals (play with the ScaledSNR cut, period distance)

(4) use previously trained model to test on both (3a) and (3b), not sure if we have enough number of True samples to train on 3a or 3b.

In [ ]:
```
```