cp_hw2

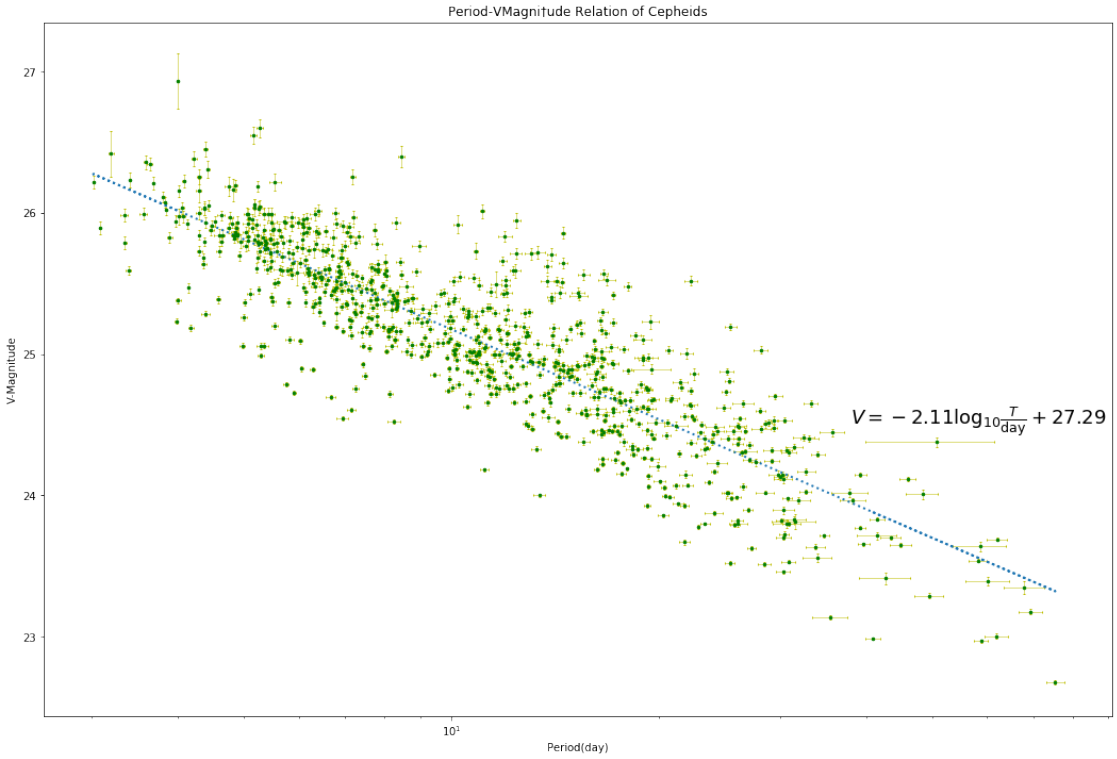
September 18, 2020

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

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

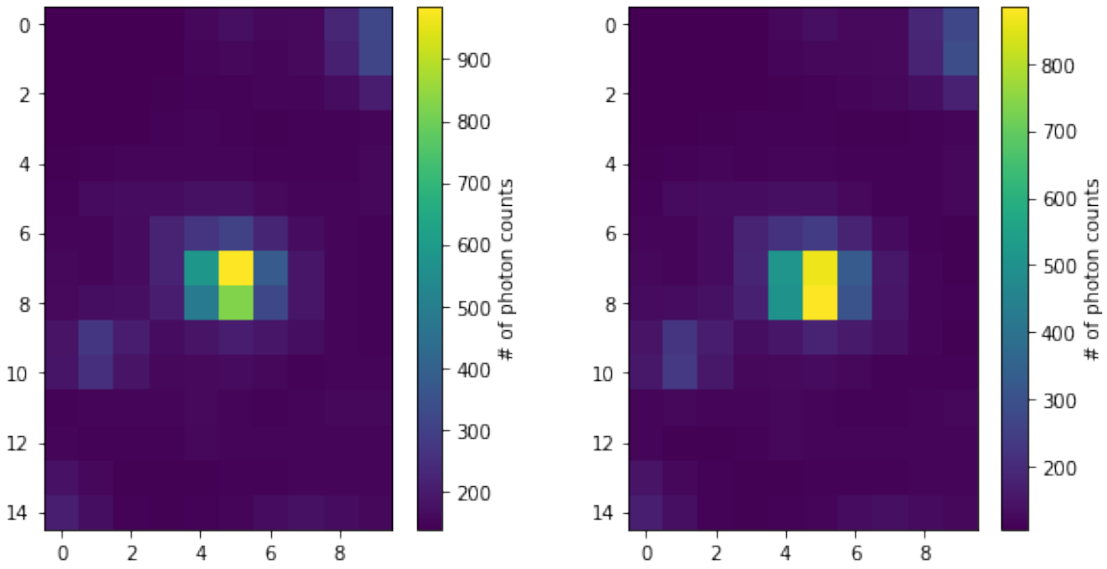
0.1 hw2.1

In [2]:
```
data = np.loadtxt('imgs2/cepheid.dat',float)

per, per_err, vmag, vmag_err = data[:,0],data[:,1],data[:,2],data[:,3]
# fit Vmag by alnT + b
a, b = np.polyfit(np.log10(per),vmag,1)

magfit = a*np.log10(per) + b
plt.figure(figsize=(18,12))
plt.errorbar(per,vmag,xerr=per_err,yerr=vmag_err,fmt='.',lw=0.5,capsize=1.2,markersize=
plt.plot(per,magfit,lw=2,linestyle=':')
plt.text(38, 24.5,'$V = -2.11\log_{10} \\frac{T}{{\\rm day}} + 27.29$',fontsize=18)
plt.xscale('log',)
plt.xlabel('Period(day)')
plt.ylabel('V-Magnitude')
plt.title('Period-VMagniude Relation of Cepheids')
plt.show()
```



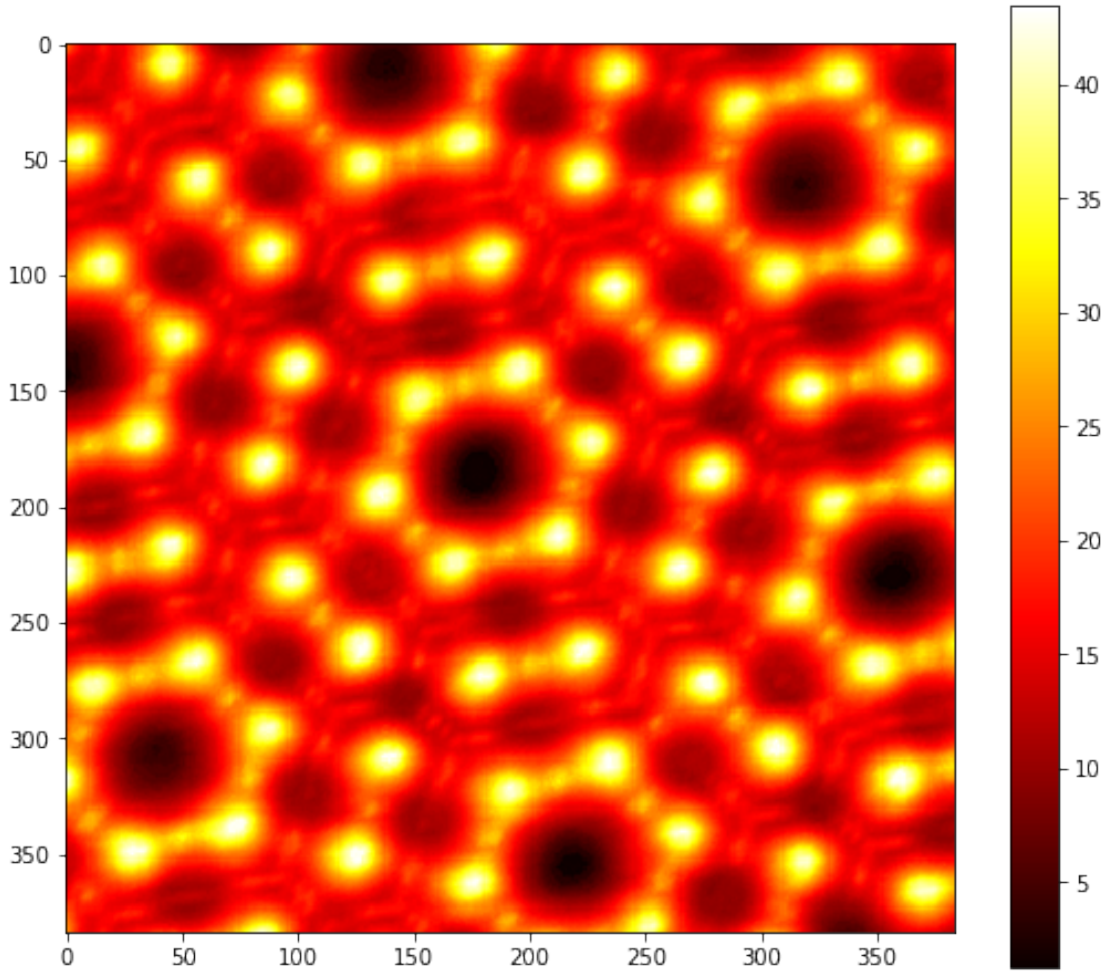
0.2 hw 2.2

```
In [3]: data1 = np.loadtxt('imgs2/star1.dat')
        data2 = np.loadtxt('imgs2/star2.dat')

        plt.figure(figsize=(10,5))
        plt.subplot(121)
        plt.imshow(data1)
        cb = plt.colorbar()
        cb.set_label('# of photon counts')
        plt.subplot(122)
        plt.imshow(data2)
        cb = plt.colorbar()
        cb.set_label('# of photon counts')
        plt.savefig('pc.eps')
```



0.3 hw2.3

```
In [4]: data = np.loadtxt('imgs2/stm.dat.txt')

        plt.figure(figsize=(9,8))
        plt.imshow(data)
        plt.colorbar()
        plt.hot()
        plt.savefig('si111.pdf')
```



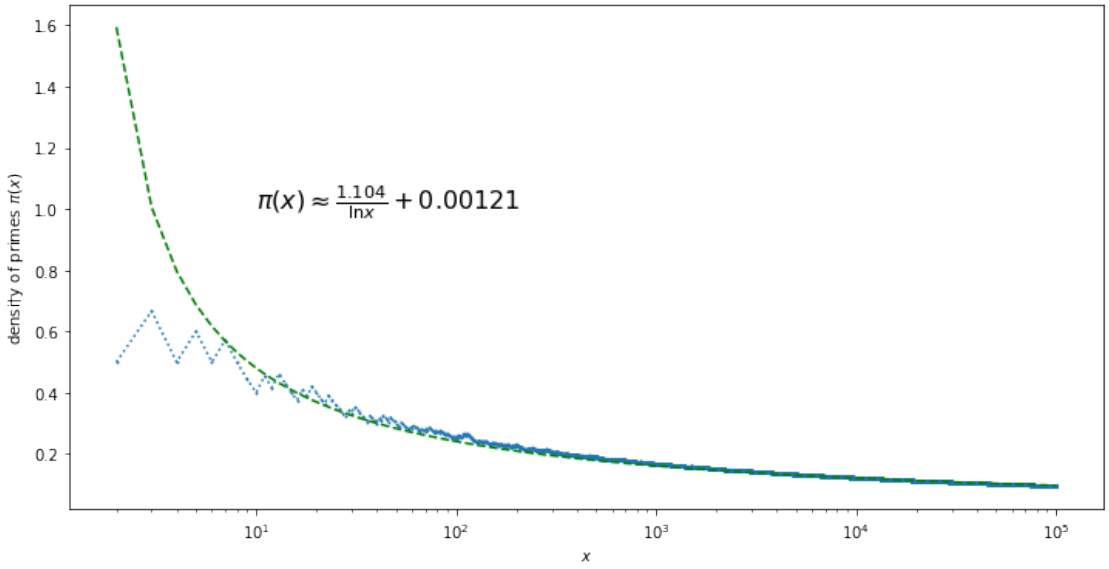
0.4 hw2.4

In [5]:
```
n = 100000
```

```
#borrow from the last hw.
def quicker_is_prime(x,ref):
    det = 1
    for num in ref:
        if num > x**2:
            break
            #break the loop when the prime < sqrt(being tested)
        if x%num == 0:
            det -= 1
            break
    return bool(det)
```

```
        def quicker_prime_list(n):
            primes = [2]
            for i in range(3,n):
                if quicker_is_prime(i,primes):
                    primes.append(i)
            return primes

        # list containing all primes less than n
        primes = quicker_prime_list(n)

        def num_of_primes_less_than(n,ref_lst):
            cnt = 0
            for num in ref_lst:
                if num <= n:
                    cnt += 1
                else:
                    break
            return cnt

        density = np.array([num_of_primes_less_than(j,primes)/j for j in range(2,n+1)])
```

```
In [8]: x = np.array(range(2,100001))
        # fit the density by a/lnx + b
        a, b = np.polyfit(1/np.log(x),density,1)


        #plt.figure(figsize=(12,6))
        #plt.plot(density)
        #plt.show()

        plt.figure(figsize=(12,6))
        plt.plot(x,density,linestyle=':')
        plt.scatter(x,density,s=0.7)
        plt.plot(x,a/np.log(x)+b,color='g',linestyle='--')
        plt.text(10,1,'$\\pi(x) \\approx \\frac{1.104}{\\ln x} + 0.00121$',fontsize=16)
        plt.xscale('log')
        plt.xlabel('$x$')
        plt.ylabel('densiy of primes $\\pi(x)$')
        plt.show()
```


density of primes $\pi(x)$
$\pi(x) \approx \frac{1.104}{\ln x} + 0.00121$
x