Lab 4

Chelsea Seydlitz

Lab Partner: Michael Wang

In [127]:
```
%matplotlib inline
import h5py
import scipy
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

from scipy import stats,signal

mpl.rcParams['agg.path.chunksize'] = 10000

# Global setup for Problem 1
P1 = h5py.File('gammaray_lab4.h5', 'r')
data = np.array(P1.get('data'))
P1.close()
tgps=data[0]
sp=data[1]
el=data[2]
pc=data[3]

# Global setup for Problem 2
P2=h5py.File('images.h5','r')
img1=np.array(P2.get('image1'))
imgstk=np.array(P2.get('imagestack'))
P2.close()
```

Problem 1.1

Details of Note:

- Data is from gamma-ray satellite
- Readings every 100ms
- 90-minute orbit
  - The full data-set maths out to be 4.8 orbits
- Columns are:
  - Time (gps seconds)
    - Accessed with: data[0]
  - Position of Sun ("Solar Phase" (degrees, relative to orbit))
    - Accessed with: data[1]
  - Position of Earth ("Earth Longitude" (degrees, giving position of spacecraft relative to the ground))

        - Accessed with: data[2]
    - Particle Counts
        - Accessed with: data[3]
- Poisson distribution is the most logical choice for background fit because it is non-continuous data over iterations of 100ms.

Next Steps

It is clear that the background will come from the particle count column. As the problem describes, we are searching for gamma rays, but consistently get noise from cosmic rays. We will need to be able to describe the background for this column so that we are able to make conclusions about our observations.

While we will want to show patterns in our other columns and explore the meaning, first I will find a graphical representation of the background distribution for particle count.

First, mu must be found. The math for this is as follows:

In [256]:
```
lnth=len(pc)
tot=0
for i in pc:
    tot+=i
mu1=tot/lnth
print(f"mu={mu1}")
```

```
mu=7.09730524315952
```

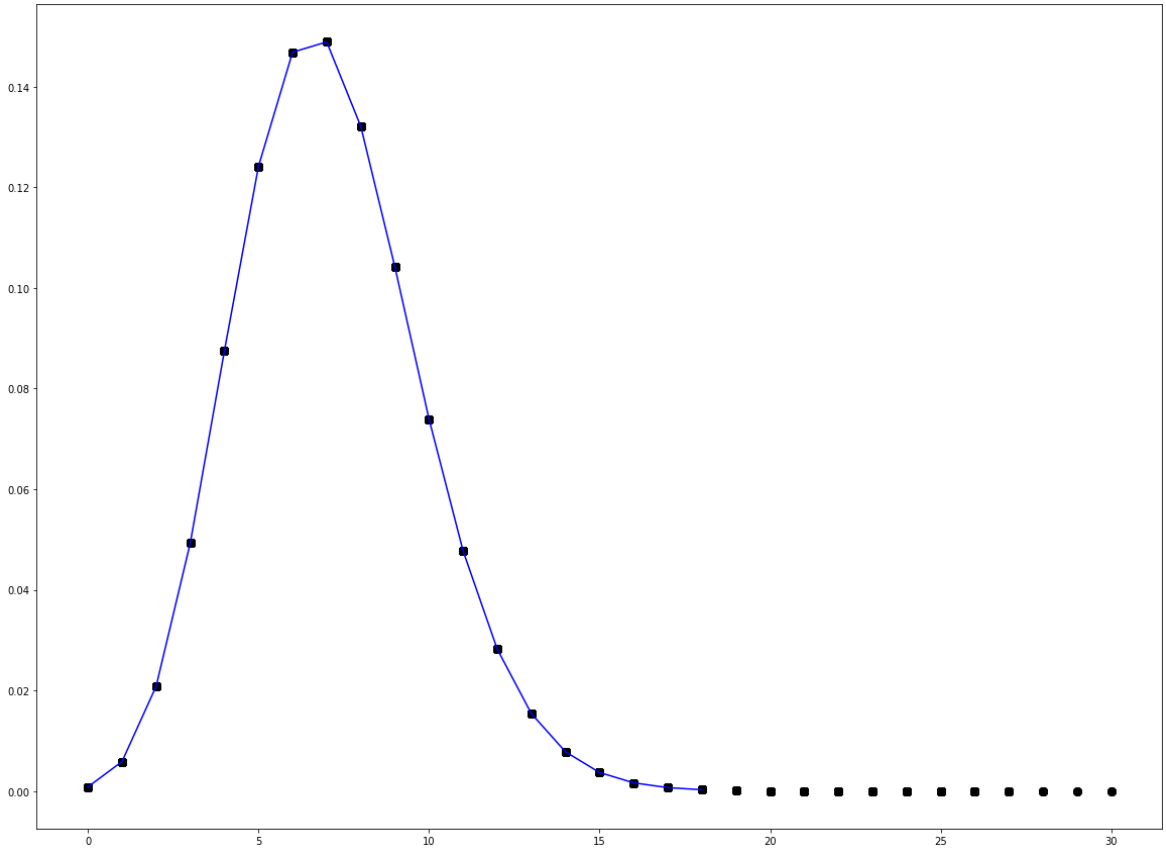
Now with a found mean:

μ = 7.0973

we can create a decent background distribution

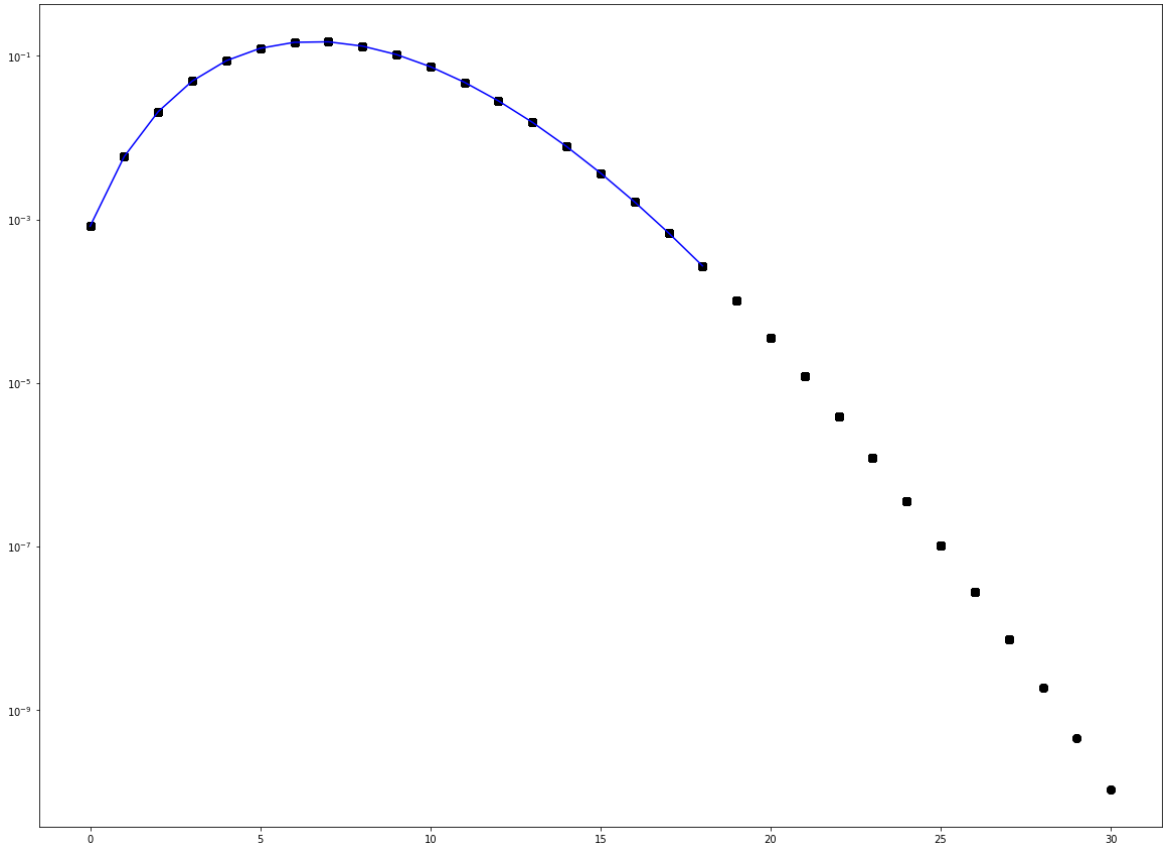
The first graph shows that this is poisson distributed. I've added a best fit line over the top. Due to hardware limitations, I was unable to create a line that fit the distribution with the actual data.

In [156]:
```
fig, ax = plt.subplots(1, 1, figsize=(20,15))
mpl.rcParams['agg.path.chunksize'] = 10000
x=pc
xlin =np.arange(stats.poisson.ppf(.0001,mu1),stats.poisson.ppf(.9999,mu1))
distlin=stats.poisson.pmf(xlin,mu1)
dist=stats.poisson.pmf(x,mu=mu1)
ax.plot(x, dist,'ko',ms=8)
ax.plot(xlin,distlin,'b')
plt.show()
```



The below graph show the log-scale graph of the above distribution. As is similar with the above, I've added a best-fit line.
The hope was to see any outliers present in the data with the log-scale distribution. This would have told us where there might be machine failures or general observation failures. However, the data presents itself as a smooth log representation of the above distribution; no obvious bad data is exposed.

In [157]:
```
fig, ax = plt.subplots(1, 1, figsize=(20,15))
mpl.rcParams['agg.path.chunksize'] = 10000
x=pc
dist=stats.poisson.pmf(x,mu=mu1)
ax.plot(x, dist,'ko',ms=8)
ax.plot(xlin,distlin,'b')
ax.set_yscale('log')
plt.show()
```



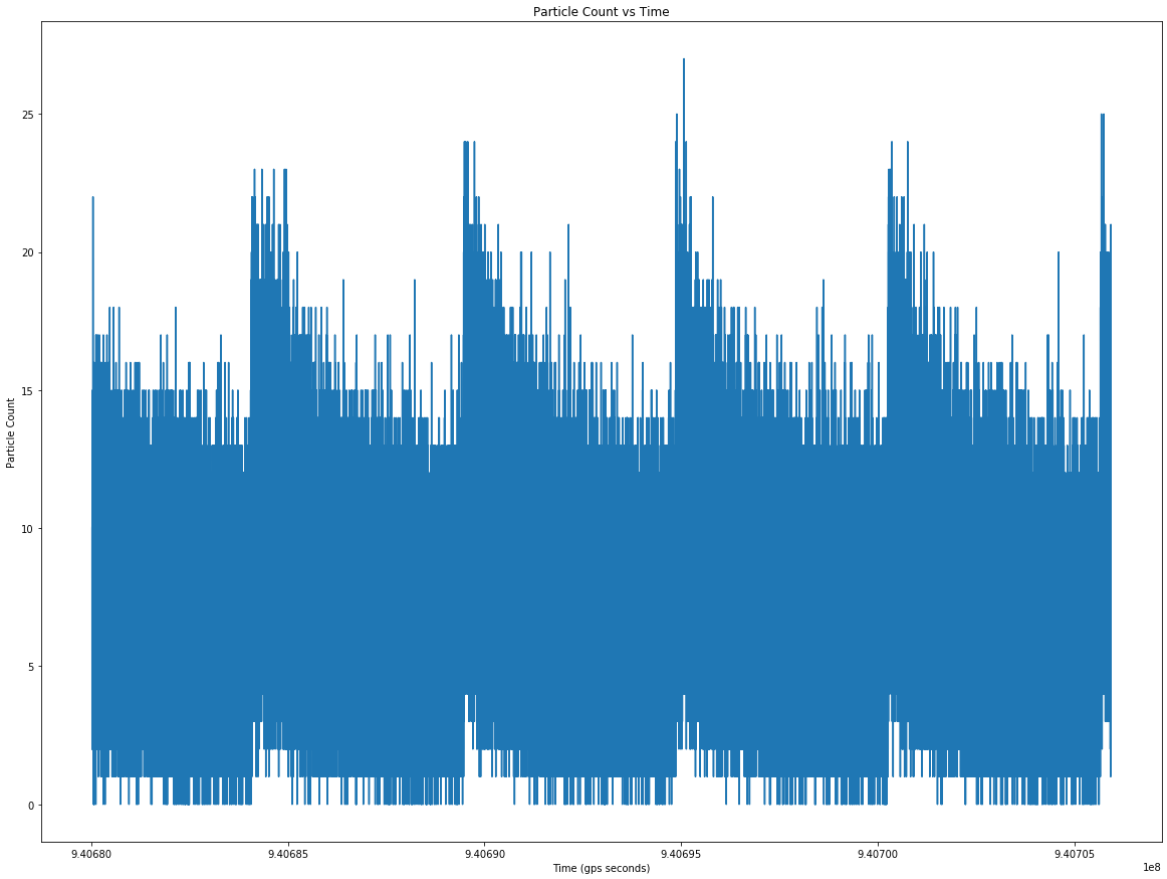
Exploring the Data

Below are three graphs with the Solar Phase, Particle Count, and Earth Longitude graphed against time.

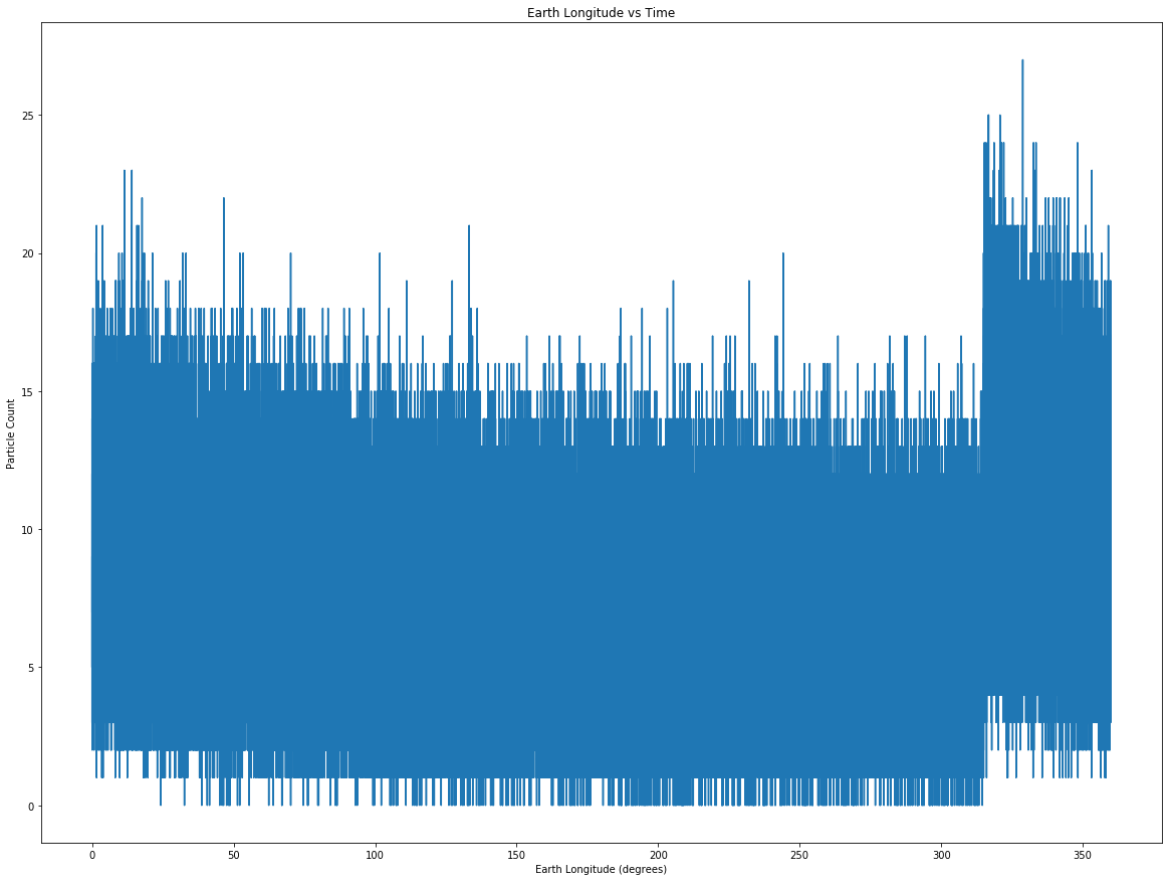
By taking a 1/100th subset of data, we are able to see patterns in the particle data. Note: 1/100th of the data ends up being about 4.32 minutes.

While we do get useful patterns from the data this way, graphing the data in this manner exposed far less than I had hoped. My initial thought was that by graphing the same subset of data over the same iteration, we could see patterns over the visualizations that might indicate a cause in the patterns seen in the particle count.

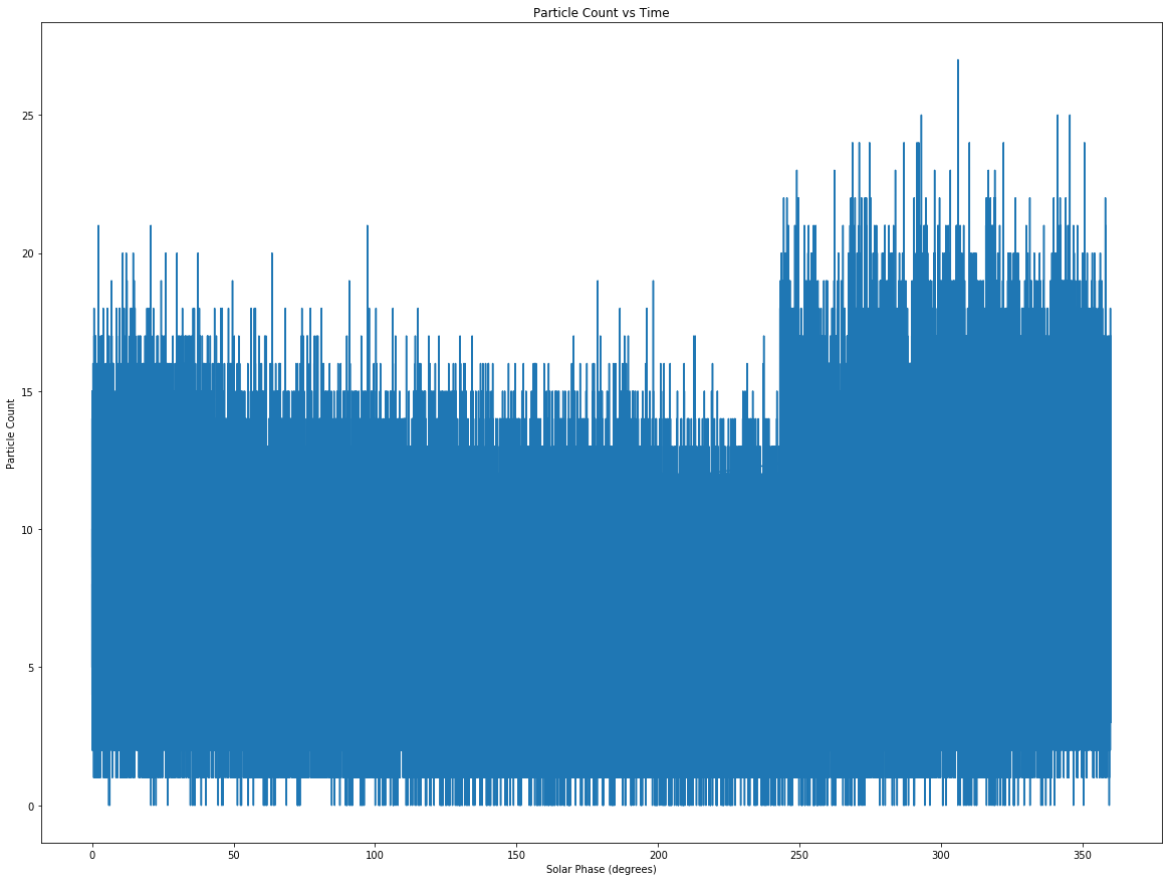
In [158]:
```
fig, ax = plt.subplots(1, 1, figsize=(20,15))
mpl.rcParams['agg.path.chunksize'] = 10000
subset=int(len(tgps)*.01)
ax.plot(tgps[0:subset],pc[0:subset])
ax.set_title("Particle Count vs Time")
ax.set_ylabel("Particle Count")
ax.set_xlabel("Time (gps seconds)")
plt.show()
```



In [160]:
```
fig, ax = plt.subplots(1, 1, figsize=(20,15))
mpl.rcParams['agg.path.chunksize'] = 10000
subset=int(len(el)*.01)
ax.plot(el[0:subset],pc[0:subset])
ax.set_title("Earth Longitude vs Time")
ax.set_ylabel("Particle Count")
ax.set_xlabel("Earth Longitude (degrees)")
plt.show()
```



In [161]:
```
fig, ax = plt.subplots(1, 1, figsize=(20,15))
mpl.rcParams['agg.path.chunksize'] = 10000
subset=int(len(sp)*.01)
ax.plot(sp[0:subset],pc[0:subset])
ax.set_title("Particle Count vs Time")
ax.set_ylabel("Particle Count")
ax.set_xlabel("Solar Phase (degrees)")
plt.show()
```



Problem 1.2

Looking for Inconsistencies

We are looking to compare and describe inconsistencies across the data. Even though the whole data set shows as a nice Poisson distribution, upon closer inspection, across some interval, we may find inconsistencies.

Based on the graph comparing particle count to time, we can see that there is some period of decay that is 90 minutes. This could be a good interval to use when looking for inconsistencies.

Path Forward:

There are 54000-100ms intervals in 90 minutes, to show the intervals, we'll put a line at every 54000 interval. Will will omit the very first interval start since we don't know how much data has been cut off.

We can get a good estimate of where our lines should be by saying if the particle equals or exceeds 20, this is where the beginning of our interval begins. To find a good starting point:

In [235]:
```
t0=0
for i in range(5000,500000):
    if pc[i]>=20:
        print(f"Time index the first time we encounter a particle count of 20
        print(f"Time at index {i}: {tgps[i]}")
        t0=i
        break

xposition=[]
for i in range(0,5):
    xposition.append(t0+(54000*i))
```

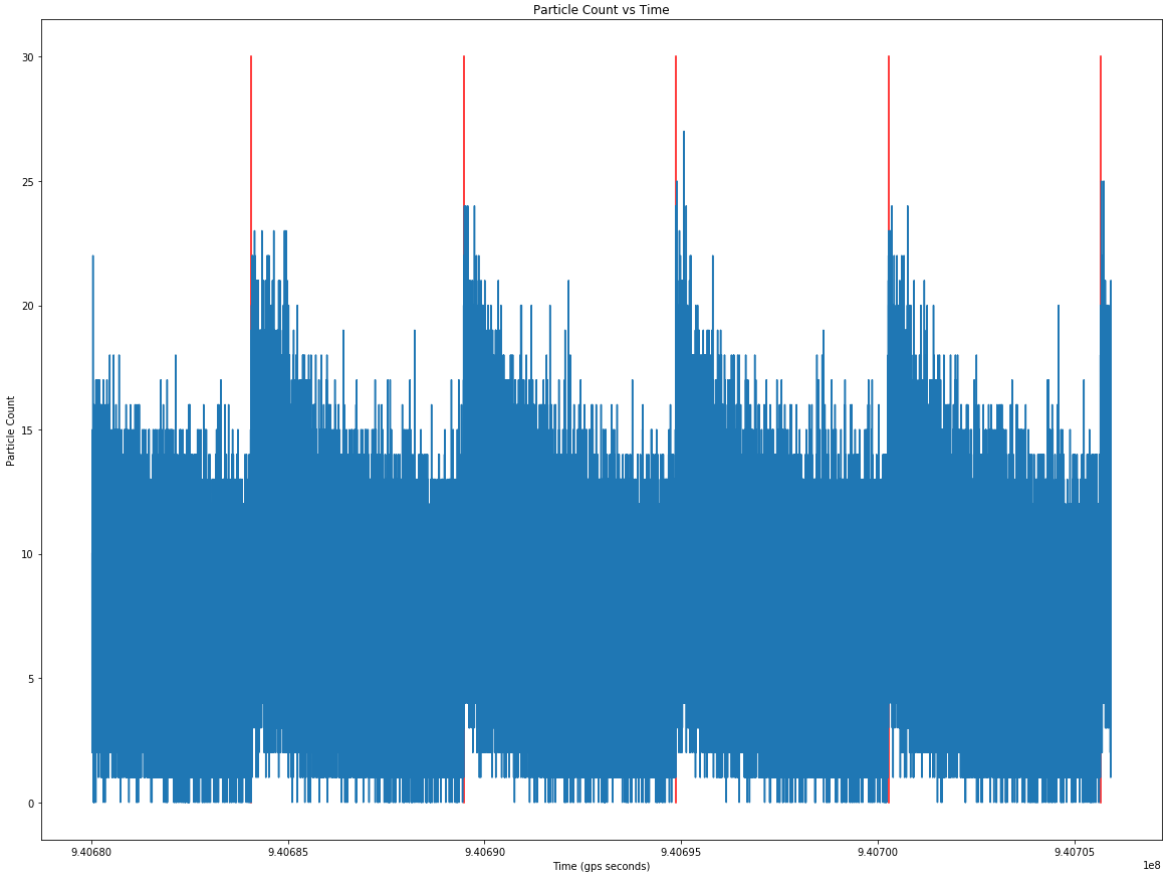
```
Time index the first time we encounter a particle count of 20 or more: 4057
2
Time at index 40572: 940684073.2
```

Proving the Interval:

So now with a starting point and the next 3 intervals (each 90 minutes apart), we have the following visualization, which gives us a level of confidence behind the statement: the period is 90 minutes.

In [236]:

```
fig, ax = plt.subplots(1, 1, figsize=(20,15))
mpl.rcParams['agg.path.chunksize'] = 10000
subset=int(len(tgps)*.01)

for i in xposition:
    ax.plot((tgps[i],tgps[i]),(0,30),'r-')

ax.plot(tgps[0:subset],pc[0:subset])
ax.set_title("Particle Count vs Time")
ax.set_ylabel("Particle Count")
ax.set_xlabel("Time (gps seconds)")
plt.show()
```



Using the Intervals:

Now that we've proven our guess that each visual period is 90 minutes (the time it takes for the instrument to orbit the Earth), we can use the interval to explore inconsistencies.

Below I'll find the average for each section, as well as create the individual sections, in order to create distributions.

In [178]:
```
averages=[]
time=[]
count=0
sect1=[]
sect2=[]
sect3=[]
sect4=[]
sects=[sect1,sect2,sect3,sect4]

for i in range(1,45):
    time.append(i)
    start=(40572+(54000*count))
    pcss=pc[start:]

    tot=0
    for j in range (0,54000):
        tot+=pcss[j]
    if count<4:
        sects[count].append(pcss[start:start+54000])
    averages.append(tot/54000)
    count+=1
```

Problem 1.3

Using the data from above, we can take a 90-minute subsection and plot the ideal background (with mean of 7.1) over the top to see how good the fit is. I will use the first full interval, [40572,94572]

In [189]:
```
xlin=np.arange(stats.poisson.ppf(.0001,7.1),stats.poisson.ppf(.9999,7.1))
distlin=stats.poisson.pmf(xlin,mu=7.1)
```

In [*]:
```
fig, ax = plt.subplots(1, 1, figsize=(8,8))
x=pcss[40572,94572]
dist=stats.poisson.pmf(x,mu=7.1)
ax.plot(xlin,distlin,'r')
ax.hist(dist,500)
plt.show()
```

Problem 1.4

Problem 2.1

Note: I will be looking for a transient signal

Problem 2.2

Plan Forward

To calculate my background pdf, I believe I will want to convolve over all 10 images instead of finding a background with all datapoints individually.

To do that, I will pull out all 10 images:

In [121]:
```
valsrows=[]

for i in imgstk:
    for j in i:
        valsrows.append(j)
row0=[]
row1=[]
row2=[]
row3=[]
row4=[]
row5=[]
row6=[]
row7=[]
row8=[]
row9=[]
alldata=[]

lists=[row0,row1,row2,row3,row4,row5,row6,row7,row8,row9]

for vrow in valsrows:
    for i in range(0,10):
        lists[i].append(vrow[i])
        alldata.append(vrow[i])
```

In [122]:

```
avgs=[]

tot=0
for i in row0:
    tot+=i
avgs.append(tot/len(row0))


tot=0
for i in row1:
    tot+=i
avgs.append(tot/len(row1))


tot=0
for i in row2:
    tot+=i
avgs.append(tot/len(row2))


tot=0
for i in row3:
    tot+=i
avgs.append(tot/len(row3))


tot=0
for i in row4:
    tot+=i
avgs.append(tot/len(row4))


tot=0
for i in row5:
    tot+=i
avgs.append(tot/len(row5))


tot=0
for i in row6:
    tot+=i
avgs.append(tot/len(row6))


tot=0
for i in row7:
    tot+=i
avgs.append(tot/len(row7))


tot=0
for i in row8:
    tot+=i
avgs.append(tot/len(row8))
```

```
tot=0
for i in row9:
    tot+=i
avgs.append(tot/len(row9))

tot=0
for i in alldata:
    tot+=i

print(f"Averages of each image: {avgs}")
```

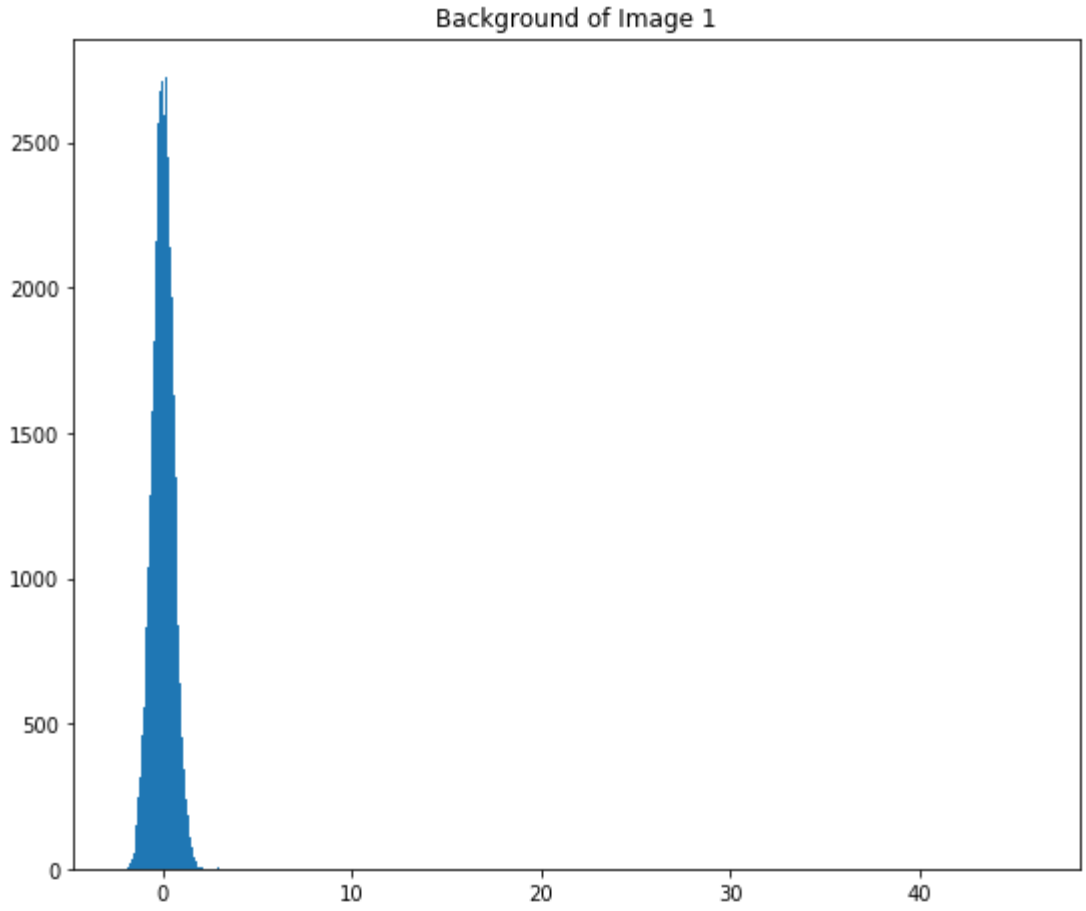
```
Averages of each image: [0.030218566576240977, 0.030805866173257236, 0.0380
4687710376932, 0.03359187121452354, 0.03441548974633696, 0.0381426778965341
1, 0.028322878663590928, 0.0318465931835549, 0.03324503541571198, 0.0358017
9806387]
```

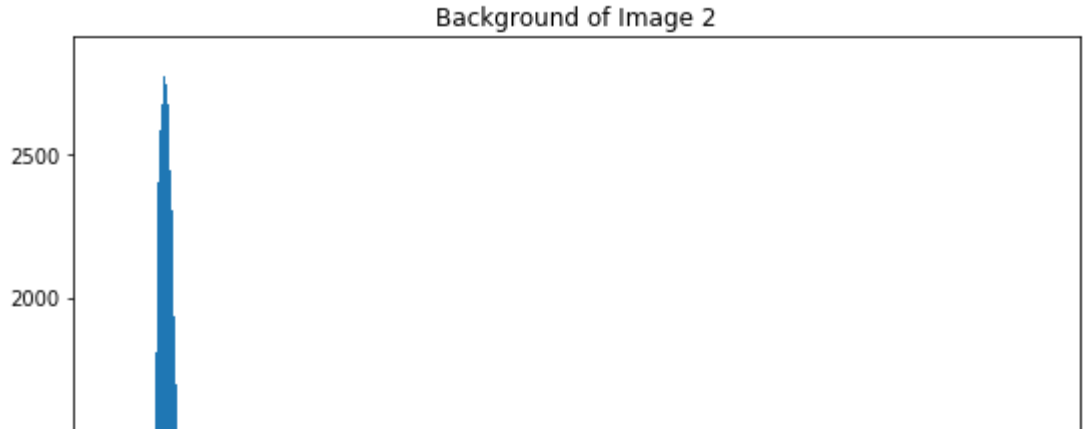
For further visualization to better understand the data, I have graphed the distribution and the log-scale background of each image.

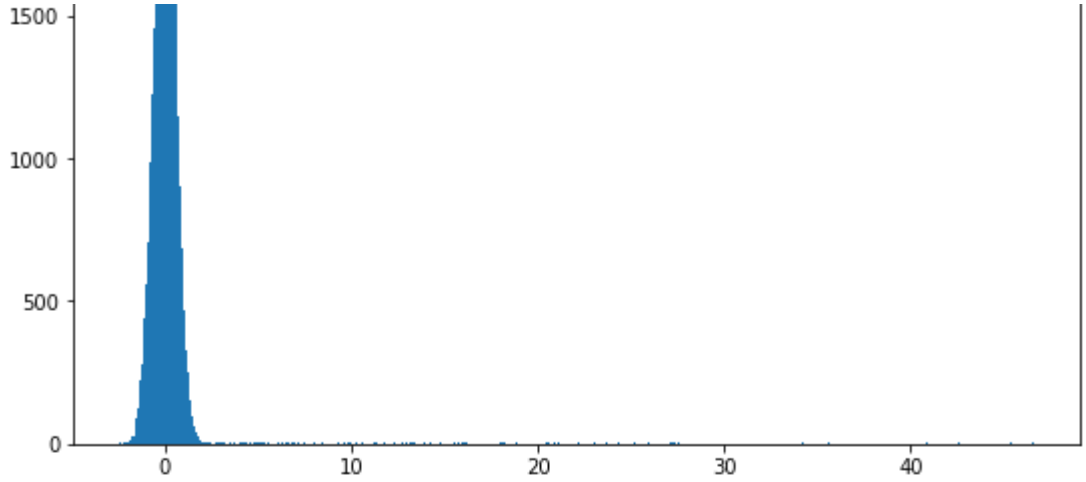
On the left column, I've added a best fit for gaussian-distributed data. It appears to be close but slightly skewed, and it is difficult to tell if the background is gaussian.

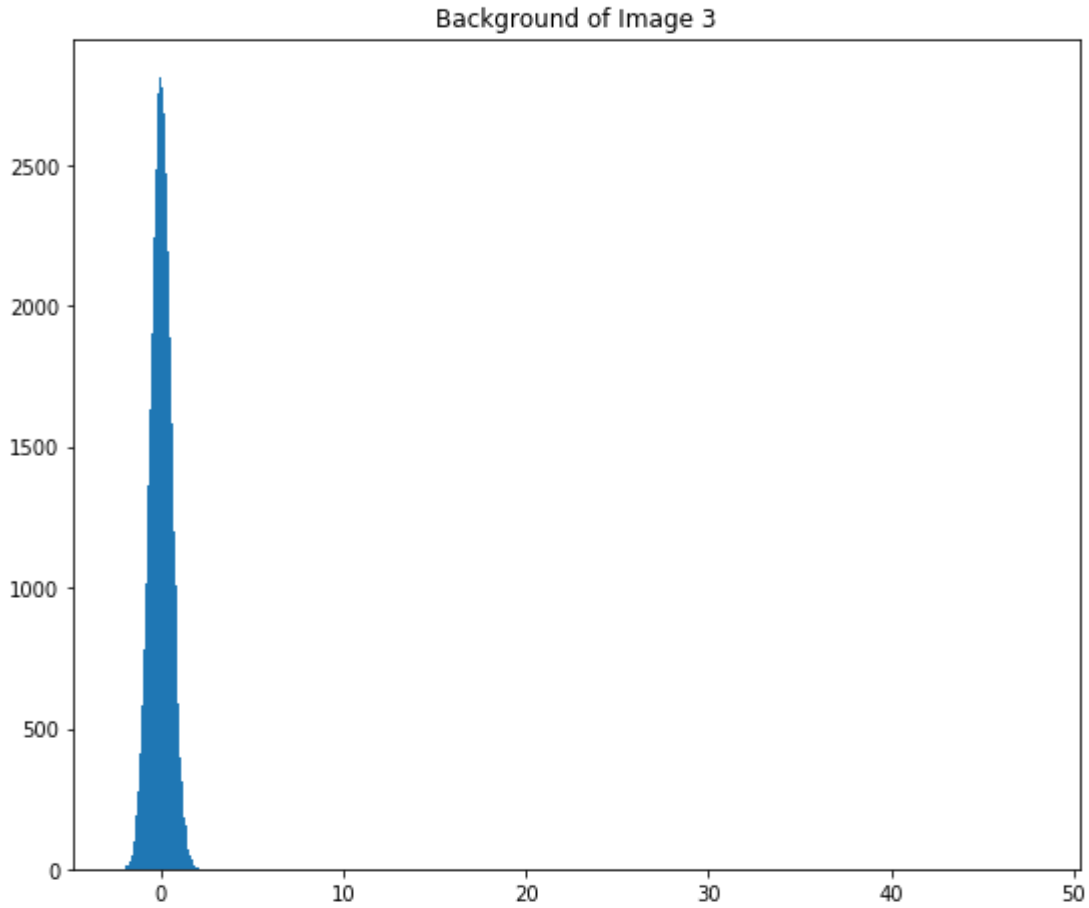
In [159]:

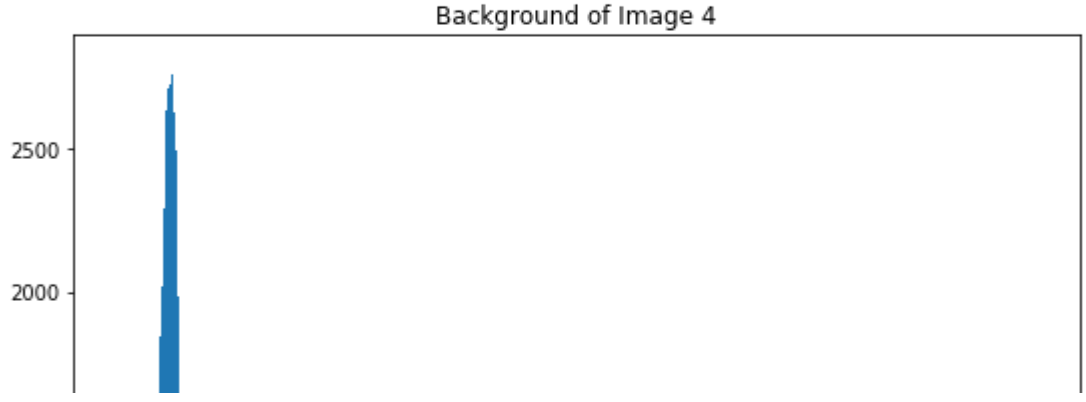
```
fig, ax = plt.subplots(9, 1, figsize=(9,81))
mpl.rcParams['agg.path.chunksize'] = 10000
listofdist=[]

for i in range(1,10):
    avg=avgs[i-1]
    x=lists[i-1]
    dist=stats.norm.pdf(x,loc=avg,scale=1)
    listofdist.append(dist)
    ax[i-1].hist(x,500)
    ax[i-1].set_title(f"Background of Image {i}")

plt.show()
```

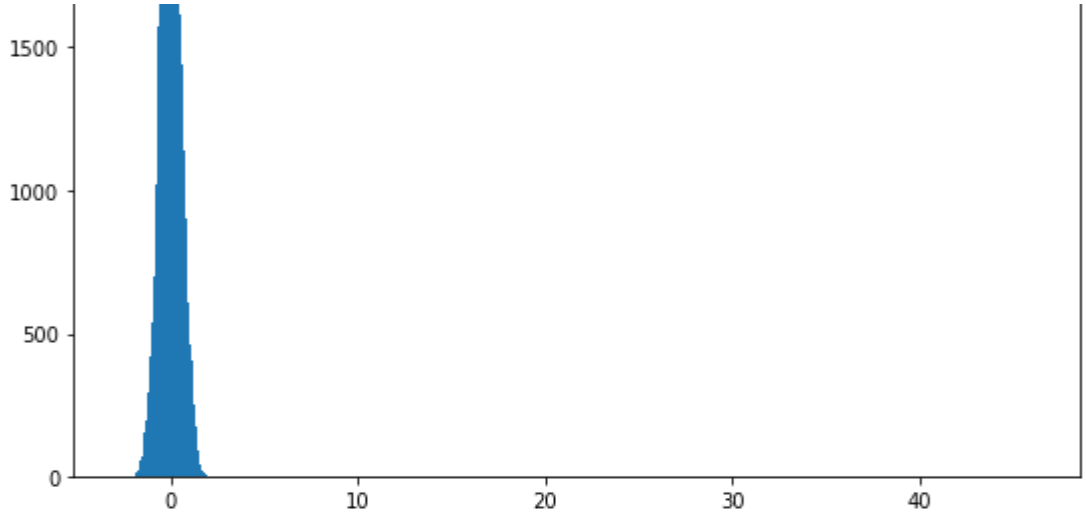


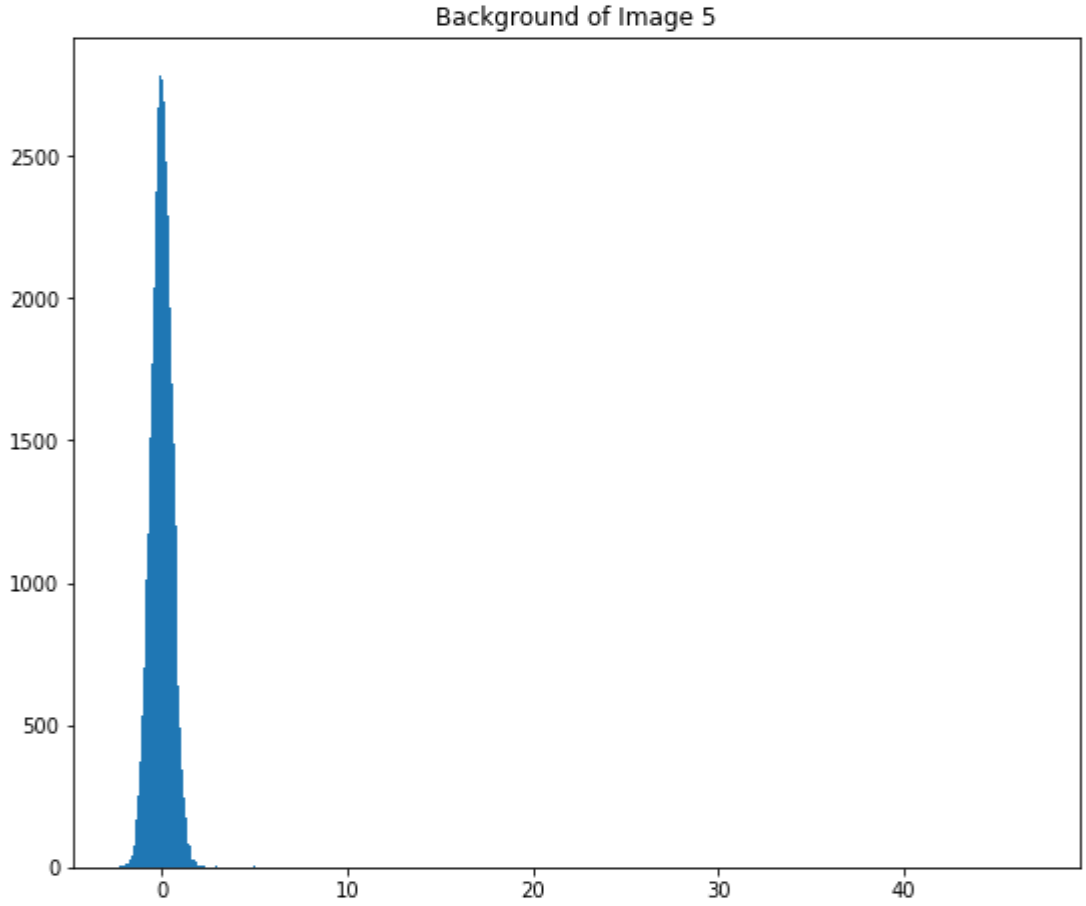


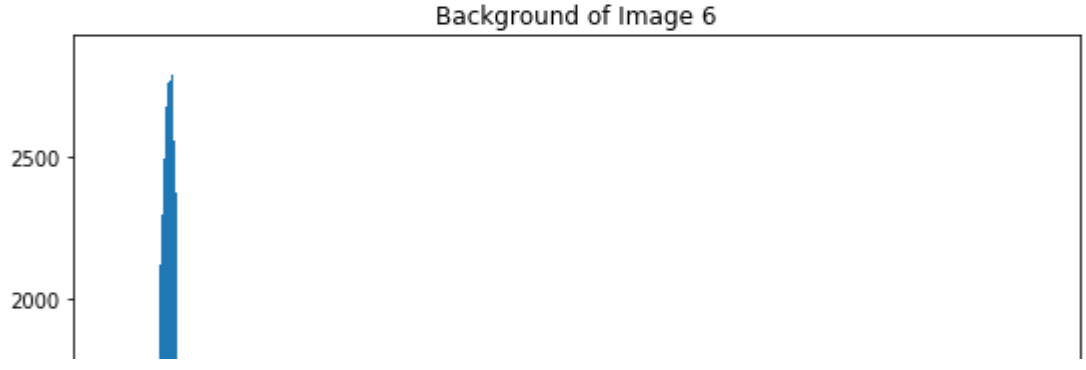

1500
1000
500
0
0
10
20
30
40

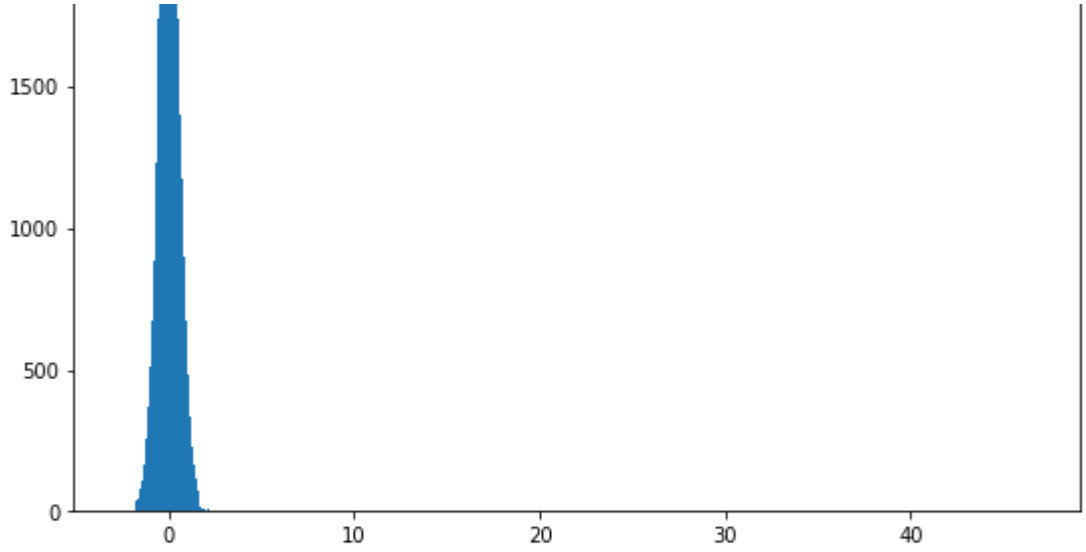

Background of Image 3
2500
2000
1500
1000
500
0
0
10
20
30
40
50

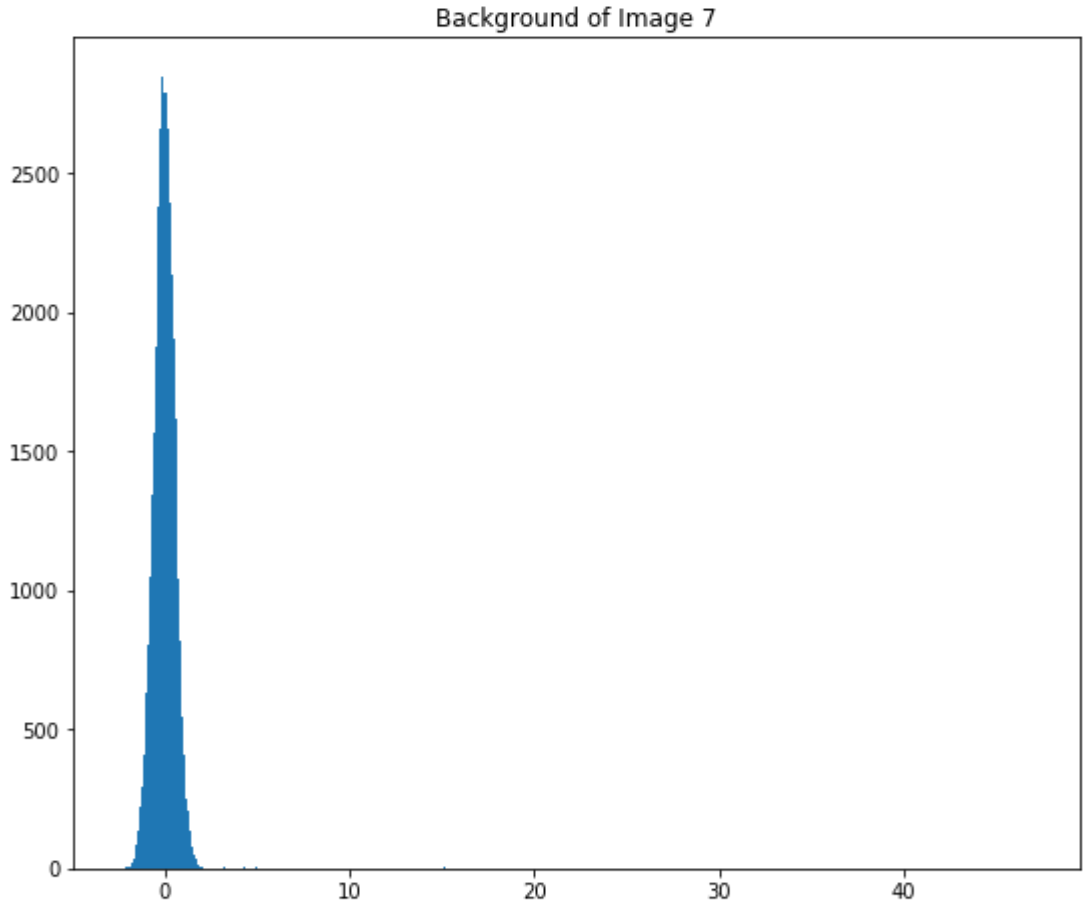

Background of Image 4
2500
2000

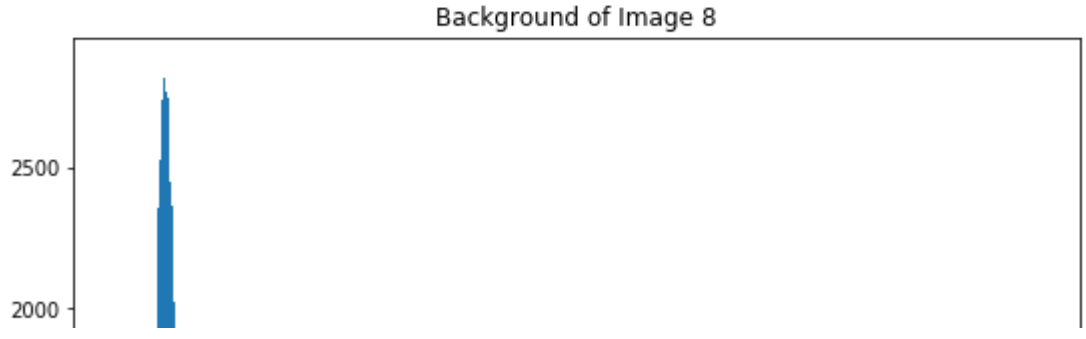

1500
1000
500
0
0
10
20
30
40

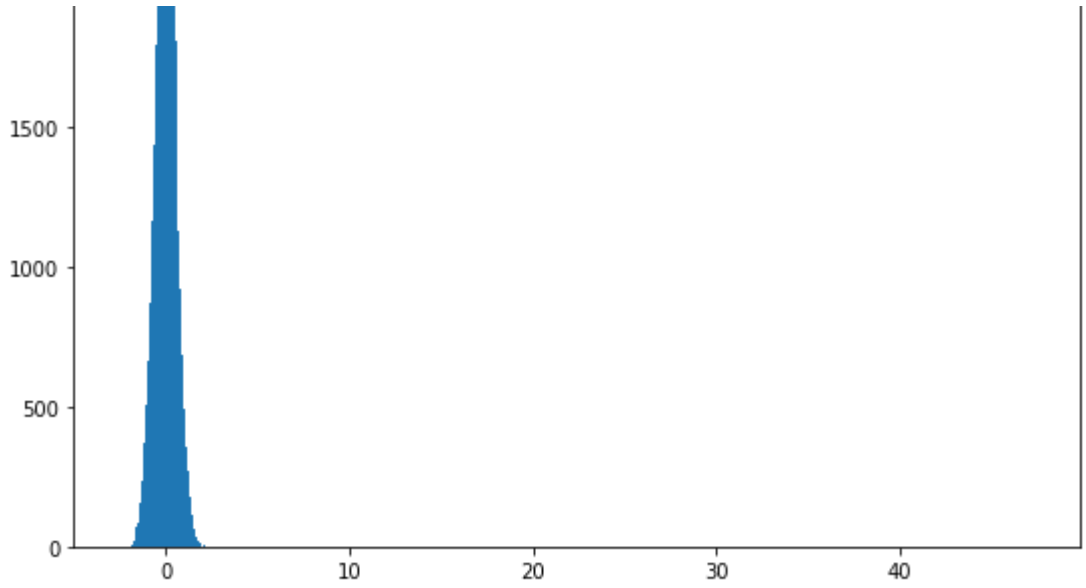

Background of Image 5
2500
2000
1500
1000
500
0
0
10
20
30
40

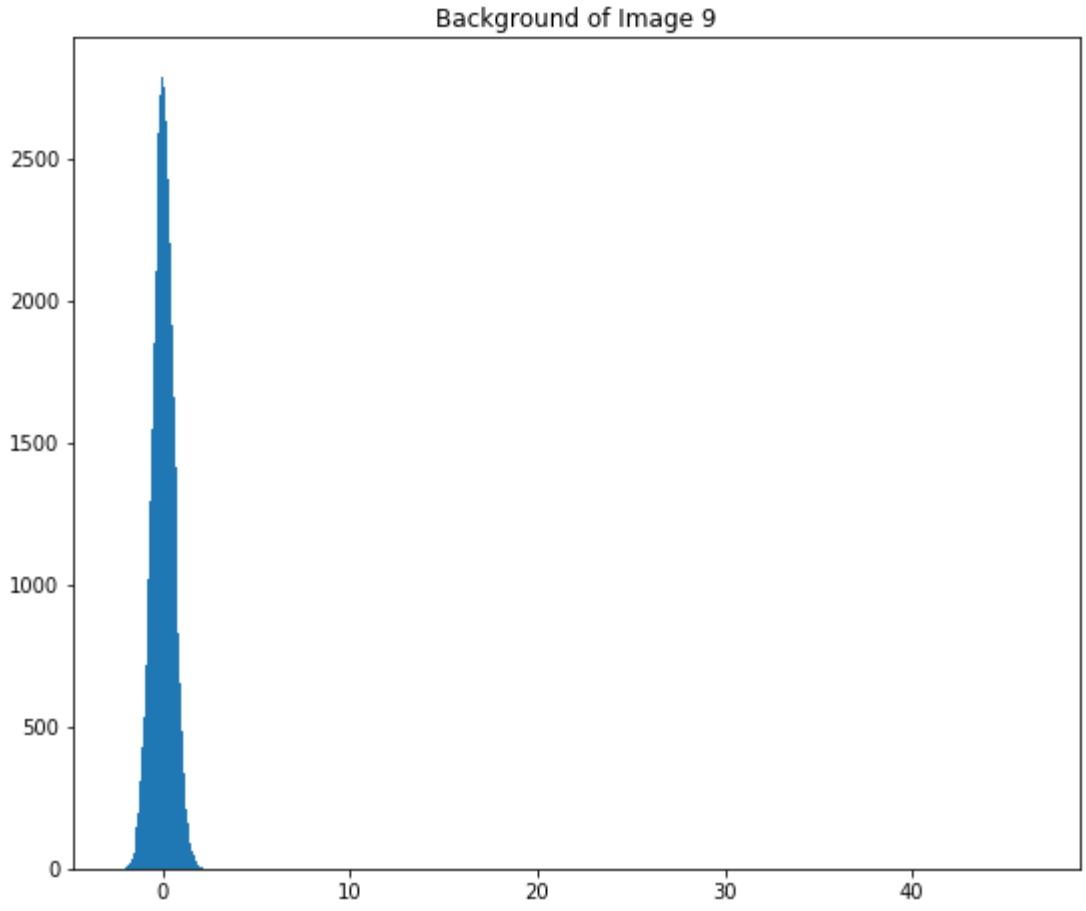

Background of Image 6
2500
2000


1500
1000
500
0
0
10
20
30
40


Background of Image 7
2500
2000
1500
1000
500
0
0
10
20
30
40


Background of Image 8
2500
2000



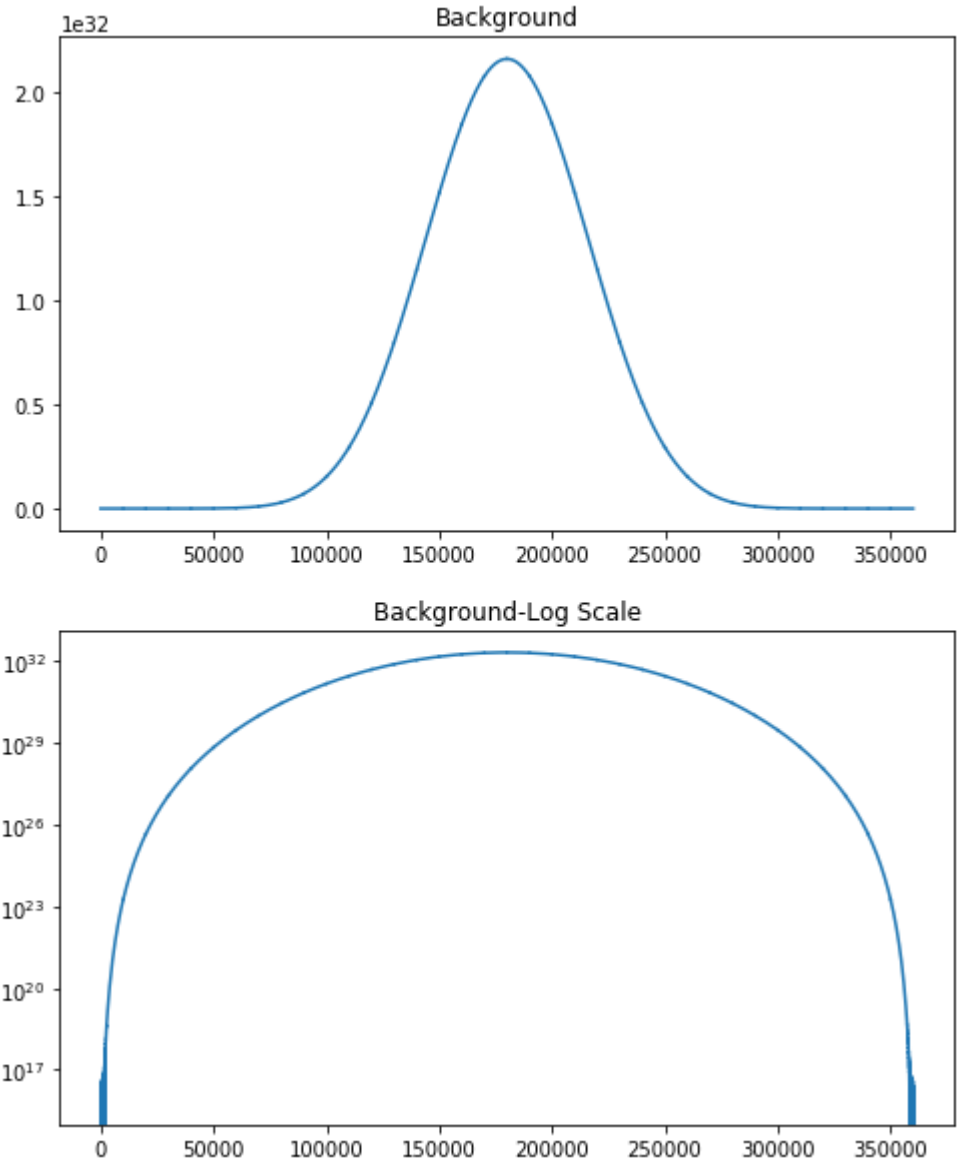


The background is not time dependent in this situation.

It looks like there is significant contamination that are present in the data. If we look at several images, the histogram has picked up signals beyond what one would expect to be the threshold for 5-sigma.

Image two is particularly interesting because of how much data is to the right of the distribution.

In [148]:
```
fig,ax=plt.subplots(2,1,figsize=(8,10))

conv=signal.convolve(listofdist[0],listofdist[1])
x=np.linspace(0,len(conv),len(conv))

for i in range(2,9):
    convIter=signal.convolve(conv,listofdist[i])
    conv=convIter
    lin=np.linspace(0,len(conv),len(conv))
    x=lin

ax[0].plot(x,conv)
ax[0].set_title("Background")
ax[1].plot(x,conv)
ax[1].set_yscale('log')
ax[1].set_title("Background-Log Scale")
plt.show()
```



The above shows that this is not Gaussian. This might suggest significant contamination.

Problem 2.3

In [ ]:

Problem 2.4

The reason why we had different PDFs is because we were looking for signals in different ways. I was looking for a signal that would only occur in one image, while he was looking for faint signals across the whole set.

One problem required, essentially, a difference in background; what occurred in one image that did not occur in the others. In addition, this problem was to identify a signal that was signal like or more signal like than the rest of the data.

The other problem required to combine backgrounds and then finding a signal that is less than signal like as compared to the other signals.

In [ ]: