MCMC Final

August 6, 2021

Numpy Documentation: https://numpy.org/devdocs/user/absolute_beginners.html

[1]:
```python
#the SIGMA values
import numpy as np
sigma_ones = np.ones(20)
print(sigma_ones)
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

[2]:
```python
#data x-values
import numpy as np
xarray = np.linspace(1,30,20)
print(xarray)
```

```
[ 1.          2.52631579  4.05263158  5.57894737  7.10526316  8.63157895
 10.15789474 11.68421053 13.21052632 14.73684211 16.26315789 17.78947368
 19.31578947 20.84210526 22.36842105 23.89473684 25.42105263 26.94736842
 28.47368421 30.        ]
```

[3]:
```python
#generates model y-values
def function(xarray,m,b):
    '''
    signature: list,int, int -> list
    given a set of x values, a slope, and a y-int returns a
    list of y values
    '''
    import numpy as np
    y_model= xarray * m + b
    return y_model
```

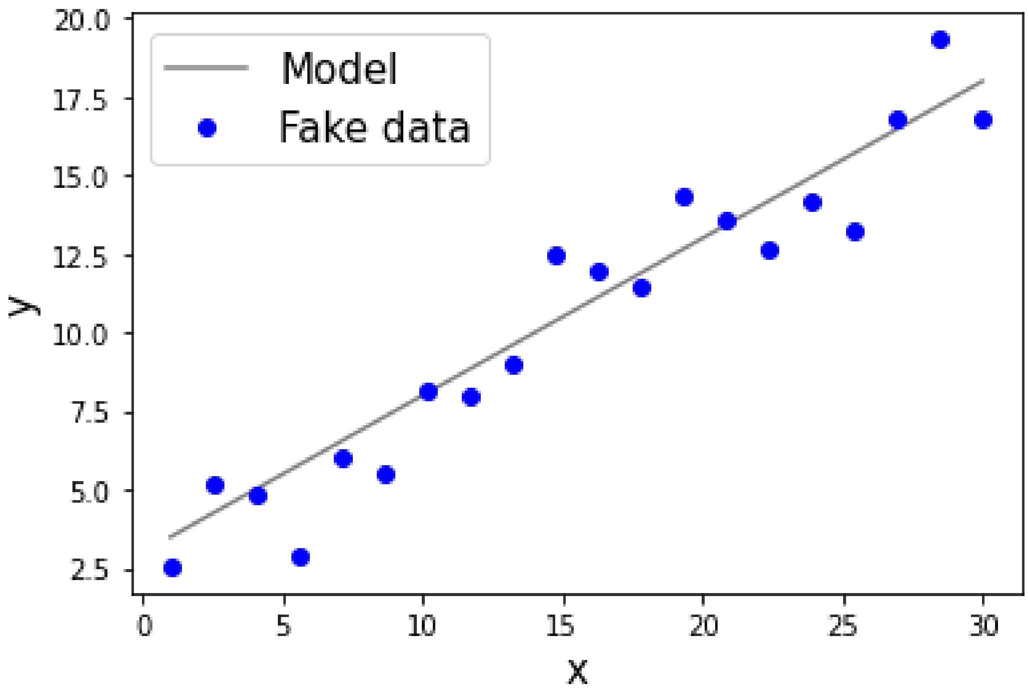
[4]:
```python
#generates noise from a normal distribution
import numpy as np
un = np.random.normal(loc=0, scale= 1, size=20)
print(un)

#generate fake data
y_model = function(xarray,.5,3)
yarray = y_model + un
print(yarray)
```

```
len(yarray)
```

```
[-0.94023846  0.95983567 -0.21373433 -2.88335284 -0.52383502 -1.77587532
  0.0744856  -0.82624481 -0.60024222  2.11581957  0.87744198 -0.40989339
  1.73399939  0.13160878 -1.52740213 -0.73937673 -2.48877108  0.34589992
  2.10143244 -1.1912503 ]
[ 2.55976154  5.22299356  4.81258146  2.90612085  6.02879656  5.53991416
  8.15343297  8.01586046  9.00502094 12.48424062 12.00902093 11.48484346
 14.39189413 13.55266141 12.65680839 14.20799169 13.22175523 16.81958413
 19.33827455 16.8087497 ]
```

[4]: 20

[5]:
```
#making a scatter plot of my data points
import numpy as np
import matplotlib.pyplot as plt
x = xarray
y = y_model
plt.plot(x, y, color = 'grey')
plt.plot(x,yarray,'o', color='blue')
plt.xlabel('x',fontsize=15)
plt.ylabel('y', fontsize=15)
plt.legend(['Model','Fake data'], fontsize = 15)
plt.show()
#source: https://jakevdp.github.io/PythonDataScienceHandbook/04.
↪02-simple-scatter-plots.html
```



[6]:
```
#def chi_square(yarray,y_model):
#    '''
#    Signature: array, array, ~> int
#    calcs the chi square betwen model and data
#    '''
#    import numpy as np
#    new = yarray-y_model
#    new2 = new**2 / sigma**2
#    return new2.sum()
```

[7]:
```
def new(x, sigma_x):
    '''
    signature: int, int ~> int
    gets one new m or b value from gaussian distribution
    '''
    import random as r
    import numpy as np
    nu = np.random.normal(loc=x, scale=sigma_x, size=1)
    return nu
```

1 Guesses:

1.0.1 m = 5, b = 9, sigma_m = 4, sigma_b=3

2 Real:

2.0.1 m = .5, b = 3

[31]:
```
def master(xarray, yarray,y_model, m, b, sigma_ones, sigma_m, sigma_b, un, i):
    '''
    master runs through MCMC i number of times.
    '''
    import random as r
    import numpy as np
    import math

    samplem = np.array([])
    sampleb = np.array([])
    aorrm = np.array([])
    aorrb = np.array([])
    chisqr = np.array([])

    #samplem.append(m)
    #sampleb.append(b)

    count = 0
```

```python
while i > count:


    #make y_model
    y_model = function(xarray,m,b)


    #calc chi-squared of original
    n1 = ((yarray-y_model)**2) / (sigma_ones**2)
    n1 = n1.sum()

    #coin flip - decide whether you vary m or b using np.random.uniform()
    flip = np.random.uniform(low=0.0, high=1.0)

    if flip < 0.5:
        new_m_value = new(m, sigma_m)

        #make new model data
        y_model2 = function(xarray,new_m_value,b)

        #calc chi-squared of y_model2
        nm = ((yarray-y_model2)**2) / (sigma_ones**2)
        nm = nm.sum()

        #accept or reject
        if nm < n1:
            aorrm = np.append(aorrm,1)
            samplem = np.append(samplem,new_m_value)
            sampleb = np.append(sampleb, b)
            m = new_m_value
            chisqr = np.append(chisqr, nm)
            count = count + 1
        else:
            if math.e**(-(nm-n1)/2) >= np.random.uniform():
                aorrm = np.append(aorrm,1)
                samplem = np.append(samplem, new_m_value)
                sampleb = np.append(sampleb, b)
                m = new_m_value
                chisqr = np.append(chisqr, nm)
                count = count + 1
            else:
                aorrm = np.append(aorrm, 0)
                sampleb = np.append(sampleb, b)
                samplem = np.append(samplem, m)
                chisqr = np.append(chisqr, n1)
                count = count + 1
```

```python
        else:
            new_b_value = new(b, sigma_b)

            #make new model data
            y_model2 = function(xarray,m,new_b_value)

            #calc chi-squared of y_model2
            nb = ((yarray-y_model2)**2) / (sigma_ones**2)
            nb = nb.sum()

            #accept or reject
            if nb < n1:
                aorrb = np.append(aorrb, 1)
                sampleb = np.append(sampleb, new_b_value)
                samplem = np.append(samplem, m)
                b = new_b_value
                chisqr = np.append(chisqr, nb)
                count= count + 1
            else:
                if math.e**(-(nb-n1)/2) >= np.random.uniform():
                    aorrb = np.append(aorrb, 1)
                    sampleb = np.append(sampleb, new_b_value)
                    samplem = np.append(samplem, m)
                    b = new_b_value
                    chisqr = np.append(chisqr, nb)
                    count= count + 1
                else:
                    aorrb = np.append(aorrb, 0)
                    sampleb = np.append(sampleb, b)
                    samplem = np.append(samplem, m)
                    chisqr = np.append(chisqr, n1)
                    count = count + 1

            #acceptance fraction
            #totm = 0
            #for item in aorrm:
            #    if item == 1:
            #        totm = totm + 1
            #accfm = totm / i
            accfm = np.sum(aorrm)/np.shape(aorrm)

            #totb = 0
            #for item in aorrb:
            #    if item == 1:
            #        totb = totb + 1
            #accfb = totb / i
            accfb=np.sum(aorrb)/np.shape(aorrb)
```

```
    #print( 'samplem: ' + str(samplem) + '\n acceptance fraction for m: ' +␣
↪str(accfm) + '\n sampleb: ' + str(sampleb) + '\n Acceptance fraction for b:␣
↪' + str(accfb) + '\n Chi-square: ' + str(chisqr))
    print('accfm: ' +str(accfm)+ '\naccfb: ' +str(accfb))
    print(np.shape(aorrb),np.shape(aorrm))
    return chisqr, samplem, sampleb
```

2.0.2 Syntax

master(xarray, yarray,y_model, m, b, sigma_ones, sigma_m, sigma_b, un, i):

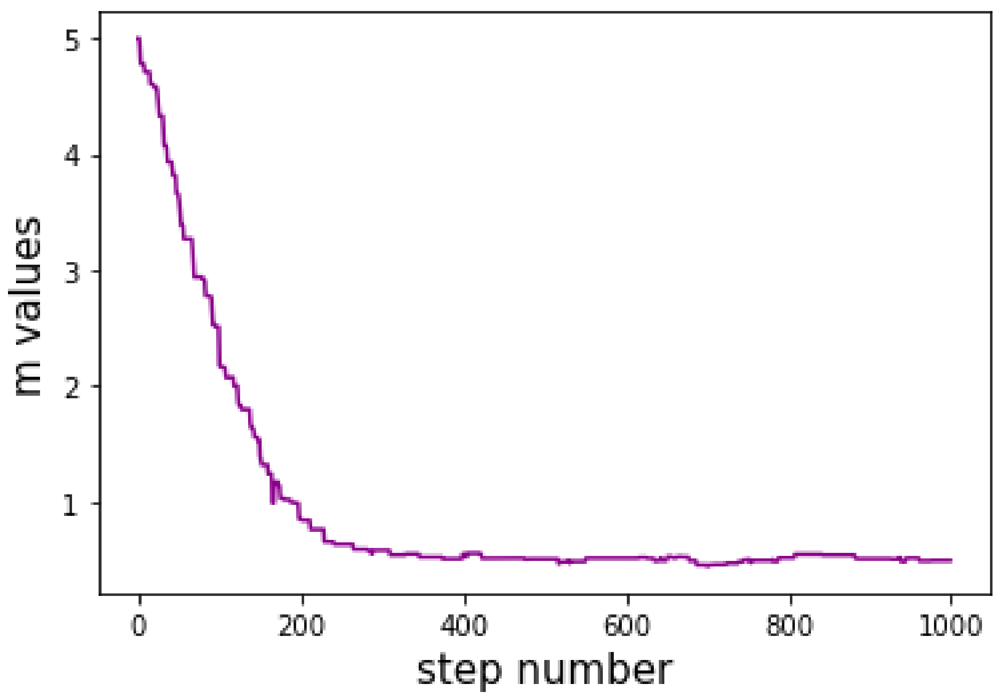
[32]:
```
chisqr, samplem, sampleb = master(xarray,yarray,y_model, 5, 9, sigma_ones,0.1 ,␣
↪.9, un, 1000)
```

```
accfm: [0.24319066]
accfb: [0.33539095]
(486,) (514,)
```

3 Plots

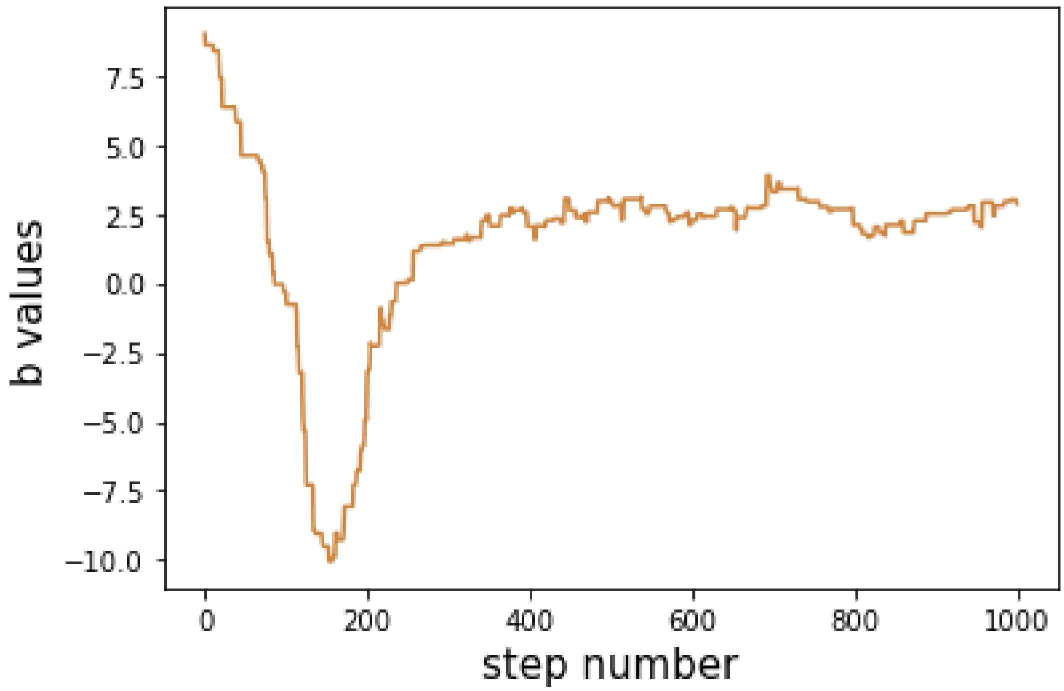
[33]:
```
import matplotlib.pyplot as plt
import numpy as np
numline = np.arange(np.shape(chisqr)[0])
#m vs step number
plt.plot(numline, samplem, color='darkmagenta')
plt.ylabel('m values',fontsize=15)
plt.xlabel(r'step number', fontsize=15)
```

[33]: Text(0.5, 0, 'step number')

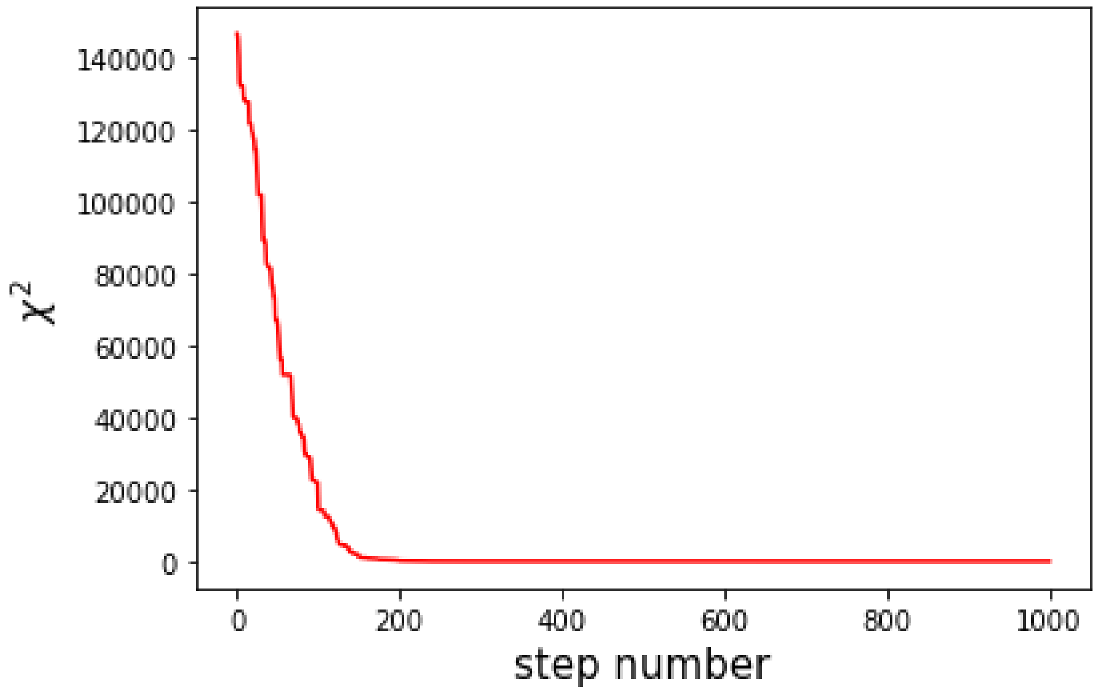


[34]:
```
import matplotlib.pyplot as plt

#b vs step number
plt.plot(numline, sampleb, color='peru')
plt.ylabel('b values',fontsize=15)
plt.xlabel(r'step number', fontsize=15)
```

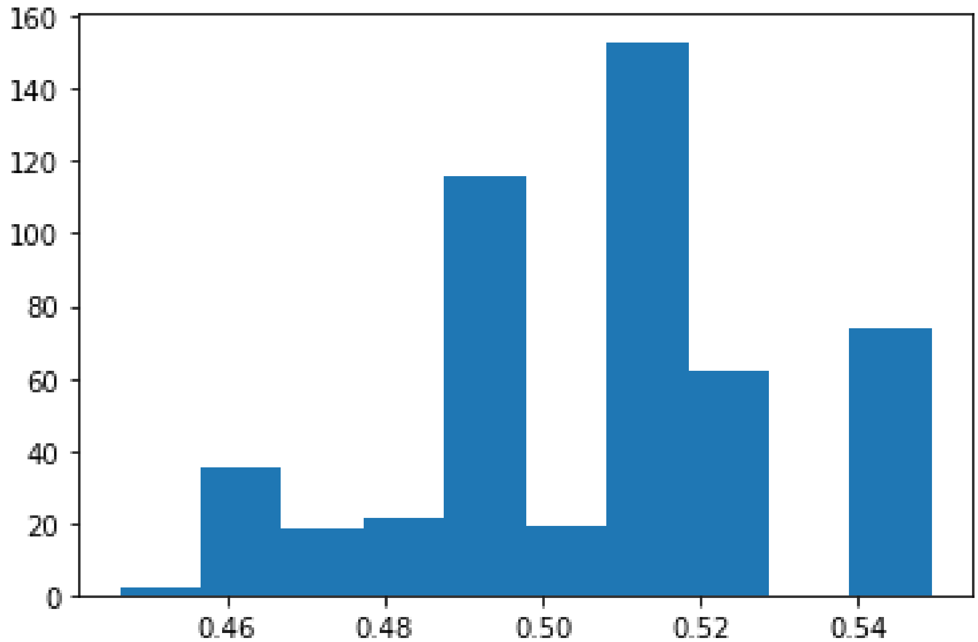
[34]: Text(0.5, 0, 'step number')



[35]:
```
import matplotlib.pyplot as plt

#chi-square vs step number
plt.plot(numline, chisqr, color='red')
plt.xlabel(r'step number',fontsize=15)
plt.ylabel(r'$\chi^2$', fontsize=15)
plt.show()
```

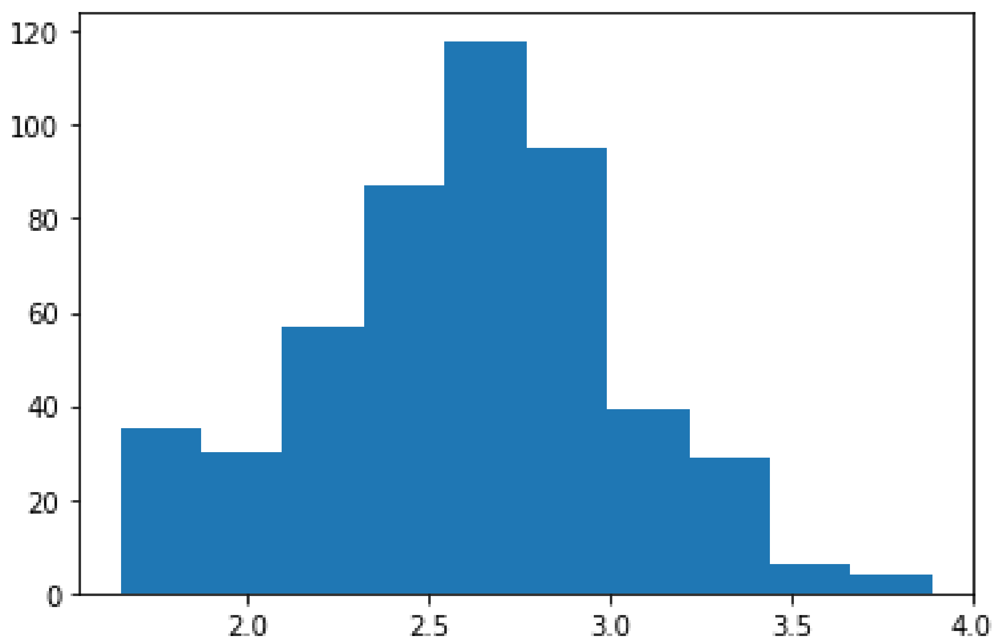


[36]:
```
import matplotlib.pyplot as plt
plt.hist(samplem[500:], bins = 10)
plt.show()
```

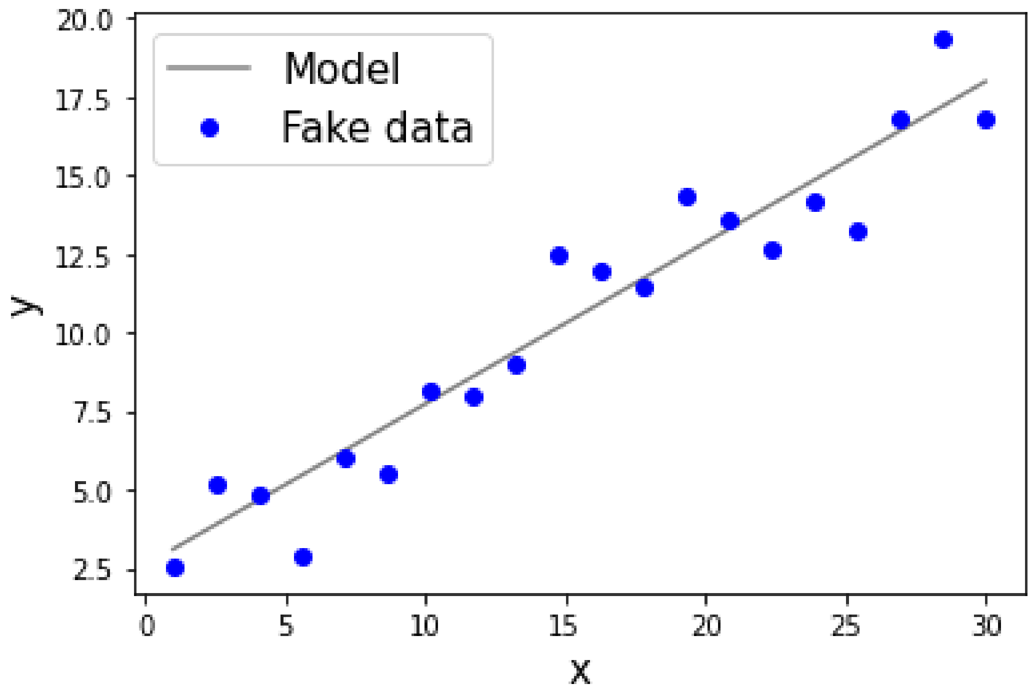


[37]:
```
import matplotlib.pyplot as plt
plt.hist(sampleb[500:], bins=10)
```

[37]:
```
(array([ 35.,  30.,  57.,  87., 118.,  95.,  39.,  29.,   6.,   4.]),
 array([1.64723916, 1.87168023, 2.09612131, 2.32056238, 2.54500346,
        2.76944453, 2.9938856 , 3.21832668, 3.44276775, 3.66720882,
        3.8916499 ]),
 <BarContainer object of 10 artists>)
```



[38]:
```
import numpy as np
import matplotlib.pyplot as plt
y_model

est = .513 * xarray + 2.6
plt.plot(xarray, est, color = 'grey')
plt.plot(xarray,yarray,'o', color='blue')
plt.xlabel('x',fontsize=15)
plt.ylabel('y', fontsize=15)
plt.legend(['Model','Fake data'], fontsize = 15)
plt.show()
```


Model
Fake data
y
x
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
5
10
15
20
25
30

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```