```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(123)

def refPeriod(): # function used after spike to see what happens in next 11 ms
    k = np.zeros(5)
    for i in np.arange(6):
        if np.random.binomial(1, 0.055*i/6) == 1:
            k = np.append(k,1)
            return k
        k = np.append(k,0)
    return k

spk_trn = np.zeros(1) # w/ ref period
while len(spk_trn) < 9000: # go through all ms
    if spk_trn[-1] == 1: # if spike occured during relative ref period
        spk_trn = np.append(spk_trn, refPeriod()) # go into next ref period
    elif np.random.binomial(1, 0.055) == 1: # else, see if spike occured now
        spk_trn = np.append(spk_trn, 1)
        spk_trn = np.append(spk_trn,refPeriod())
    else: # or didn't occur now
        spk_trn = np.append(spk_trn, 0)
spk_trn = spk_trn[1:9001]

spk_trn2 = np.zeros(9000) # w/o ref period
for i in range(9000):
    if np.random.binomial(1, 0.055) == 1:
        spk_trn2[i] = 1

ISIs = np.diff(np.flatnonzero(spk_trn))
ISIs2 = np.diff(np.flatnonzero(spk_trn2))

plt.figure()
plt.suptitle('Spike Trains')
plt.subplot(2,1,1)
plt.plot(range(9000),spk_trn, lw = 0.2)
ax = plt.gca()
ax.axes.yaxis.set_ticks([])
plt.subplot(2,1,2)
plt.plot(range(9000), spk_trn2, lw = 0.2, color = 'gray')
plt.xlabel('time (ms)')
ax = plt.gca()
ax.axes.yaxis.set_ticks([])
plt.show()

plt.figure()
plt.suptitle('TIH plots')
plt.subplot(2,1,1)
plt.hist(ISIs, bins = np.arange(0,150), density = True)
plt.xlim((0,150))
plt.subplot(2,1,2)
plt.hist(ISIs2, bins = np.arange(0,150), density = True, color='gray')
plt.xlim((0,150))
plt.xlabel('ISI (ms)')
plt.show()

plt.figure()
plt.suptitle('TIH plots - Log Scale')
plt.subplot(2,1,1)
plt.hist(ISIs, bins = np.arange(0,150), density = True, log = True)
plt.xlim((0,150))
plt.subplot(2,1,2)
plt.hist(ISIs2, bins = np.arange(0,150), density = True, log = True, color='gray')
```

```
plt.xlim((0,150))
plt.xlabel('ISI (ms)')
plt.show()

z = plt.hist(ISIs, bins = np.arange(0,150), density = True)[0] # make pdf
z2 = plt.hist(ISIs2, bins = np.arange(0,150), density = True)[0] # make pdf
s = 1 - np.cumsum(z) # make survivor function
s2 = 1 - np.cumsum(z2) # make survivor function

plt.figure()
plt.suptitle('Survival Functions')
plt.subplot(2,1,1)
plt.plot(s)
plt.xlim((0,150))
plt.subplot(2,1,2)
plt.plot(s2, color = 'gray')
plt.xlim((0,150))
plt.xlabel('ISI (ms)')
plt.show()

h = z/s # make hazard function
h[max(ISIs)] = 0
h2 = z2/s2 # make hazard function
h2[max(ISIs2)] = 0

plt.figure()
plt.suptitle('Hazard Functions')
plt.subplot(2,1,1)
plt.plot(h)
plt.xlim((0,60))
plt.ylim((0,0.25))
plt.axhline(np.mean(spk_trn), color = 'black', lw = 0.2)
plt.subplot(2,1,2)
plt.plot(h2, color = 'gray')
plt.xlim((0,60))
plt.ylim((0,0.25))
plt.axhline(np.mean(spk_trn2), color = 'black', lw = 0.2)
plt.xlabel('ISI (ms)')
plt.show()

acs = np.zeros(201) # make auto-correlations
for i in range(1,101):
    acs[100+i] = np.sum(np.logical_and(spk_trn[:-i], spk_trn[i:]))/np.sum(spk_trn[:-i])
acs[100] = 0
acs[:100] = np.flip(acs[101:])
acs = acs/0.001
acs2 = np.zeros(201)
for i in range(1,101):
    acs2[100+i] = np.sum(np.logical_and(spk_trn2[:-i], spk_trn2[i:]))/np.sum(spk_trn2[:-i])
acs2[100] = 0
acs2[:100] = np.flip(acs2[101:])
acs2 = acs2/0.001

plt.figure()
plt.suptitle('Autocorrelation Plots')
plt.subplot(2,1,1)
plt.plot(np.arange(start=-100, stop=101), acs)
plt.axhline(np.mean(spk_trn)*1000, color = 'black', lw = 0.2)
plt.subplot(2,1,2)
plt.plot(np.arange(start=-100, stop=101), acs2, color = 'gray')
plt.axhline(np.mean(spk_trn2)*1000, color = 'black', lw = 0.2)
plt.xlabel('Lag (ms)')
plt.show()
```