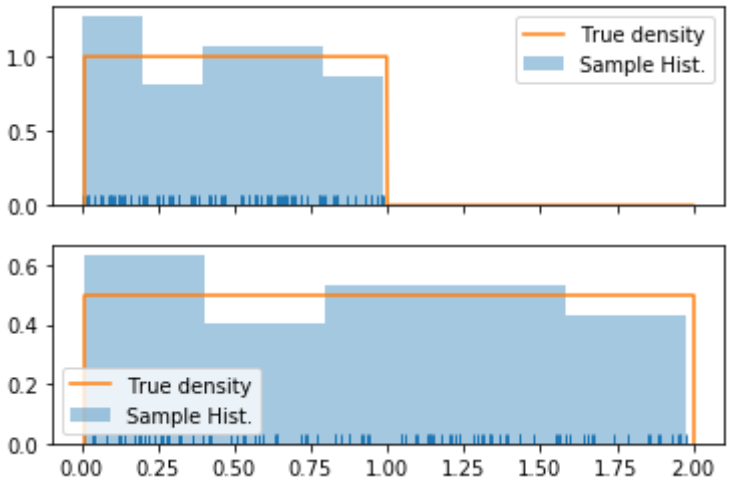
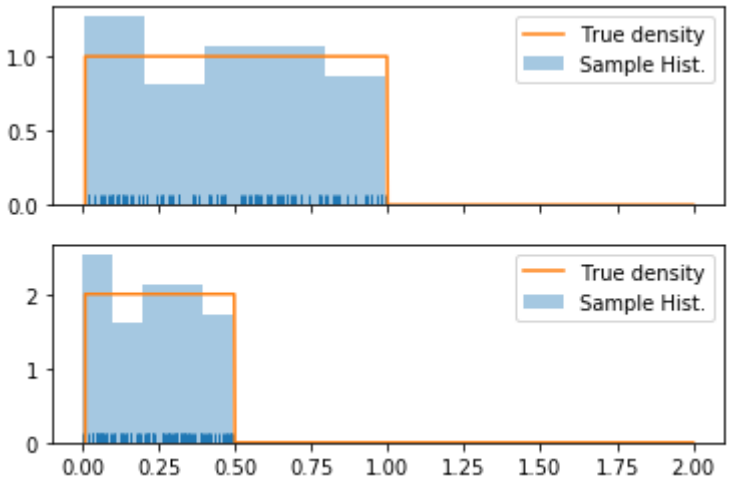
In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def show_samples_and_density(samples, density, ax=None):
    if ax is None:
        ax = plt.gca()
    sns.distplot(samples, kde=False, rug=True, hist=True, norm_hist=True
, ax=ax, label='Sample Hist.')
    xq = np.linspace(0.01, 2, num=1000)
    density_xq = density(xq)
    ax.plot(np.concatenate(([xq[0]],xq,[xq[-1]])), np.concatenate(([0],d
ensity(xq),[0])), label='True density')
    ax.legend()
```

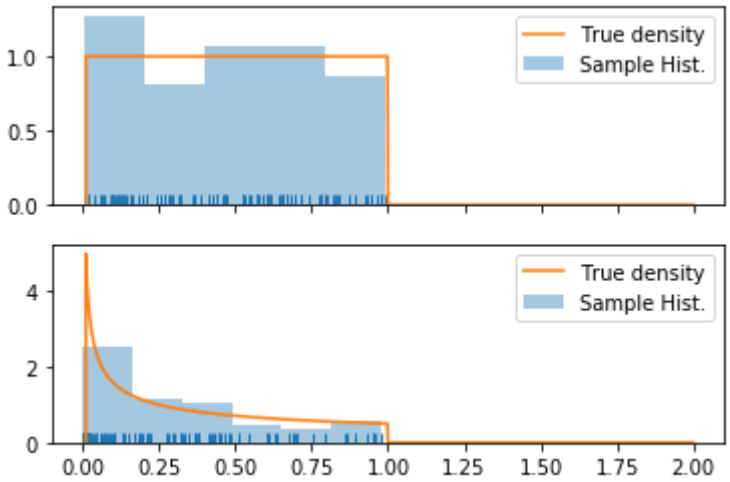
In [2]:
```
rng = np.random.RandomState(0)
n_samples = 100
Z = rng.rand(n_samples)
def G(Z):
    return 2*Z
def Ginv(X):
    return X/2
X = G(Z)

def pz(z):
    return np.ones_like(z) * np.logical_and(z >= 0, z <= 1)
def px(x):
    #return pz(x)
    #return 2*pz(x) #G(pz(x))
    #return 1/2*pz(x) #Ginv(pz(x))
    #return pz(2*x) #pz(G(x))
    #return pz(x/2) #pz(Ginv(x))
    return 1/2*pz(x/2) #|dGinv(x)/dx| pz(Ginv(x))

fig, axes = plt.subplots(2,1, figsize=(6, 4), sharex=True)
for samples, density, ax in zip([Z, X], [pz, px], axes.ravel()):
    show_samples_and_density(samples, density, ax)
```



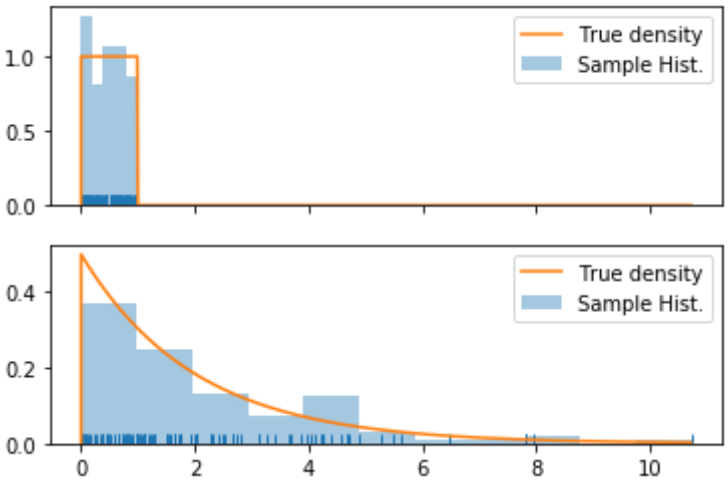
In [3]:
```
rng = np.random.RandomState(0)
n_samples = 100
Z = rng.rand(n_samples)
def G(Z):
    return Z/2
def Ginv(X):
    return 2*X
X = G(Z)

def pz(z):
    return np.ones_like(z) * np.logical_and(z >= 0, z <= 1)
def px(x):
    return 2*pz(2*x) #|dGinv(x)/dx| pz(Ginv(x))

fig, axes = plt.subplots(2,1, figsize=(6, 4), sharex=True)
for samples, density, ax in zip([Z, X], [pz, px], axes.ravel()):
    show_samples_and_density(samples, density, ax)
```



In [4]:
```
rng = np.random.RandomState(0)
n_samples = 100
Z = rng.rand(n_samples)
def G(Z):
    return Z**2
def Ginv(X):
    return np.sqrt(X)
X = G(Z)

def pz(z):
    return np.ones_like(z) * np.logical_and(z >= 0, z <= 1)
def px(x):
    return 1/2*x**(-1/2)*pz(Ginv(x)) #|dGinv(x)/dx| pz(Ginv(x))

fig, axes = plt.subplots(2,1, figsize=(6, 4), sharex=True)
for samples, density, ax in zip([Z, X], [pz, px], axes.ravel()):
    show_samples_and_density(samples, density, ax)
```



In [5]:
```
import torch
def torch_show_samples_and_density(samples, density, xlim=[0.01, 2], ax=
None):
    if ax is None:
        ax = plt.gca()
    xq = torch.linspace(*xlim, 1000)
    density_xq = density(xq)
    # Convert to numpy to display
    sns.distplot(samples.detach().numpy(), kde=False, rug=True, hist=Tru
e, norm_hist=True, ax=ax, label='Sample Hist.')
    density_xq = density_xq.detach().numpy()
    xq = xq.detach().numpy()
    ax.plot(np.concatenate(([xq[0]],xq,[xq[-1]])), np.concatenate(([0],d
ensity_xq,[0])), label='True density')
    ax.legend()

rng = np.random.RandomState(0)
n_samples = 100
Z = rng.rand(n_samples)
Z = torch.from_numpy(Z)

def G(Z):
    #return Z**2
    return -torch.log(Z**2)
def Ginv(X):
    #return torch.sqrt(X)
    return torch.sqrt(torch.exp(-X))
X = G(Z)

def pz(z):
    return torch.ones_like(z) * (z >= 0).float() * (z <= 1).float()
def px(x):
    x.requires_grad_(True)
    z = Ginv(x)
    sum_z = torch.sum(z)
    sum_z.backward()
    with torch.no_grad():
        #return x.grad * pz(Ginv(x))  #|dGinv(x)/dx| pz(Ginv(x))
        return torch.abs(x.grad) * pz(Ginv(x))  #|dGinv(x)/dx| pz(Ginv
(x))

fig, axes = plt.subplots(2,1, figsize=(6, 4), sharex=True)
for samples, density, ax in zip([Z, X], [pz, px], axes.ravel()):
    torch_show_samples_and_density(samples, density, xlim=[0.01, torch.m
ax(X)], ax=ax)
```


True density
Sample Hist.
1.0
0.5
0.0
True density
Sample Hist.
0.4
0.2
0.0
0
2
4
6
8
10