Hw3 Chen Feng Tsai

1\.

(a) $Y_t \sim \text{Bernoulli}(\mu(t))$

(b) $\mu(t) = 1 - e^{(-\rho_t V)}$

(c) $1-\mu(t) = e^{-\rho_t V}$

$$-\log(1-\mu(t)) = \rho_t V$$

$$\log(-\log(1-\mu(t))) = \log \rho_t + \log V$$

$$= \log \rho_0 + \log V - t \cdot \log 2$$

$\Rightarrow \beta_0 = \log \rho_0 + \log V$, $\beta_1 = -\log 2$

(d) $\beta_0 = \log \rho_0 + \log v$

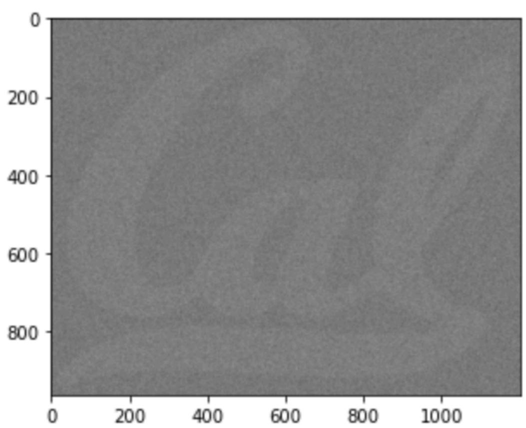
$$\log \rho_0 = \beta_0 - \log v$$

$$\rho_0 = e^{(\beta_0 - \log v)}$$

2. (a)

```
import matplotlib.pyplot as plt
import pandas as pd
X = pd.read_pickle("X.pkl")
plt.imshow(X, cmap = plt.cm.get_cmap('gray'))
```

```
<matplotlib.image.AxesImage at 0x7fb51d32d1f0>
```



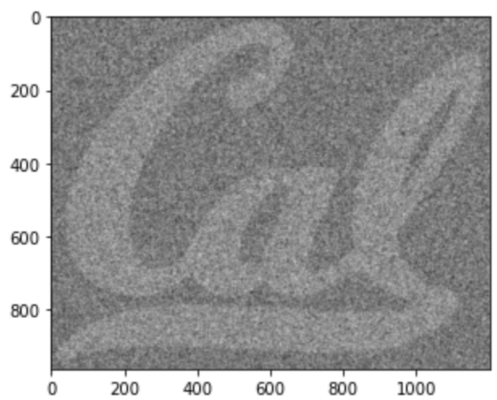
(b) $$p\left(z_{1,3} \mid z_{1,1} = z_{1,1}^{(t)},\ z_{1,2} = z_{1,2}^{(t)},\ z_{1,4} = z_{(1,4)}^{(t-1)},\ \ldots,\ z_{n,m} = z_{n,m}^{(t-1)},\ X\right)$$

(c)

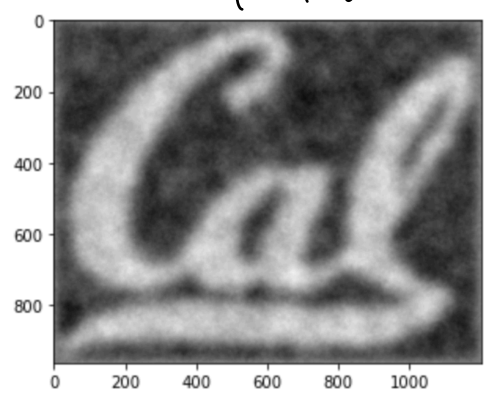
```
Z = X
Z_t = Z
for t in range(T):
    Z = Z_t
    for i in range(X.shape[0]):
        for j in range(X.shape[1]):
            N = [(i-1,j), (i,j-1),(i+1, j),(i, j+1)]
            S = sum(Z[u,v] for (u,v) in N if 0 <= u < X.shape[0] and 0 <= v < X.shape[1])
            Z[i,j] = ((tau*X[i,j] + b*S)/(a+tau)+np.random.randn()*np.sqrt(1/(a+tau)))
    Z_t = Z
```

(d)

T = 1



T = 100




```
a = 250
b = 62.5
tau = 0.01
T = 1

Z = X
Z_t = Z
for t in range(T):
    Z = Z_t
    for i in range(X.shape[0]):
        for j in range(X.shape[1]):
            N = [(i-1,j), (i,j-1),(i+1, j),(i, j+1)]
            S = sum(Z[u,v] for (u,v) in N if 0 <= u < X.shape[0] and 0 <= v < X.shape[1])
            Z[i,j] = ((tau*X[i,j] + b*S)/(a+tau)+np.random.randn()*np.sqrt(1/(a+tau)))
    Z_t = Z
plt.imshow(Z_t, cmap = plt.cm.get_cmap('gray'))
```

It takes about 11 1/2 minutes for $T=100$.

(e) $p(z|x)$ in (c) is the same as

$$\propto \exp\Big(-\frac{1}{2}\sum_{(i,j)\in I_{even}}\Big[(a+\tau)z_{ij}^2 - 2\tau z_{ij}x_{ij} - b\sum_{(i',j')\in N(i,j)} z_{ij}z_{i'j'}\Big]$$

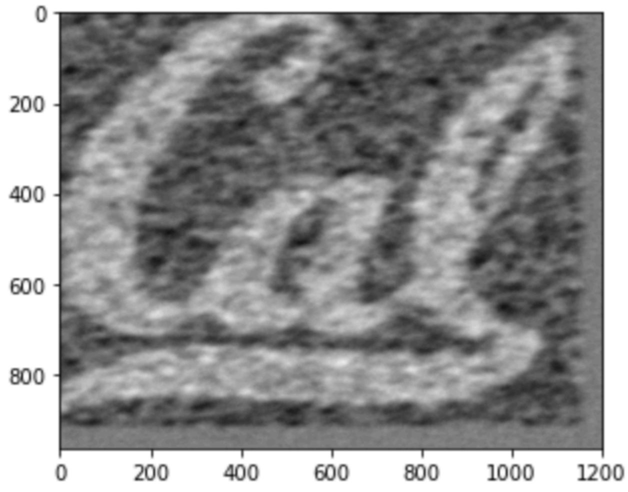
$$\times \exp\Big(-\frac{1}{2}\sum_{(i,j)\in I_{odd}}\Big[(a+\tau)z_{ij}^2 - 2\tau z_{ij}x_{ij} - b\sum_{(i',j')\in N(i,j)} z_{ij}z_{i'j'}\Big]$$

and the neighbors of a specific pixel, which is $N(i,j)$, are even when the pixel's $i+j$ is odd, vice versa. Therefore, they will not affect each other when updating at once.

(f) It takes about 12 seconds to run the cell

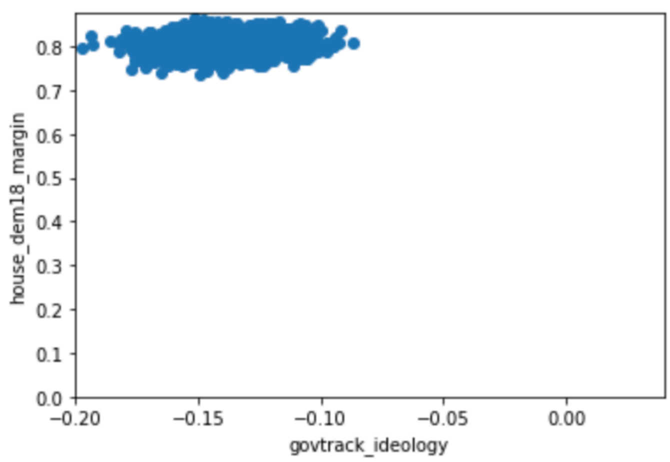
```
%%time
a = 250
b = 62.5
tau = 0.01
T = 100
I, J = np.meshgrid(np.arange(X.shape[1]), np.arange(X.shape[0]))
even = (I+J)%2 == 0
odd = (I+J)%2 == 1

Z = np.pad(X,1)
e = np.pad(even, 1)
o = np.pad(odd, 1)

for t in range(T):
    S = Z[2:,1:-1] + Z[1:-1, 2:] + Z[1:-1, 2:] + Z[1:-1, :-2]
    mean = (tau*X + b*S)/(a+tau)
    delta = mean + np.random.randn(X.shape[0], X.shape[1])*np.sqrt(1/a+tau)
    Z[e] = np.pad(delta*even, 1)[e]

    S = Z[2:,1:-1] + Z[1:-1, 2:] + Z[1:-1, 2:] + Z[1:-1, :-2]
    mean = (tau*X + b*S)/(a+tau)
    delta = mean + np.random.randn(X.shape[0], X.shape[1])*np.sqrt(1/a+tau)
    Z[o] = np.pad(delta*odd, 1)[o]

plt.imshow(Z, cmap = plt.cm.get_cmap('gray'))
```

```
CPU times: user 10.9 s, sys: 489 ms, total: 11.4 s
Wall time: 11.5 s
```



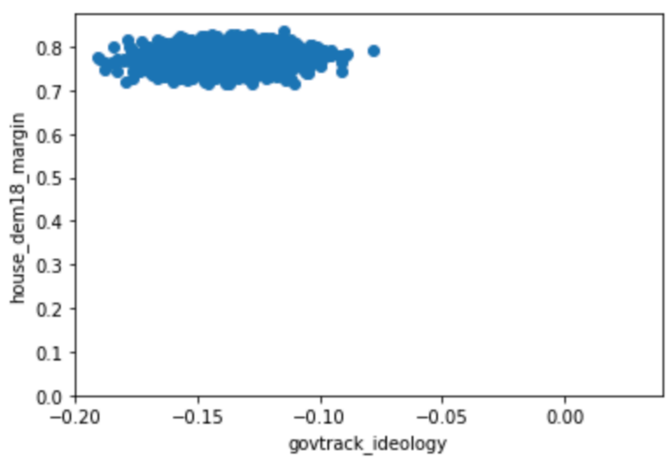
3. (a) $\sigma_0^2 = 1$

```
with pm.Model() as mdl_ols_glm:
    my_priors = {"govtrack_ideology": pm.Normal.dist(mu=0, sigma=1),
                "house_dem18_margin": pm.Normal.dist(mu=0, sigma=1)}
    pm.glm.GLM.from_formula("house_dem20_margin ~ 0 + govtrack_ideology + house_dem18_margin",
                            data, family=pm.glm.families.Normal(), priors = my_priors)

    traces_ols_glm = pm.sample(1000)
```



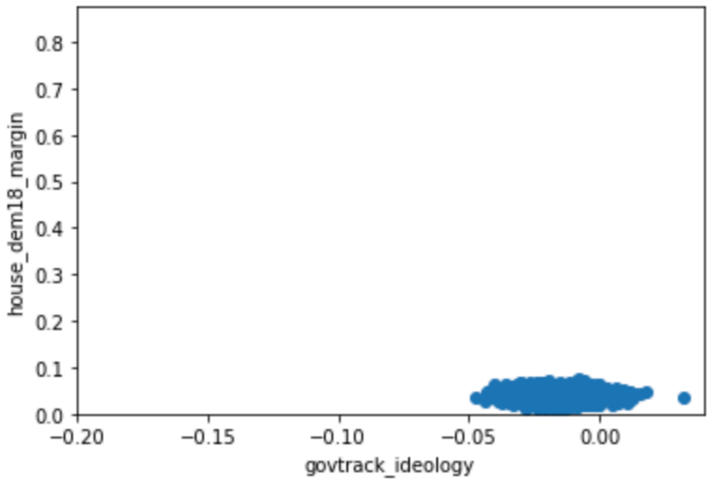
$\sigma_0^2 = 0.01$

```
with pm.Model() as mdl_ols_glm:
    my_priors = {"govtrack_ideology": pm.Normal.dist(mu=0, sigma=0.1),
                "house_dem18_margin": pm.Normal.dist(mu=0, sigma=0.1)}
    pm.glm.GLM.from_formula("house_dem20_margin ~ 0 + govtrack_ideology + house_dem18_margin",
                            data, family=pm.glm.families.Normal(), priors = my_priors)

    traces_ols_glm1 = pm.sample(1000)
```



$\sigma_0^2 = 10^{-4}$

```python
with pm.Model() as mdl_ols_glm:
    my_priors = {"govtrack_ideology": pm.Normal.dist(mu=0, sigma=0.01),
                 "house_dem18_margin": pm.Normal.dist(mu=0, sigma=0.01)}
    pm.glm.GLM.from_formula("house_dem20_margin ~ 0 + govtrack_ideology + house_dem18_margin",
                            data, family=pm.glm.families.Normal(), priors = my_priors)

    traces_ols_glm2 = pm.sample(1000)
```



(b) It produces different results since for $\sigma_0^2 = 10^{-4}$, it means that the data will be squeezed near (0, 0) which means the predictive coefficient has less space for variation, since the prior distribution is already pretty concentrated. And the largest σ_0 will have the most scattered plot.

(c) If σ_0 is small, we are assuming that ideology almost has only half the effect on the 2020 outcome prediction than the 2018 outcome does, but in opposite direction. If σ_0 is larger, the effect of ideology are $\frac{1}{4}$ compared to 2018 outcome. The differences between the two affects with small σ_0 are smaller than large σ_0.