Homework 2

Due 2/6 12pm

1.

$$P(r|\lambda) = \exp(-\lambda)\frac{\lambda^r}{r!}$$

(a) Taking the log of the likelihood function:

$$L(\lambda) = \ln P(r|\lambda) = -\lambda + r\ln\lambda - \ln r!$$

finding the maximum by taking the derivative with respect to λ and setting it to zero:

$$\frac{dL}{d\lambda} = -1 + \frac{r}{\lambda} = 0 \implies \hat{\lambda} = r$$

so the maximum likelihood estimate for λ is $\hat{\lambda} = r$.
(b) Given the derivative with respect to the function $\ln\lambda$:

$$\frac{\mathrm{d}}{\mathrm{d}(\ln\lambda)}u^n = nu^n, \qquad \frac{\mathrm{d}}{\mathrm{d}(\ln\lambda)}\ln\lambda = 1$$

we can find the curvature of the log likelihood function:

$$\frac{\mathrm{d}}{\mathrm{d}(\ln\lambda)}L(\lambda) = -\lambda + r = 0 \implies \hat{\lambda} = r$$
$$\frac{\mathrm{d}^2}{\mathrm{d}(\ln\lambda)^2}L(\lambda) = -\lambda = k$$

For a normal distribution with width σ, the curvature is $k = -1/\sigma^2$. So the width is approximately

$$\sigma \propto \frac{1}{\sqrt{-k}} = \frac{1}{\sqrt{\lambda}}$$

and the 95% confidence interval at the MLE is approximately

$$\hat{\lambda} \pm 2\sigma = r \pm \frac{2}{\sqrt{\hat{\lambda}}}$$

(c) Given the new Poisson distribution

$$P(r|\lambda) = \exp(-(\lambda+b))\frac{(\lambda+b)^r}{r!}$$

the log likelihood function is

$$L(\lambda) = -(\lambda+b) + r\ln(\lambda+b) - \ln r!$$

and the maximum likelihood estimate for λ is

$$\frac{dL}{d\lambda} = -1 + \frac{r}{\lambda+b} = 0 \implies \hat{\lambda} = r - b$$

the value $\hat{\lambda} = 9 - 13 = -4$ is not physically meaningful, so the MLE will be the lowest possible value for λ which is $\hat{\lambda} = 0$. From this we can infer that the remote star is very dim. The Bayesian posterior distribution for λ is

$$P(\lambda|r) = \frac{P(r|\lambda)P(\lambda)}{P(r)}$$

and sketched in the figure below

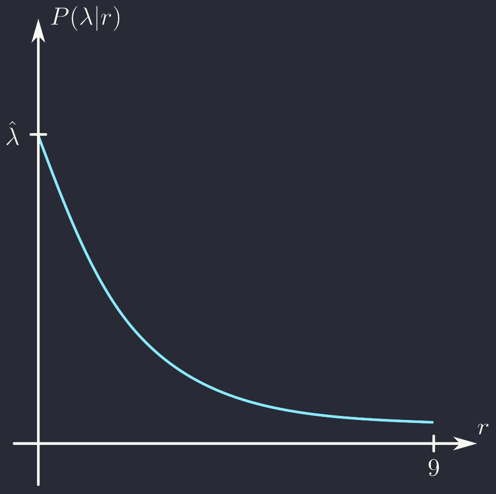

Figure 1.1: The posterior distribution for λ given r

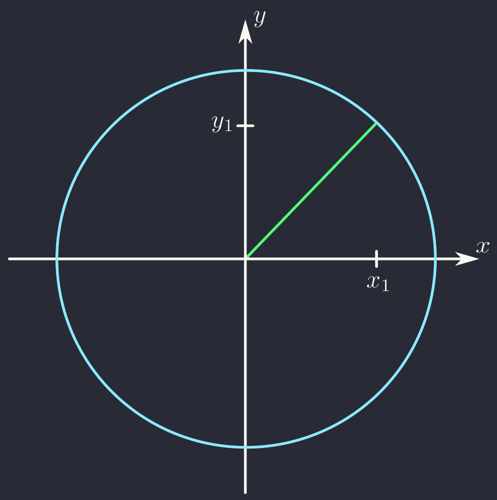

Figure 1.2: Segment of Gaussian distribution at $\{x_1, y_1\}$

2. (a) From the geometric picture as shown in Figure 1.2, the segment of the Gaussian is a circle with radius $\rho = \sqrt{x_1^2 + y_1^2}$ and the circumference is $2\pi\rho$ which directly relates to the extra factor of ρ and canceling the 2π in the denominator. This is also related to the Jacobian when transforming from Cartesian to polar coordinates when computing the integral. Using the integral of a 1D Gaussian:

$$\int_{-\infty}^{\infty} \exp(-ax^2)\,\mathrm{d}x = \sqrt{\frac{\pi}{a}}$$

Verifying that the integral is normalized:

$$\begin{aligned}
\frac{1}{2\pi\sigma^2} \iint_{-\infty}^{\infty} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \mathrm{d}x\,\mathrm{d}y &= \frac{1}{\sigma^2}\int_0^{\infty} \exp\left(-\frac{\rho^2}{2\sigma^2}\right)\rho\,\mathrm{d}\rho \\
\frac{1}{2\pi\sigma^2} \int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{2\sigma^2}\right) \mathrm{d}x \int_{-\infty}^{\infty} \exp\left(-\frac{y^2}{2\sigma^2}\right) \mathrm{d}y &= \frac{1}{\sigma^2}\int_0^{\infty} \exp\left(-\frac{\rho^2}{2\sigma^2}\right)\rho\,\mathrm{d}\rho \\
\frac{1}{2\pi\sigma^2}\sqrt{2\pi\sigma^2}\sqrt{2\pi\sigma^2} &= \frac{1}{\sigma^2}\int_0^{\infty} \sigma^2 \exp(-u)\,\mathrm{d}u \\
\frac{2\pi\sigma^2}{2\pi\sigma^2} &= \left[-e^{-u}\right]\Big|_0^{\infty} = 1
\end{aligned}$$

so both integrals are normalized.

(b)

$$P(\rho) = \frac{\rho}{\sigma_w^k} \exp\left(-\frac{\rho^2}{2\sigma_w^2}\right)$$

The sketch for the distribution of $P(\rho)$ is shown in Figure 1.3.

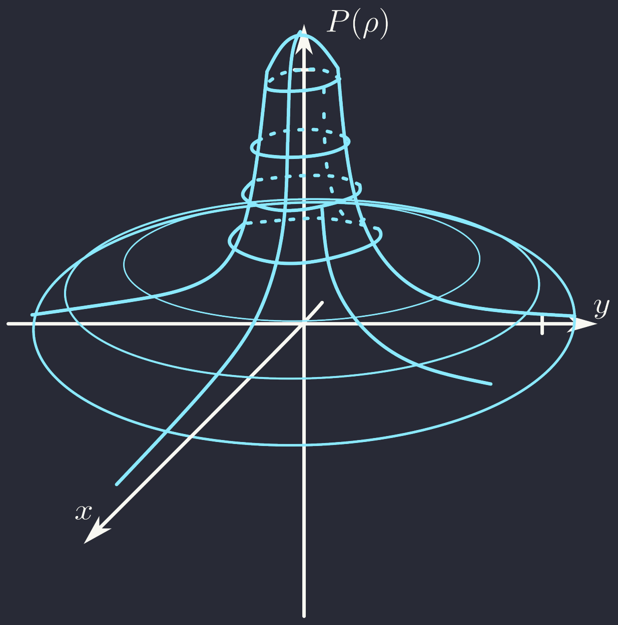

Figure 1.3: The distribution of ρ for $k = 1000$

(c) The standard deviation is

$$\sigma_w = \sqrt{\frac{\sum(w-\mu)^2}{k}}$$

and because the distribution is centered at the origin, the mean is $\mu = 0$, so

$$\sigma_w = \sqrt{\frac{\rho^2}{k}}$$

Since most of the probability mass lies around ρ or equivalently a radius of the thin shell where $\rho = r = \sigma_w\sqrt{k}$, and the thickness of the shell is equivalent to the standard deviation of the gaussian $= r/\sqrt{k}$.
(d) Taking the ratio of the probability density from the origin to a point $\rho = \omega_w\sqrt{k}$ away:

$$\frac{P(0)}{P(\sigma_w\sqrt{k})} = \exp\left\{\frac{\sigma_w^2 k}{2\sigma_w^2}\right\} = \exp\left\{\frac{k}{2}\right\}$$

(e) For a shell to contain 95% of the probability mass, $\sigma_w = 2$ and thus the radius and thickness of the shell are

$$r = \sigma_w\sqrt{k} = 2\sqrt{1000} = 63.25, \qquad \frac{r}{\sqrt{1000}} = 2$$

and the probability density is $\exp\{500\} \approx 10^{217}$ larger at the origin than at the edge of the shell.
(f) For a 1% difference in σ_w at the origin, the radius term is at zero so the exponents are 1, so the ratio of the probability densities is

$$\frac{(1.01\sigma_w)^k}{\sigma_w^k} = 1.01^k \approx 20959.$$

(g) Because of the large amount of parameters, the MLE would have an expected value at where the probability density is the largest, which is at the origin, but as we have seen, the probability mass is almost all at the edge of the thin shell of radius $\sigma_w\sqrt{k}$. The narrow peak at the origin will overwhelm the MLE and thus we don't see the full picture of the distribution.

3. (a) From class

$$\mathcal{R} = \frac{\frac{F_a!F_b!}{(F+1)!}}{1/2^F} = \frac{2^F F_a!F_b!}{(F+1)!}$$

(b) Taking the log of the ratio and using Stirling's approximation as shown in the code we get the plot of the three simulations in Figure 1.4

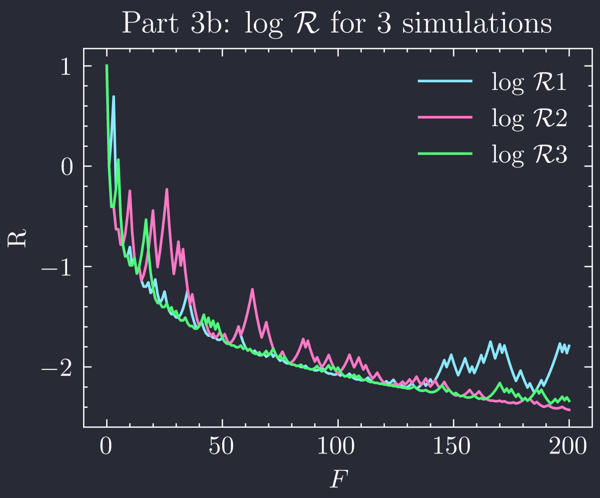

Figure 1.4: The log of the ratio of the likelihoods for the three simulations

(c) The trajectory of the two biased coin models are shown in Figure 1.5 and 1.6.

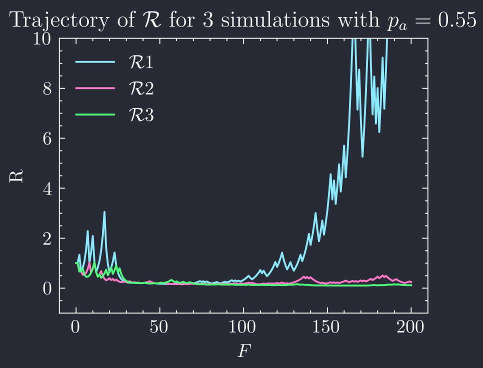

Figure 1.5: There isn't clear evidence for a bias in the first 200 flips

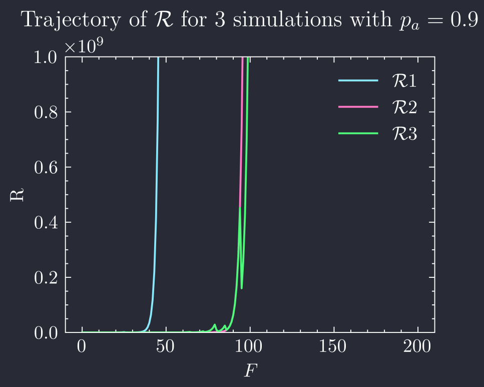

Figure 1.6: A clear bias is shown in the first 200 flips here

and the posterior distribution for the two biased coin models are shown in Figure 1.7 and 1.8.

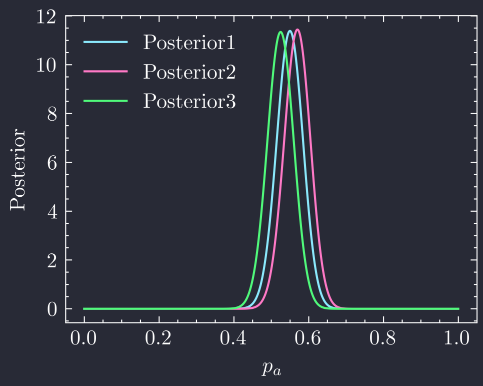

Figure 1.7: We can see that the distribution is centered around roughly $p_a = 0.55$ as expected

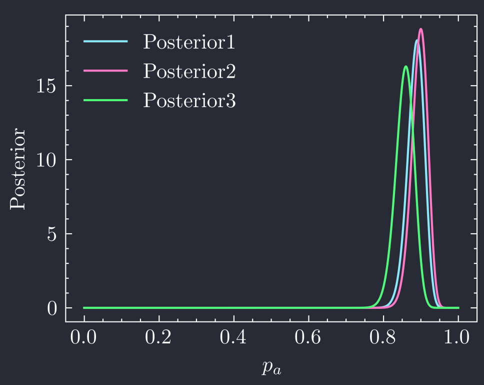

Figure 1.8: There is definitely a bias towards $p_a = 0.9$ in this distribution

(d) From the Gaussian, we can approximate the error bars as $p_a - \frac{1}{2}$, so

$$p_a - 0.5 = \frac{\sigma}{\sqrt{F}} \to F = \frac{\sigma^2}{(p_a - 0.5)^2}$$

so we can get a rough estimate of the number of flips needed to distinguish between the two models for within 2-3 standard deviations as shown in 1.9 and 1.10.

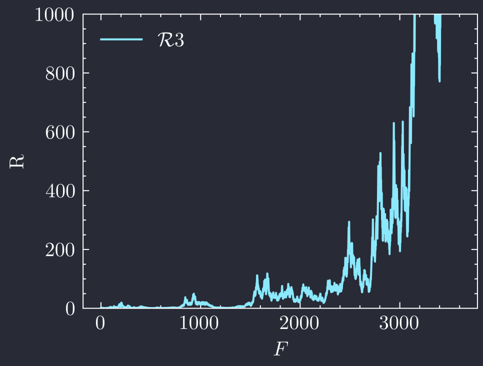

Figure 1.9: For $\sigma = 3$ the model is clearly distinguishable after ≈ 3600 flips

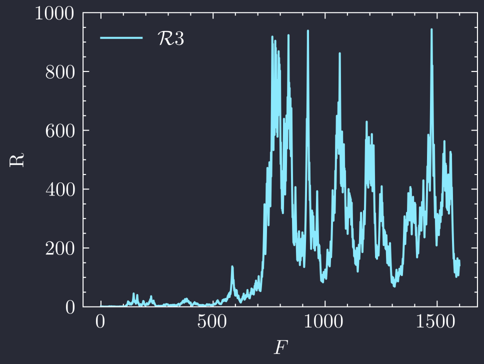

Figure 1.10: For $sigma = 2$ the model does seem to to be biased, but it is not as confident as the previous case

(e) It is suprising to find out that finding evidence against $\mathcal{H}_1$ would be slower than finding evidence for it, but in hindsight it makes sense for cases where the biased probabilities are close to the unbiased coin model since we would need an exponential number of flips as the bent coin model probabilities is closer to $p_o = 1/2$.
From the log of the ratio

$$\begin{aligned}\ln \mathcal{R} &= F \ln 2 + \ln F_a! + \ln F_b! - \ln(F+1)! \\ &= F \ln 2 + \ln(F - F_b)! + \ln(F - F_a)! - \ln(F+1)! \\ &= F \ln 2 + (F - F_b)\ln(F - F_b) - (F - F_b) + \frac{1}{2}\ln 2\pi(F - F_b)\dots\end{aligned}$$

taking the derivative

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}F}\ln \mathcal{R} &= \ln 2 + 1 + \ln(F - F_b) - 1 + \frac{1}{2}\frac{1}{F - F_b}\dots \\ &= \ln 2 + \ln(F_a) + \frac{1}{2F_a} + \ln(F_b) + \frac{1}{2F_b} - \ln(F+1) - \frac{1}{2(F+1)}\end{aligned}$$

The fastest $\log \mathcal{R}$ can grow is when F is very large, and the fractional terms at very large F will be negligible and the log terms grow much slower (e.g. $\ln\left(10^{20}\right) \approx 46$), so the derivative will be be approximately a constant which relates to a linear growth. The fastest $\log \mathcal{R}$ can fall is when F is small and where the $-\ln(F+1)$ term will dominate thus the growth will be approximately logarithmic.

PYTHON CODE BELOW

```python
# %%
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import matplotlib as mpl
import scienceplots
import gmpy2 as gm

# Science plot package + Dracula theme
plt.style.use(['science', 'dark_background'])
plt.rcParams['axes.facecolor'] = '#282a36'
plt.rcParams['figure.facecolor'] = '#282a36'
colorcycle = ['#8be9fd', '#ff79c6', '#50fa7b', '#bd93f9', '#ffb86c', '#ff5555', '#f1fa8c
    ',
'#6272a4']
plt.rcParams['axes.prop_cycle'] = mpl.cycler(color=colorcycle)
white = '#f8f8f2' # foreground

# change dpi
plt.rcParams['figure.dpi'] = 1024

# %%
# 3b
# Calculating the ratio R
# Stirling's approximation
def power(n, exp):
    result = 1
    for _ in range(exp):
        result = gm.mul(n, result)
    return result

def fact(n):
    if n < 10:
        return np.math.factorial(n)
    a = power(n, n)
    b = gm.exp(-n)
    c = gm.sqrt(2 * np.pi * n)
    return a * b * c

def logratio(F_a,F_b):
    return (F_a + F_b) * gm.log(2) + gm.log(fact(F_a)) + gm.log(fact(F_b)) - gm.log(fact
    (F_a + F_b + 1))

def flip():
    if np.random.rand() < 0.5:
        return 1
    else:
        return 0

def simulate(n):
    F_a = 0
    F_b = 0
    R = [1]
    for i in range(n):
        if flip():
            F_a += 1
        else:
            F_b += 1
        R.append(logratio(F_a,F_b))
    return R
print(fact(100))
print(2 * gm.log(fact(100)))
print(gm.log(fact(201)))
print(logratio(100, 100))
# numpy array with 1 to 200 on the x-axis
n = np.arange(0, 201)
# calculate the ratio for each n
# R = simulate(200)
# simulating 3 times and plotting them side by side
plt.figure()
for i in range(3):
    R = simulate(200)
    # Plot on the first subplot
    plt.plot(n, R, label='log $\mathcal{R}$' + str(i+1))
    plt.xlabel('$F$')
    plt.ylabel('R')
plt.title('Part 3b: log $\mathcal{R}$ for 3 simulations')
plt.legend()
plt.show()

# plot the ratio
# plt.plot(n, R, label='R')
# plt.xlabel('$F$')
# plt.ylabel('R')
# plt.legend()
# plt.show()

# %%
# for p_a = 0.55
def bflip():
    if np.random.rand() < 0.55:
        return 1
    else:
        return 0

def vbflip():
    if np.random.rand() < 0.9:
        return 1
    else:
        return 0

Fa_toss_count_55 = []
Fa_toss_count_90 = []

def ratio(F_a,F_b):
    return fact(F_a) * fact(F_b) / fact(F_a + F_b + 1) * 2 ** (F_a + F_b)

def simulate_b(n):
    F_a = 0
    F_b = 0
    R = [1]
    for i in range(n):
        if bflip():
            F_a += 1
        else:
            F_b += 1
        R = np.append(R, ratio(F_a,F_b))
    return R, F_a

print(2 ** 100 * gm.factorial(90) * gm.factorial(10) / gm.factorial(101))

print(2 ** 500 * gm.factorial(275) * gm.factorial(225) / gm.factorial(501))
def simulate_vb(n):
    F_a = 0
    F_b = 0
    R = [1]
    for i in range(n):
        if vbflip():
            F_a += 1
        else:
            F_b += 1
        R = np.append(R, ratio(F_a,F_b))
    return R, F_a

# simulating 3 times and plotting them
plt.figure()
for i in range(3):
    R, F_a = simulate_b(200)
    Fa_toss_count_55.append(F_a)
    plt.plot(n, R, label='$\mathcal{R}$' + str(i+1))
plt.xlabel('$F$')
plt.ylabel('R')
plt.title('Trajectory of $\mathcal{R}$ for 3 simulations with $p_a = 0.55$')
plt.ylim(-1, 10)
plt.legend()

# for p_a = 0.9
plt.figure()
for i in range(3):
    R, F_a = simulate_vb(200)
    Fa_toss_count_90.append(F_a)
    plt.plot(n, R, label='$\mathcal{R}$' + str(i+1))
plt.xlabel('$F$')
plt.ylabel('R')
plt.title('Trajectory of $\mathcal{R}$ for 3 simulations with $p_a = 0.9$')
plt.legend()
plt.ylim(0, 1e9)
plt.show()

# %%
# posterior distribution for p_a = 0.55
def normalconst(F_a):
    return gm.factorial(F_a) * gm.factorial(200 - F_a) / gm.factorial(201)

def posterior(F_a, p_a):
    return p_a ** (F_a) * (1 - p_a) ** (200 - F_a) / normalconst(F_a)

for i, F_a in enumerate(Fa_toss_count_55):
    p_a = np.linspace(0, 1, 1000)
    plt.plot(p_a, posterior(F_a, p_a), label='Posterior' + str(i+1))
plt.xlabel('$p_a$')
plt.ylabel('Posterior')
plt.legend()
plt.show()

# posterior distribution for p_a = 0.9
for i, F_a in enumerate(Fa_toss_count_90):
    p_a = np.linspace(0, 1, 1000)
    plt.plot(p_a, posterior(F_a, p_a), label='Posterior' + str(i+1))
plt.xlabel('$p_a$')
plt.ylabel('Posterior')
plt.legend()
plt.show()

# %%
# back-envelope calculation for p_a close to 0.5
# for a Gaussian the width of the error bars is sigma / sqrt(n)
# the width should be less than (p_a - 0.5)
# p_a - 0.5 > sigma / sqrt(n)
# n > sigma ** 2 / (p_a - 0.5) ** 2
def backenvelope(sigma):
    return sigma ** 2 / (0.55 - 0.5) ** 2
print(backenvelope(3))
print(backenvelope(2))

l = np.arange(0, 3601)
plt.figure()
R, F_a = simulate_b(3600)
plt.plot(l, R, label='$\mathcal{R}$' + str(i+1))
plt.xlabel('$F$')
plt.ylabel('R')
plt.ylim(0, 1e3)
plt.legend()

plt.figure()
R, F_a = simulate_b(1600)
plt.plot(np.arange(0,1601), R, label='$\mathcal{R}$' + str(i+1))
plt.xlabel('$F$')
```

```python
plt.ylabel('R')
plt.ylim(0, 1e3)
plt.legend()

# %%
# 2 e
print(np.e ** 500)
```