STAT 201B HW4

Oliver Maynard

Question 1

```
import numpy as np

eggs_data = np.array([22.0, 23.9, 20.9, 23.8, 25.0, 24.0, 21.7, 23.8, 22.8, 23.1, 23.1, 23.5, 23.0, 2
max_egg = eggs_data.max()

alpha_mle = eggs_data.size / (eggs_data.size * np.log(max_egg) - np.sum(np.log(eggs_data)))
print(alpha_mle)
```

```
12.594868873445822
```

Question 4c - decided to do this in python

```
import numpy as np
import pandas as pd
from scipy import optimize, special

# --- Load and prepare data ---
df = pd.read_csv("berkeleyprecip.csv", header=0)
df.columns = [c.strip() for c in df.columns]
df = df.replace(-99999, np.nan)

# Total winter precipitation (Dec + Jan + Feb)
winter_precip = (df["DEC"] + df["JAN"] + df["FEB"]).dropna().values

# Define log-likelihood for Gamma(alpha, beta) where beta = rate
def loglik(params, x):
    alpha, beta = params
    if alpha <= 0 or beta <= 0:
        return -np.inf
    n = len(x)
    ll = (
        n * (alpha * np.log(beta) - special.gammaln(alpha))
        + (alpha - 1) * np.sum(np.log(x))
        - beta * np.sum(x)
    )
    return ll

def neg_loglik(params, x):
    return -loglik(params, x)

# Method of moments starting values
xbar = np.mean(winter_precip)
s2 = np.var(winter_precip, ddof=1)
alpha0 = xbar**2 / s2
beta0 = xbar / s2  # since   = rate = 1/scale
start = np.array([alpha0, beta0])

# Optimize (maximize log-likelihood via minimizing negative)
res = optimize.minimize(
    fun=neg_loglik,
    x0=start,
    args=(winter_precip,),
    bounds=[(1e-6, None), (1e-6, None)],
    method="L-BFGS-B",
    hess=True,
)
```

```python
alpha_hat, beta_hat = res.x

# Approximate observed Fisher information
if hasattr(res, "hess_inv") and hasattr(res.hess_inv, "todense"):
    hess_inv = res.hess_inv.todense()
else:
    hess_inv = np.linalg.inv(res.hess_inv if hasattr(res, "hess_inv") else np.eye(2))

se_alpha, se_beta = np.sqrt(np.diag(hess_inv))

# 95% Confidence Intervals
ci_alpha = (alpha_hat - 1.96 * se_alpha, alpha_hat + 1.96 * se_alpha)
ci_beta = (beta_hat - 1.96 * se_beta, beta_hat + 1.96 * se_beta)

print(f"MLE   (shape): {alpha_hat:.4f}")
print(f"MLE   (rate):  {beta_hat:.6f}")
print(f"95% CI for  : ({ci_alpha[0]:.4f}, {ci_alpha[1]:.4f})")
print(f"95% CI for  : ({ci_beta[0]:.6f}, {ci_beta[1]:.6f})")

if res.success:
    print("\nOptimization converged successfully.")
    print("Log-likelihood appears smooth near optimum → likely global maximum.")
else:
    print("\nOptimization did not fully converge - try different starting values.")
```

```
MLE   (shape): 5.2589
MLE   (rate):  0.003882
95% CI for  : (3.7026, 6.8152)
95% CI for  : (0.002676, 0.005087)

Optimization converged successfully.
Log-likelihood appears smooth near optimum → likely global maximum.
```

```
/tmp/ipykernel_3475081/105422432.py:37: RuntimeWarning:

Method L-BFGS-B does not use Hessian information (hess).
```