BGN: 2191A
P6
Q7

a.i. Let L = likelihood = $P(x_1, x_2, \dots, x_n \mid \alpha)$

$$\log L = \sum_{i=1}^{n} \log P(x_i \mid \alpha)$$

$$= \sum_{i=1}^{n} \log\left(\frac{\alpha}{x_i^{\alpha+1}}\right)$$

$$= \sum_{i=1}^{n} \log\alpha - (\alpha+1)\log x_i$$

Let $\hat{\alpha}$ maximise $\log L$

$$\therefore \left(\frac{\partial}{\partial\alpha}\log L\right)\Big|_{\alpha=\hat{\alpha}} = \sum_{i=1}^{n} \frac{1}{\hat{\alpha}} - \log x_i$$

$$= \frac{n}{\hat{\alpha}} - \sum_{i=1}^{n} \log x_i$$

$$= 0$$

$$\therefore \hat{\alpha} = \frac{n}{\sum_{i=1}^{n} \log x_i}$$

ii. $P(\text{Pareto}(1, \alpha) \le q) = 0.99$

$\therefore q \ge 1$ and $1 - q^{-\alpha} = 0.99$

$\therefore q^{-\alpha} = 0.01 = \frac{1}{100}$

$\therefore q^{\alpha} = 100$

$\therefore q = 100^{1/\alpha}$

iii. Calculate $\hat{\alpha} = \frac{n}{\sum_{i=1}^{n} \log x_i}$ as in part i.

Generate a n datapoints independently sampled from Pareto$(1, \hat{\alpha})$, $\hat{x}_i^{(1)}$. Repeat this a large number, N, times, yielding $\hat{x}_i^{(j)}$ values.

For each iteration, calculate a new $\hat{\alpha}^{(j)}$ and $q^{(j)} = 100^{-\hat{\alpha}^{(j)}}$

Sort the values of $q^{(j)}$ into $q_\uparrow$

$[a = q_\uparrow^{([\frac{s}{2}N])}, b = q_\uparrow^{([(1-\frac{s}{2})N])}]$ is the two-tailed confidence interval at significance level s (confidence $1-s$)

Pseudo code:

$\hat{\alpha} = \frac{n}{\text{sum}(\log(x))}$

$\hat{x} = \text{pareto\_sample}(1, \hat{\alpha}, \text{shape} = (n, N))$

$\hat{\alpha} = \frac{n}{\text{sum}(\log(\hat{x}), \text{axis} = 0)}$

$q = 100^{-\hat{\alpha}}$

$q = \text{sort}(q)$

$a, b = q[[\frac{s}{2}N]], q[[(1-\frac{s}{2})N]]$

e.g. $N = 10000$, $s = 5\%$.

b. Generate n data points sampled according the the empirical distribution of $\{x_1, \ldots, x_n\}$.
Repeat N times to get N datasets, where N is large

sort each dataset and access position $\lfloor 0.99n \rfloor$ for each of them, q.

If s is the significance level (1 − confidence) then to get a two-tailed confidence interval, calculate

$$[a = q[\lfloor \tfrac{s}{2} N \rfloor], b = q[\lfloor (1 - \tfrac{s}{2}) N \rfloor]]$$

Pseudo code:

```
r = random_choice(x, shape=(n, N))
r = sort(r, axis=0)

q = r[⌊0.99n⌋, :]

q = sort(q)

a, b = q[⌊s/2 N⌋], q[⌊(1-s/2) N⌋]
```

This would be unreliable when n is small, and so the empirical distribution of x would be unlikely to closely resemble the "true" underlying distribution of each x_i.

c. For some confidence level c, we wish to find a, b such that

$$\mathbb{P}(a \leq Q \leq b) = c$$

where Q is the random variable representing the 99th percentile

We must assign a prior (arbitrarily) to X, say Normal(0, 1)

We then calculate the likelihood of the data given a parameter (e.g. μ) and multiply by the prior (and normalise) to give the posterior for X.

Using this, calculate the posterior for Q.

Pseudocode:

```
μ = linspace(-100, 100, δμ)
prior = Normal(0, 1)
likelihood = [Normal(μ, 1).pdf(x_i) for x_i in x]
likelihood = product(likelihood, axis = 0)
posterior = likelihood × product(prior.pdf(x_i) for x_i in x)
posterior = posterior ÷ sum(posterior · δμ)

// note: posterior is a function of μ_i (Pr(μ_i | data))

percentile = Normal(μ, 1).cdf(0.99) // range of percentiles

perm = argsort(posterior)
percentile, posterior = percentile[perm], posterior[perm]
a = cumsum(posterior)
s = 1 - c        // significance
i, j = first_above(posterior, s/2), last_below(posterior, 1 - s/2)
a, b = percentile[i], percentile[j] // [a, b] is the confidence interval.
```