University of Liège

Examination project

MATH2022 - Monte Carlo methods in statistics

Duy Vu Dinh (s2401627)

Academic year 2024-2025

Contents

List of Figures

List of Tables

1 Starting from the general form $\eta x^{\alpha-1}(1-x)^{\beta-1}$, derive the normalizing constant for the beta distribution.

The Beta distribution, defined over the interval $x \in (0,1)$, takes general unnormalized form:

$$f(x \mid \alpha, \beta) = \eta x^{\alpha-1}(1-x)^{\beta-1} \tag{1.1}$$

where $\alpha > 0$, $\beta > 0$, and η is a normalizing constant chosen so that the total probability integrates to 1:

$$\int_0^1 f(x \mid \alpha, \beta)\, dx = 1 \tag{1.2}$$

$$\eta \int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx = 1 \quad \text{(since Eq. (1.1))} \tag{1.3}$$

The integral on the left-hand side is known as the *Beta function*, defined as:

$$B(\alpha, \beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx \tag{1.4}$$

Therefore, the normalizing constant is the reciprocal of the Beta function:

$$\eta = \frac{1}{B(\alpha, \beta)} \tag{1.5}$$

On the other hand, the Gamma function is defined as:

$$\Gamma(s) = \int_0^\infty t^{s-1} e^{-t} dt, \quad \text{for } s > 0 \tag{1.6}$$

The Beta function can be derived from the Gamma function. Consider the product:

$$\Gamma(\alpha)\Gamma(\beta) = \left(\int_0^\infty x^{\alpha-1} e^{-x} dx\right)\left(\int_0^\infty y^{\beta-1} e^{-y} dy\right) \tag{1.7}$$

$$= \int_0^\infty \int_0^\infty x^{\alpha-1} y^{\beta-1} e^{-(x+y)} dx\, dy \tag{1.8}$$

Perform the change of variables: $t = x + y$, $x = t\mu$, $y = t(1-\mu)$ where $t \in (0, \infty)$ and $\mu \in (0,1)$ (because of $x \in (0,t)$). The Jacobian of this transformation is $|J| = t$, so $dx\, dy = t\, d\mu\, dt$. The integral becomes:

$$\Gamma(\alpha)\Gamma(\beta) = \int_0^1 \int_0^\infty (t\mu)^{\alpha-1}(t(1-\mu))^{\beta-1} e^{-t} t\, dt\, d\mu \tag{1.9}$$

$$= \int_0^1 \mu^{\alpha-1}(1-\mu)^{\beta-1} d\mu \int_0^\infty t^{\alpha+\beta-1} e^{-t} dt \tag{1.10}$$

$$= B(\alpha, \beta) \cdot \Gamma(\alpha+\beta) \tag{1.11}$$

Hence, the Beta function can be expressed in terms of Gamma functions:

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{1.12}$$

Therefore, the normalizing constant η and the probability density function of the Beta distribution are given by:

$$\eta = \frac{1}{B(\alpha, \beta)} = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \quad \text{(since Eq. (1.5))}$$

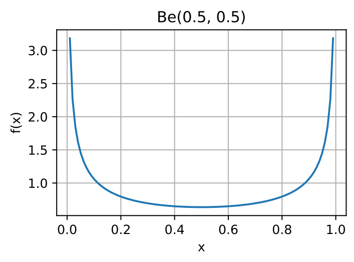
$$f(x \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad x \in (0,1) \tag{1.13}$$

2 Plot the distribution $\mathrm{Be}(\alpha, \beta)$ for different values of its shape parameters, covering all different possibilities. Describe the shape and skewness of the distribution in function of α and β. What happens if both shape parameters converge to $+\infty$? Can we recover the uniform distribution $\mathcal{U}[0, 1]$ using the beta distribution?

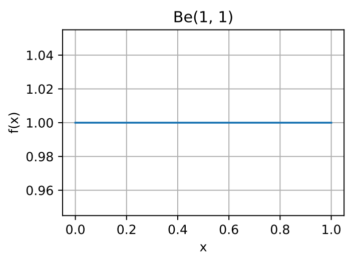
The Beta distribution is a flexible family of continuous probability distributions defined on the interval $[0, 1]$. It is parameterized by two shape parameters α and β, which control the distribution's shape and skewness. Its probability density function is given by:

$$f(x|\alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad x \in (0, 1) \tag{2.1}$$

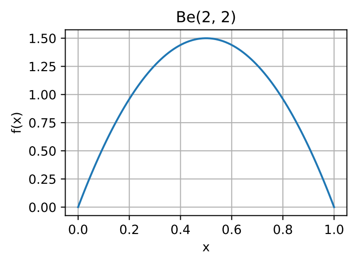
where $B(\alpha, \beta)$ is the Beta function.



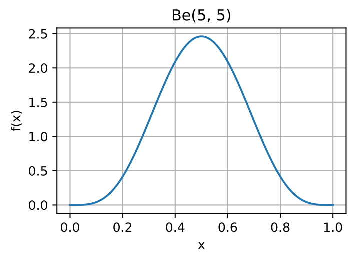
(a) $\alpha = 0.5$ and $\beta = 0.5$



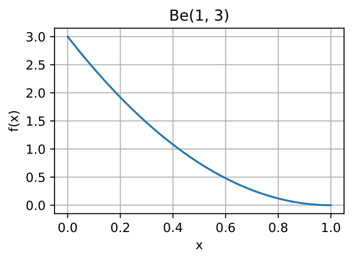
(b) $\alpha = 1$ and $\beta = 1$



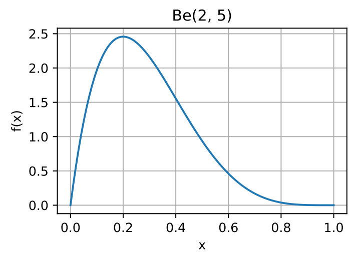
(c) $\alpha = 2$ and $\beta = 2$



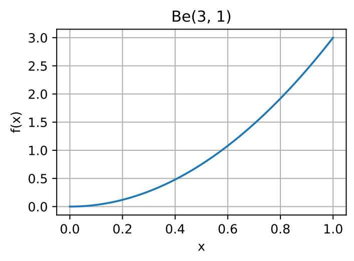
(d) $\alpha = 5$ and $\beta = 5$



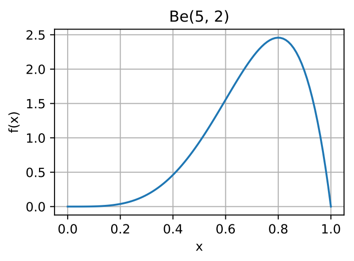
(e) $\alpha = 1$ and $\beta = 3$



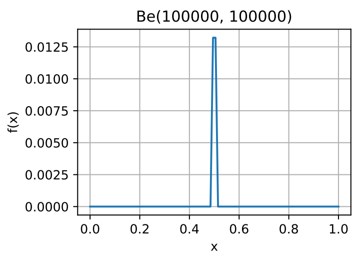
(f) $\alpha = 2$ and $\beta = 5$



(g) $\alpha = 3$ and $\beta = 1$



(h) $\alpha = 5$ and $\beta = 2$



(i) $\alpha = 10^6$ and $\beta = 10^6$

Figure 1: Beta distribution for various combinations of α and β.

Figure 1 shows the Beta distribution for various combinations of shape parameters, highlighting key behaviors such as symmetry, skewness, and limiting cases.

Symmetric cases ($\alpha = \beta$)

- $\alpha = \beta = 0.5$ (Figure 1a): U-shaped distribution with infinite density at the endpoints 0 and 1.
- $\alpha = \beta = 1$ (Figure 1b): The distribution reduces to the uniform distribution $\mathcal{U}[0,1]$. Specifically,
$$f(x \mid 1,1) = \frac{1}{B(1,1)} x^0 (1-x)^0 = 1$$
which implies all outcomes in $[0,1]$ are equally likely.
- $\alpha = \beta = 2$ (Figure 1c): Bell-shaped and symmetric about $x = 0.5$.
- $\alpha = \beta = 10^6$ (Figure 1i): The distribution becomes highly concentrated around $x = 0.5$. As α and β grow very large, the Beta distribution's probability mass increasingly clusters near 0.5, forming a very sharp peak.

Left-skewed cases ($\alpha < \beta$)

- $\alpha = 1, \beta = 3$ (Figure 1e), $\alpha = 2, \beta = 5$ (Figure 1f): The distribution is skewed to the left with higher density near 0.

Right-skewed cases ($\alpha > \beta$)

- $\alpha = 3, \beta = 1$ (Figure 1g), $\alpha = 5, \beta = 2$ (Figure 1h): The distribution is skewed to the right with higher density near 1.

Observations and limits

- When $\alpha, \beta < 1$, the distribution becomes U-shaped or bimodal, diverging at 0 and 1.
- When $\alpha, \beta > 1$, the distribution becomes unimodal and more peaked around the center.
- When $\alpha = \beta = 1$, we recover the uniform distribution $\mathcal{U}[0,1]$.
- As $\alpha, \beta \to +\infty$ with $\alpha = \beta$, the distribution concentrates more and more around $x = 0.5$, forming an increasingly sharp peak near that point.

In general, the Beta distribution exhibits a wide range of shapes depending on the values of α and β, making it an extremely flexible model for distributions over the unit interval. Its ability to model skewness, symmetry, uniformity, and concentration makes it particularly useful in Bayesian modeling and probabilistic simulations.

3 Can we apply the inverse transform method to sample random values of a beta distribution? If yes, explain and implement the algorithm. Simulate 15,000 values of a Be$(3, 9)$ distribution. If no, explain why not.

Applicability to the Beta distribution

The inverse transform method is a fundamental technique to sample random variables from a given distribution using its CDF. It is based on the fact that if $U \sim \mathcal{U}[0, 1]$ and F is a continuous CDF, then the random variable $X = F^{-1}(U)$ has distribution F.

For a Be(α, β) distribution, the CDF is given by the regularized incomplete Beta function:

$$F(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} \int_0^x t^{\alpha-1}(1-t)^{\beta-1} dt \tag{3.1}$$

Although the Beta CDF is smooth and strictly increasing on the interval $[0, 1]$, it does not have a closed-form inverse for arbitrary α and β. This means that analytically inverting F to find F^{-1} is not feasible.

However, in practice, numerical algorithms can approximate F^{-1} to a high degree of accuracy. Scientific libraries such as `scipy.stats.beta.ppf` in Python implement this inverse CDF, referred to as the PPF. The PPF satisfies $F\big(F^{-1}(p)\big) = p$, with $p \in (0, 1)$. This allows us to generate Beta-distributed random variables by transforming uniform random variables via the PPF. The monotonicity and continuity of the Beta CDF guarantee the existence of a well-defined inverse function, ensuring the correctness of this sampling method.

Simulation and results

The inverse transform sampling procedure implemented is as follows:

1. Generate $U_1, U_2, \ldots, U_{15000} \sim \mathcal{U}[0, 1]$.
2. Compute $X_i = F^{-1}(U_i)$ using the Beta quantile function.
3. Return the sample $\{X_1, \ldots, X_{15000}\}$.

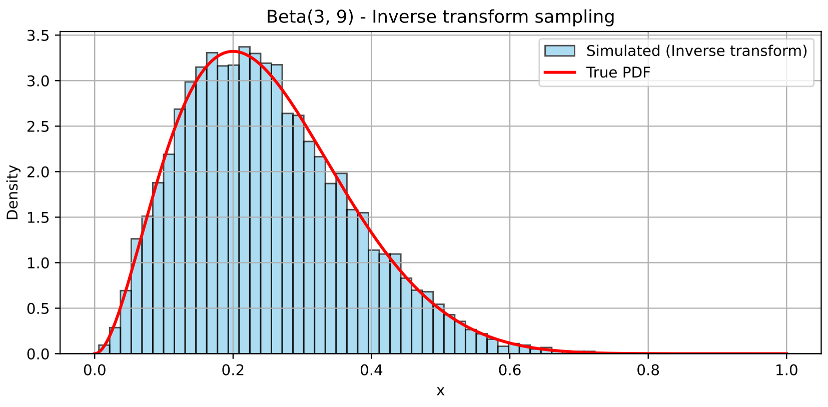


Figure 2: Histogram of 15,000 samples from Be$(3, 9)$ using inverse transform sampling.

Figure 2 illustrates that the histogram of generated values aligns closely with the true Beta(3, 9) probability density function. This confirms that the numerical inverse transform method is a valid and practical approach to sampling from Beta distributions.

4 In Chapter 3, a formula is given to simulate $\mathrm{Be}(\alpha, \beta)$ from the uniform distribution, given that both parameters α and β are natural numbers (and not equal to 0). Implement this transformation method algorithm and apply it to simulate 15,000 values of a $\mathrm{Be}(3, 9)$ distribution.

Theoretical background

Let $U_j \sim \mathcal{U}[0, 1]$ be i.i.d. uniform random variables. Then, as shown in chapter 3 of the lecture:

$$X = \frac{\sum_{j=1}^{\alpha} - \log(U_j)}{\sum_{j=1}^{\alpha+\beta} - \log(U_j)} \sim \mathrm{Be}(\alpha, \beta), \quad \text{for } \alpha, \beta \in \mathbb{N} \tag{4.1}$$

This transformation is numerically more stable than working directly with products of uniforms, especially when α or β is large.

Simulation and results

To simulate 15,000 values from $\mathrm{Be}(3, 9)$:

1. Generate a $15{,}000 \times 12$ matrix of $U_{ij} \sim \mathcal{U}[0, 1]$.
2. Compute $S_3 = \sum_{j=1}^{3} - \log(U_j)$ and $S_{12} = \sum_{j=1}^{12} - \log(U_j)$ for each sample.
3. Return the ratio $X = S_3/S_{12}$.

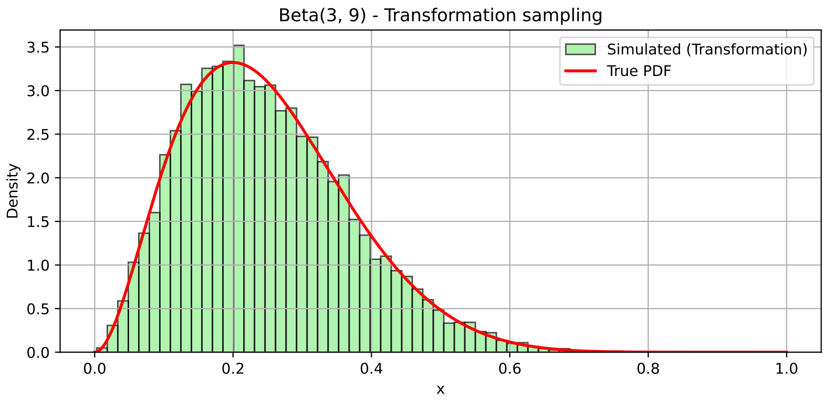


Figure 3: Histogram of 15,000 samples from $\mathrm{Be}(3, 9)$ using the transformation method.

As shown in Figure 3, the simulated samples closely follow the theoretical $\mathrm{Be}(3, 9)$ density. The transformation method based on uniform variables offers an accurate and efficient approach when the shape parameters are integers.

5 It is also possible to generate a beta distribution using gamma distributions. A gamma distribution $Ga(\alpha, \beta)$ is defined by the density function $f(x|\alpha,\beta) := \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \mathbb{I}_{[0,+\infty[}(x)$for $\alpha, \beta > 0$.

(a) Show formally that if $Y_1 \sim Ga(\alpha, 1)$ and $Y_2 \sim Ga(\beta, 1)$ then $X = \frac{Y_1}{Y_1+Y_2} \sim Be(\alpha, \beta)$.

Let $Y_1 \sim \text{Ga}(\alpha, 1)$ and $Y_2 \sim \text{Ga}(\beta, 1)$, where the gamma density function is defined as:

$$f_{Y_1}(y_1 \mid \alpha, 1) = \frac{1}{\Gamma(\alpha)} y_1^{\alpha-1} e^{-y_1}, \quad y_1 > 0, \tag{5.1}$$

$$f_{Y_2}(y_2 \mid \beta, 1) = \frac{1}{\Gamma(\beta)} y_2^{\beta-1} e^{-y_2}, \quad y_2 > 0. \tag{5.2}$$

Since Y_1 and Y_2 are independent, their joint density is the product:

$$f_{Y_1,Y_2}(y_1, y_2 \mid \alpha, \beta) = \frac{1}{\Gamma(\alpha)\Gamma(\beta)} y_1^{\alpha-1} y_2^{\beta-1} e^{-(y_1+y_2)}, \quad y_1, y_2 > 0. \tag{5.3}$$

Define the new variable that $X = \frac{Y_1}{Y_1+Y_2}$, $T = Y_1 + Y_2$. This transformation maps $(Y_1, Y_2) \mapsto (X, T)$, with inverse of $Y_1 = XT$, $Y_2 = (1 - X)T$. To derive the joint density of (X, T), we compute the Jacobian determinant:

$$J = \left| \frac{\partial(Y_1, Y_2)}{\partial(X, T)} \right| = \begin{vmatrix} \frac{\partial Y_1}{\partial X} & \frac{\partial Y_1}{\partial T} \\ \frac{\partial Y_2}{\partial X} & \frac{\partial Y_2}{\partial T} \end{vmatrix} = \begin{vmatrix} T & X \\ -T & 1 - X \end{vmatrix} = T(1 - X) + TX = T$$

Now, the joint density becomes:

$$\begin{aligned} f_{X,T}(x, t \mid \alpha, \beta) &= f_{Y_1,Y_2}(xt, (1 - x)t \mid \alpha, \beta) \cdot |J| && (5.4) \\ &= \frac{1}{\Gamma(\alpha)\Gamma(\beta)} (xt)^{\alpha-1} e^{-xt} \cdot ((1 - x)t)^{\beta-1} e^{-(1-x)t} \cdot t && (5.5) \\ &= \frac{1}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1 - x)^{\beta-1} t^{\alpha+\beta-1} e^{-t} && (5.6) \end{aligned}$$

for $x \in (0, 1)$, $t > 0$.

Using Eq. (1.12), stating that $B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$, the joint density is rewritten as:

$$f_{X,T}(x, t \mid \alpha, \beta) = \underbrace{\frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1 - x)^{\beta-1}}_{f_X(x|\alpha,\beta)} \cdot \underbrace{\frac{1}{\Gamma(\alpha + \beta)} t^{\alpha+\beta-1} e^{-t}}_{f_T(t|\alpha,\beta)}. \tag{5.7}$$

Since the joint density factorizes into a product of marginal densities, X and T are independent. Furthermore, their marginal distributions are:

$$X \sim \text{Be}(\alpha, \beta), \quad x \in (0, 1),$$

$$T \sim \text{Ga}(\alpha + \beta, 1), \quad t > 0.$$

(b) Use this relation to construct an algorithm to generate a beta random variable and simulate 15,000 values from a $Be(3, 9)$.

Based on the result in part (a), we can simulate a Beta-distributed random variable using two independent Gamma random variables. Specifically, if $Y_1 \sim \text{Gamma}(\alpha, 1)$ and $Y_2 \sim \text{Gamma}(\beta, 1)$, then the ratio

$$X = \frac{Y_1}{Y_1 + Y_2}$$

follows a $\text{Be}(\alpha, \beta)$ distribution.

Simulation and results

To simulate 15,000 samples from $\text{Be}(3, 9)$, we use the following algorithm:

1. Generate $Y_1 \sim \text{Gamma}(3, 1)$ and $Y_2 \sim \text{Gamma}(9, 1)$.
2. Compute $X = \frac{Y_1}{Y_1+Y_2}$.
3. Repeat this process 15,000 times to obtain a sample of size 15,000.

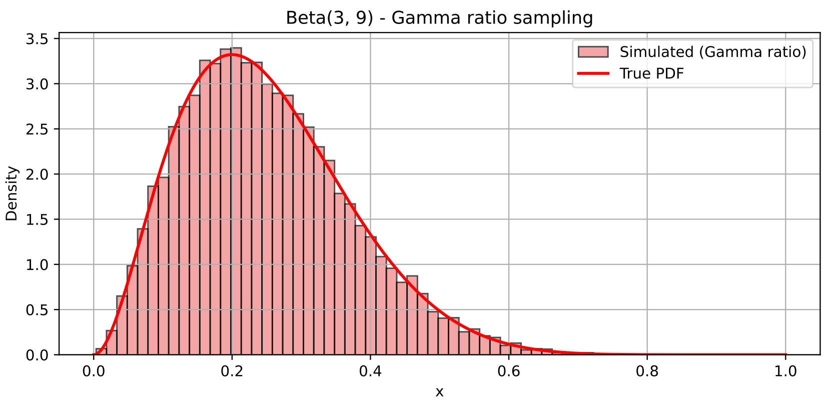


Figure 4: Histogram of 15,000 samples from $\text{Be}(3, 9)$ using the Gamma ratio method (red), with the true density overlaid.

As shown in Figure 4, the empirical distribution obtained from the Gamma ratio sampling aligns very well with the theoretical probability density function of the $\text{Be}(3, 9)$ distribution. This confirms the correctness and efficiency of the method.

6 Compare the implemented procedures from questions 4 and 5 and the beta distribution generator incorporated in R in terms of their runtime (keep $n = 15{,}000$) as well as their precision (use different values of n).

We compared three methods to generate samples from the Beta distribution $\text{Be}(3, 9)$: the transformation method using uniform logarithms (Question 4), the gamma ratio method (Question 5), and the built-in Beta generator from the SciPy library. Runtime was measured for a fixed sample size of $n = 15{,}000$, while precision was evaluated across varying sample sizes. To ensure the stability of the test results, 200 experiments are run, and then the average results are reported.

Runtime comparison: The average runtimes over 200 experiments for $n = 15,000$ samples are shown in Table 1.

Table 1: Average runtime in terms of 15,000 samples for Beta sampling methods.

Method	Transformation	Gamma ratio	Built-in Beta
Average Runtime (s)	0.002252	0.000736	0.000736

The gamma ratio and built-in methods are significantly faster than the transformation method.

Precision comparison: Precision is assessed using the Kolmogorov-Smirnov (K-S) statistic, quantifying the difference between the empirical and theoretical Beta distribution. Lower values indicate better precision. The average K-S statistics are computed over 200 experiments for various sample sizes.

Table 2: Average K-S statistic in terms of different sample sizes for Beta sampling methods.

Sample Size (n)	10	100	1,000	10,000	100,000	1,000,000
Transformation	0.2657	0.0853	0.0273	0.0088	0.0028	0.0009
Gamma Ratio	0.2601	0.0900	0.0276	0.0088	0.0027	0.0009
Built-in Beta	0.2575	0.0872	0.0275	0.0089	0.0028	0.0009

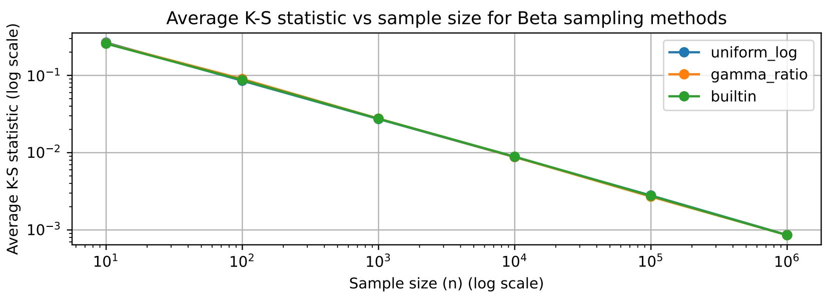


Figure 5: Average K-S statistic in terms of different sample sizes for Beta sampling methods.

Table 2 and Figure 5 show that all three methods yield nearly indistinguishable precision performance over a broad range of sample sizes. Differences between methods are minimal. They produce samples closely matching the Beta distribution, with K–S statistics decreasing as n grows. The K–S statistic decreases approximately linearly on the log-log scale, consistent with theoretical convergence rates.

Regarding both results, they suggest the gamma ratio and built-in methods are preferable for efficient and accurate Beta random variable generation, especially for large-scale simulations.

7 In the case where α and β are larger than 1 (but not necessarily natural numbers), it is possible to use the uniform distribution $\mathcal{U}[0, 1]$ as instrumental distribution.

(a) Explain why this is possible and how to determine the limiting constant C.

In the Accept–Reject (A–R) algorithm, the target density f can be sampled using an instrumental density g if there exists a constant $C \geq 1$ such that:

$$f(x) \leq Cg(x) \quad \forall x \in \operatorname{supp}(f) \tag{7.1}$$

For the Beta distribution $\operatorname{Be}(\alpha, \beta)$ with $\alpha > 1$ and $\beta > 1$, the probability density function is:

$$f(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad x \in [0, 1] \tag{7.2}$$

This density is smooth, unimodal, and bounded on $[0, 1]$. Hence, the uniform distribution $\mathcal{U}[0, 1]$ is a natural choice for the instrumental distribution because:

- It shares the same support, $[0, 1]$.
- Its density is constant: $g(x) = 1$.

With $g(x) = 1$, the density ratio simplifies to:

$$\frac{f(x)}{g(x)} = f(x) \tag{7.3}$$

and the limiting constant becomes:

$$C = \sup_{x \in [0,1]} f(x) \tag{7.4}$$

When $\alpha > 1$ and $\beta > 1$, the mode of the Beta distribution lies strictly inside $(0, 1)$ and is given by:

$$x_{\text{mode}} = \frac{\alpha - 1}{\alpha + \beta - 2} \tag{7.5}$$

Therefore, C can be computed by evaluating the Beta PDF at the mode:

$$C = f(x_{\text{mode}}) = \frac{1}{B(\alpha, \beta)} \left(\frac{\alpha - 1}{\alpha + \beta - 2}\right)^{\alpha-1} \left(1 - \frac{\alpha - 1}{\alpha + \beta - 2}\right)^{\beta-1} \tag{7.6}$$

(b) Determine (using R) this limiting constant C used in the Accept-Reject algorithm if the target distribution is Be$(3.3, 9.5)$.

To apply the Accept–Reject algorithm with the uniform instrumental distribution $\mathcal{U}[0,1]$, the constant C must satisfy:

$$\frac{f(x)}{g(x)} \leq C, \quad \forall x \in [0,1] \tag{7.7}$$

where $f(x)$ is the Beta PDF and $g(x) = 1$.

For $\alpha > 1$ and $\beta > 1$, the maximum of $f(x)$ occurs at the mode:

$$x_{\text{mode}} = \frac{\alpha - 1}{\alpha + \beta - 2} \tag{7.8}$$

Substituting $\alpha = 3.3$ and $\beta = 9.5$ yields:

$$x_{\text{mode}} = \frac{3.3 - 1}{3.3 + 9.5 - 2} = \frac{2.3}{10.8} \approx 0.213$$

The limiting constant is then:

$$C = f(x_{\text{mode}}) = \frac{1}{B(3.3, 9.5)} \cdot (0.213)^{2.3} \cdot (1 - 0.213)^{8.5}$$

Numerical evaluation gives:

$$C \approx 3.3681$$

(c) Implement the algorithm and perform 15,000 simulations to obtain values from a Be$(3.3, 9.5)$ distribution.

To simulate values from a Be$(3.3, 9.5)$ distribution, we implemented the Accept–Reject algorithm using the uniform distribution $\mathcal{U}[0,1]$ as the instrumental distribution and a limiting constant $C \approx 3.3681$ (computed in part (b)).

The procedure is as follows:

1. Sample $x \sim \mathcal{U}[0,1]$.
2. Sample $u \sim \mathcal{U}[0,1]$.
3. Accept x if $u < \frac{f(x)}{C}$, where $f(x)$ is the Beta density.
4. Repeat until 15,000 accepted values are obtained.

In total, we simulated values until 15,000 accepted samples were obtained.

(d) Create a plot that shows all simulated values, distinguishing between the accepted (in green) and rejected values (in red). Also indicate the instrumental density and the true density function of the Be$(3.3, 9.5)$ distribution. Calculate the acceptance rate.

The results of the Accept–Reject simulation from part (c) are visualized in the figure below. We distinguish between:

- Accepted samples shown in green, these follow the target Be$(3.3, 9.5)$ distribution.
- Rejected proposals shown in red, these are the values that did not satisfy the acceptance criterion.

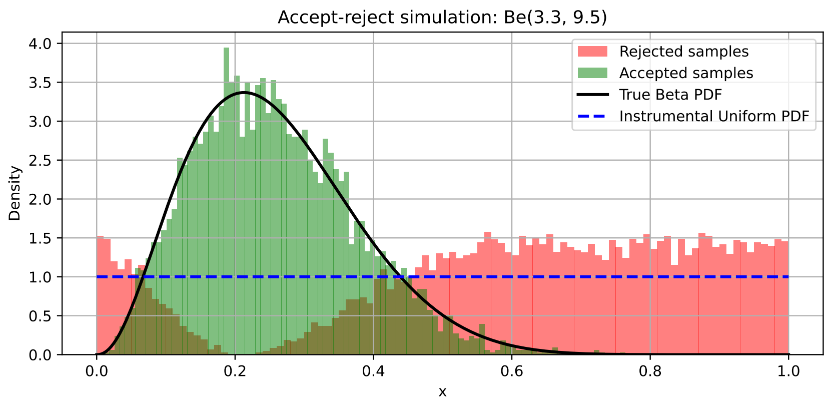


Figure 6: Accepted and rejected samples with uniform distribution as instrumental density (15,000 samples).

We observe from Figure 6 that accepted values are concentrated around the mode of the Beta distribution (approximately $x \approx 0.213$), while rejected values are more common in low-density tail regions.

The empirical acceptance rate was calculated as:

$$\text{Acceptance rate} = \frac{\#\text{ accepted}}{\#\text{ accepted} + \#\text{ rejected}} = \frac{\#\text{ accepted}}{15{,}000} \approx 29.37\%$$

(e) Show, in a general setting, that the probability of acceptance in an Accept-Reject algorithm with limiting constant C on the density ratio f/g is equal to $1/C$. Compare this result with the calculated value in the previous question.

Consider the Accept–Reject method where:

- $f(x)$ is the target density.
- $g(x)$ is the instrumental (proposal) density.
- There exists a constant $C \geq 1$ such that:

$$f(x) \leq Cg(x), \quad \forall x \tag{7.9}$$

- $X \sim g$ and $U \sim \mathcal{U}[0,1]$ are independent.

Define the acceptance event as:

$$A := \left\{ U \leq \frac{f(X)}{Cg(X)} \right\} \tag{7.10}$$

The probability of acceptance is:

$$P(A) = \mathbb{E}_X \left[P\left(U \leq \frac{f(X)}{Cg(X)} \mid X \right) \right] \tag{7.11}$$

Since U is uniform on $[0,1]$, this inner probability is simply:

$$P\left(U \leq \frac{f(X)}{Cg(X)} \mid X \right) = \frac{f(X)}{Cg(X)} \tag{7.12}$$

Therefore,

$$\begin{aligned} P(A) &= \int_0^1 \frac{f(x)}{Cg(x)} g(x)\, dx && (7.13) \\ &= \frac{1}{C} \int_0^1 f(x)\, dx && (7.14) \\ &= \frac{1}{C} && (7.15) \end{aligned}$$

Since f is a probability density function, it integrates to 1. Thus, the acceptance probability is exactly the reciprocal of the constant C.

Comparison with the Beta simulation:

From part (b), the limiting constant was computed as $C \approx 3.3681$, which implies a theoretical acceptance probability of:

$$p_{\text{accept}} = \frac{1}{C} \approx 29.69\%$$

From the empirical results in part (d), the observed acceptance rate was approximately 29.37%.

The close agreement between theory and empirical results validates the correctness of the Accept–Reject algorithm and confirms that the expected acceptance rate is $1/C$.

(f) Rewrite the algorithm to obtain $n = 15{,}000$ simulated (accepted) values from the Be$(3.3, 9.5)$ distribution. Does this influence the acceptance rate obtained by the Accept-Reject algorithm? Plot the simulated values and compare with the true distribution. Discuss your findings.

The revised algorithm proceeds as follows:

1. Repeat:
   - Sample $x \sim \mathcal{U}[0, 1]$
   - Sample $u \sim \mathcal{U}[0, 1]$
   - Accept x if $u < \frac{f(x)}{C}$, where $f(x)$ is the Beta(3.3, 9.5) density and $C \approx 3.3681$
2. Stop when 15,000 accepted values are obtained.

The empirical acceptance rate is:

$$\text{Acceptance rate} = \frac{\#\ \text{accepted}}{\#\ \text{accepted} + \#\ \text{rejected}} = \frac{15{,}000}{15{,}000 + \#\ \text{rejected}} \approx 29.61\%$$

This value is nearly identical to the rate observed in part (d), confirming that the process of stopping at exactly 15,000 accepted values does not significantly influence the acceptance rate. The theoretical rate remains approximately $1/C \approx 29.69\%$, with only minor variation due to stochastic sampling.

The Figure 7 shows the histogram of the simulated values, compared against the true density of the Be$(3.3, 9.5)$ distribution.

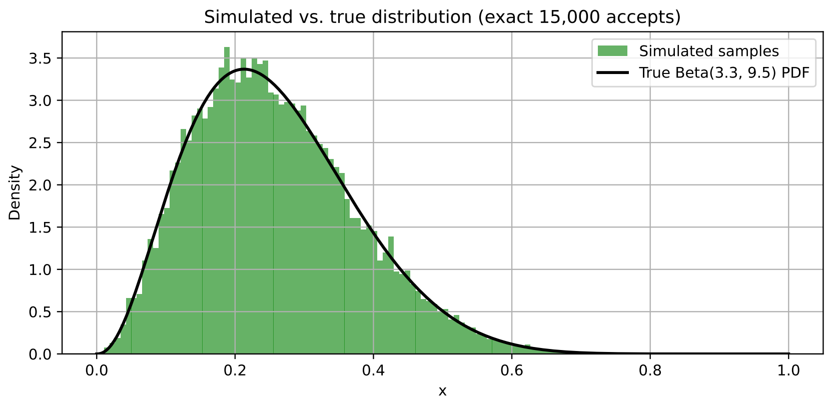


Figure 7: Simulated values and true Beta(3.3, 9.5) PDF.

The simulated histogram closely follows the true Beta density, with the mode and overall shape accurately reproduced. Fixing the number of accepted samples does not affect the theoretical acceptance rate, which remains approximately $1/C \approx 29.69\%$. The empirical acceptance rate of approximately 29.61% confirms the consistency and robustness of the Accept–Reject algorithm under this configuration. This approach is practical in applications where a fixed sample size is required regardless of the number of trials.

8 Another option for the instrumental distribution in the Accept-Reject algorithm is to use another beta distribution. Since we saw before how to simulate random values from a beta distribution for which the shape parameters take on natural values, we will use this kind of beta distribution.

(a) Show formally that, for the ratio f/g to be bounded when f is a Be(α, β) density and g is a Be(a, b) density, we must have both $a \leq \alpha$ and $b \leq \beta$. Determine the maximal ratio in terms of α, β, a, and b.

Let $f(x)$ and $g(x)$ be the probability density functions of Be(α, β) and Be(a, b), respectively:

$$f(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad x \in [0,1] \tag{8.1}$$

$$g(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}, \quad x \in [0,1] \tag{8.2}$$

Consider the ratio of densities:

$$\frac{f(x)}{g(x)} = \frac{\frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}}{\frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}} \tag{8.3}$$

$$= \frac{B(a,b)}{B(\alpha,\beta)} \cdot x^{\alpha-a}(1-x)^{\beta-b} \tag{8.4}$$

Since the support is the interval $[0, 1]$, for the ratio $\frac{f}{g}$ to be bounded on $[0, 1]$, the powers of x and $(1 - x)$ must not cause divergences at the boundaries:

- If $\alpha - a < 0$, then $x^{\alpha-a} \to \infty$ as $x \to 0$.
- If $\beta - b < 0$, then $(1 - x)^{\beta-b} \to \infty$ as $x \to 1$.

Hence, to avoid unbounded behavior: $\alpha - a \geq 0$ and $\beta - b \geq 0 \Rightarrow a \leq \alpha, \; b \leq \beta$

Finding the maximum of the ratio

Define:

$$h(x) = x^{\alpha-a}(1-x)^{\beta-b}, \quad x \in (0,1) \tag{8.5}$$

The derivative is:

$$h'(x) = (\alpha - a)x^{\alpha-a-1}(1-x)^{\beta-b} - (\beta - b)x^{\alpha-a}(1-x)^{\beta-b-1} \tag{8.6}$$

$$= x^{\alpha-a-1}(1-x)^{\beta-b-1}\left[(\alpha - a)(1-x) - (\beta - b)x\right] \tag{8.7}$$

The critical points satisfy:

$$(\alpha - a)(1-x) - (\beta - b)x = 0 \quad \Rightarrow \quad x = \frac{\alpha - a}{\alpha - a + \beta - b} \tag{8.8}$$

The maximum of the ratio $\frac{f}{g}$ is attained at this point:

$$C = \max_{x\in[0,1]} \frac{f(x)}{g(x)} = \frac{B(a,b)}{B(\alpha,\beta)} \left(\frac{\alpha - a}{\alpha - a + \beta - b}\right)^{\alpha-a} \left(\frac{\beta - b}{\alpha - a + \beta - b}\right)^{\beta-b}$$

This constant C serves as the limiting constant in the Accept–Reject algorithm when using Be(a, b) as instrumental to sample from Be(α, β).

(b) Redo questions 7(b) to 7(d) with the appropriate beta distribution as instrumental distribution. Compare both methods.

The density ratio is:

$$\frac{f(x)}{g(x)} = \frac{B(3,9)}{B(3.3,9.5)} \cdot x^{0.3}(1-x)^{0.5} \tag{8.9}$$

with its maximum at:

$$x_{\max} = \frac{\alpha - a}{\alpha - a + \beta - b} = \frac{3.3-3}{(3.3-3)+(9.5-9)} = \frac{0.3}{0.8} = 0.375$$

Evaluating the ratio at $x_{\max}$ yields the limiting constant:

$$C \approx 1.0761$$

We performed Accept–Reject sampling until exactly $n = 15,000$ accepted values were obtained. The empirical acceptance rate was:

$$\text{Acceptance rate} = \frac{\#\text{ accepted}}{\#\text{ accepted} + \#\text{ rejected}} = \frac{\#\text{ accepted}}{15{,}000} \approx 92.99\%$$

Figure 8 shows the histogram of accepted and rejected samples, along with the true Beta(3.3, 9.5) density and the instrumental Beta(3, 9) density.

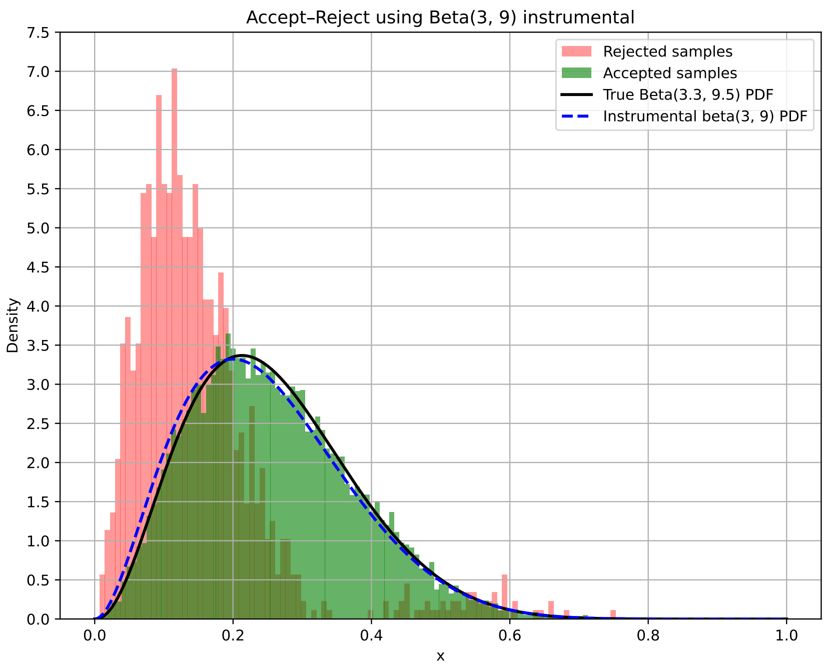


Figure 8: Accept–Reject using Beta(3, 9) as instrumental distribution for sampling from Beta(3.3, 9.5).

- Acceptance rate: Using Beta(3, 9) as the instrumental distribution results in a significantly higher acceptance rate (92.99%) compared to the uniform instrumental method (29.41%).
- Efficiency: Selecting an instrumental distribution close to the target greatly reduces the number of rejected samples, improving computational efficiency.
- Simulation quality: The accepted samples closely match the true Beta distribution, validating the effectiveness of this approach.

9 Show that, for X an observation from the negative binomial distribution $NB(r,p)$, the family of beta distributions $\mathrm{Be}(\alpha,\beta)$ is a family of conjugate priors.

In Bayesian inference, the unknown parameter p is modeled as a random variable with prior distribution $\pi(p)$. After observing data x, Bayes' theorem updates this belief via:

$$\pi(p \mid x) = \frac{f(x \mid p)\pi(p)}{\int f(x \mid p)\pi(p)\,dp} \tag{9.1}$$

where $f(x \mid p)$ is the likelihood function.

Assuming a Beta prior for p with parameters $\alpha, \beta > 0$:

$$\pi(p) = \frac{1}{B(\alpha,\beta)} p^{\alpha-1}(1-p)^{\beta-1}, \quad p \in (0,1) \tag{9.2}$$

and a Negative Binomial likelihood for the number of failures $x \in \mathbb{N}_0$ before r successes:

$$f(x \mid r,p) = \binom{x+r-1}{x} p^r (1-p)^x \tag{9.3}$$

the posterior distribution can be expressed (up to a normalization constant independent of p) as:

$$\pi(p \mid x) \propto p^{r+\alpha-1}(1-p)^{x+\beta-1} \tag{9.4}$$

This expression corresponds to the kernel of a Beta distribution, implying the posterior is:

$$p \mid x \sim \mathrm{Be}(r+\alpha,\ x+\beta) \tag{9.5}$$

This result demonstrates that the Beta distribution is conjugate to the Negative Binomial likelihood with respect to p, enabling analytic Bayesian updating.

A closely related example is the Beta–Binomial conjugacy, where if $Y \sim \mathrm{Binomial}(n,p)$ with Beta prior $\mathrm{Be}(\alpha,\beta)$, then the posterior is $\mathrm{Be}(\alpha+y, \beta+n-y)$.

Therefore, the conjugacy property simplifies posterior inference for p in Negative Binomial models by preserving the Beta family form in the posterior distribution.

10 In Chapter 2, we estimated the shape parameters of the beta distribution using the maximum likelihood estimator. The problem, however, has no explicit solution. Another option is to use the Method of Moments (MOM) to determine $\hat{\alpha}$ and $\hat{\beta}$. As the name implies, the moments up to order k will be used to estimate the k parameters of a distribution.

(a) Show that for the gamma function, the following property holds: $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$, $\alpha > 0$. As such, the gamma function can be seen as a generalization of factorials.

The Gamma function is defined for $\alpha > 0$ as:

$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}\,dt \tag{10.1}$$

Also,

$$\Gamma(\alpha+1) = \int_0^\infty t^{\alpha}e^{-t}\,dt \tag{10.2}$$

Using integration by parts, let:

$$u = t^\alpha, \quad dv = e^{-t}dt \tag{10.3}$$

so that

$$du = \alpha t^{\alpha-1}dt, \quad v = -e^{-t} \tag{10.4}$$

Applying integration by parts $\int u\,dv = uv - \int v\,du$:

$$\Gamma(\alpha+1) = -t^\alpha e^{-t}\Big|_0^\infty + \alpha\int_0^\infty t^{\alpha-1}e^{-t}dt \tag{10.5}$$

The boundary term evaluates to zero because:

- As $t \to \infty$, the exponential decay dominates the polynomial growth, so $t^\alpha e^{-t} \to 0$.
- At $t = 0$, $t^\alpha e^{-t} = 0$.

Therefore,

$$\begin{aligned}\Gamma(\alpha+1) &= \alpha\int_0^\infty t^{\alpha-1}e^{-t}dt & (10.6)\\ &= \alpha\Gamma(\alpha) & (10.7)\end{aligned}$$

This recursive relation generalizes the factorial function because for natural numbers n,

$$\Gamma(n) = (n-1)! \tag{10.8}$$

which follows by repeatedly applying the recursive formula starting from $\Gamma(1) = 1$.

Thus, the Gamma function extends the factorial to real (and complex) arguments.

(b) Determine the moment of order t of a Be(α, β) distribution.

Let $X \sim \text{Be}(\alpha, \beta)$ be a Beta-distributed random variable with parameters $\alpha > 0$ and $\beta > 0$. The probability density function (PDF) of X is:

$$f_X(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad x \in (0,1) \tag{10.9}$$

where $B(\alpha, \beta)$ is the Beta function.

The t-th moment of X is defined as:

$$\mathbb{E}[X^t] = \int_0^1 x^t f_X(x)\, dx \tag{10.10}$$

Substituting the PDF:

$$\mathbb{E}[X^t] = \frac{1}{B(\alpha, \beta)} \int_0^1 x^{t+\alpha-1}(1-x)^{\beta-1}\, dx \tag{10.11}$$

Recognizing the integral as the Beta function $B(t+\alpha, \beta)$, we get:

$$\mathbb{E}[X^t] = \frac{B(t+\alpha, \beta)}{B(\alpha, \beta)} \tag{10.12}$$

Using the Gamma function representation of the Beta function:

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \tag{10.13}$$

Eq. 10.12 can be rewritten as:

$$\begin{aligned}
\mathbb{E}[X^t] &= \frac{\Gamma(t+\alpha)\Gamma(\beta)}{\Gamma(t+\alpha+\beta)} \cdot \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} && (10.14)\\
&= \frac{\Gamma(t+\alpha)\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(t+\alpha+\beta)} && (10.15)\\
&= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot \frac{\Gamma(t+\alpha)\Gamma(\beta)}{\Gamma(t+\alpha+\beta)} && (10.16)\\
&= \frac{B(t+\alpha, \beta)}{B(\alpha, \beta)} && (10.17)
\end{aligned}$$

(c) Use this formula to obtain the expected value $\mathbb{E}[X]$ and variance $\mathbf{Var}(X)$ for $X \sim \mathbf{Be}(\alpha, \beta)$.

Using the moment formula derived in part (b), the t-th moment of a Beta-distributed variable $X \sim \text{Be}(\alpha, \beta)$ is:

$$\mathbb{E}[X^t] = \frac{B(t+\alpha, \beta)}{B(\alpha, \beta)} \tag{10.18}$$

Expected value $\mathbb{E}[X]$

For the first moment ($t = 1$):

$$\mathbb{E}[X] = \frac{B(\alpha+1, \beta)}{B(\alpha, \beta)} \tag{10.19}$$

Using the Beta function identity in terms of Gamma functions:

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \tag{10.20}$$

we get:

$$\begin{aligned}
\mathbb{E}[X] &= \frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta+1)} \cdot \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} && (10.21)\\
&= \frac{\Gamma(\alpha+1)\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\alpha+\beta+1)} && (10.22)
\end{aligned}$$

Using the recursive Gamma property $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$, we simplify:

$$\mathbb{E}[X] = \frac{\alpha}{\alpha+\beta} \tag{10.23}$$

Variance $\mathbf{Var}(X)$

The variance is:

$$\text{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \tag{10.24}$$

Using the moment formula for $t = 2$:

$$\begin{aligned}
\mathbb{E}[X^2] = \frac{B(\alpha+2, \beta)}{B(\alpha, \beta)} &= \frac{\Gamma(\alpha+2)\Gamma(\beta)}{\Gamma(\alpha+\beta+2)} \cdot \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} && (10.25)\\
&= \frac{\Gamma(\alpha+2)\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\alpha+\beta+2)} && (10.26)
\end{aligned}$$

Using $\Gamma(\alpha+2) = (\alpha+1)\alpha\Gamma(\alpha)$, we get:

$$\mathbb{E}[X^2] = \frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)} \tag{10.27}$$

Therefore:

$$\begin{aligned}
\text{Var}(X) &= \frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)} - \left(\frac{\alpha}{\alpha+\beta}\right)^2 && (10.28)\\
&= \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} && (10.29)
\end{aligned}$$

(d) Use the Monte Carlo approach to estimate the shape parameters of a Be$(9, 3)$ distribution (set $n = 15{,}000$).

Using the Monte Carlo approach, we generated $n = 15{,}000$ samples from a Be$(9, 3)$ distribution. From these samples, we computed the sample mean $\bar{X}$ and sample variance S^2, which serve as empirical estimates of the theoretical moments.

Recall the theoretical formulas for the mean and variance of a Beta-distributed random variable $X \sim \text{Be}(\alpha, \beta)$:

$$\mathbb{E}[X] = \frac{\alpha}{\alpha + \beta} \tag{10.30}$$

$$\text{Var}(X) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \tag{10.31}$$

By matching these theoretical moments to the sample moments, $\bar{X} \approx \mathbb{E}[X]$, $S^2 \approx \text{Var}(X)$, we invert the formulas to estimate the shape parameters:

$$\hat{\alpha} = \bar{X}\left(\frac{\bar{X}(1 - \bar{X})}{S^2} - 1\right) \tag{10.32}$$

$$\hat{\beta} = (1 - \bar{X})\left(\frac{\bar{X}(1 - \bar{X})}{S^2} - 1\right) \tag{10.33}$$

Applying this method to the simulated data, we obtained:

$$\hat{\alpha} \approx 9.0754, \quad \hat{\beta} \approx 3.0359$$

which closely approximate the true parameters $\alpha = 9$ and $\beta = 3$.

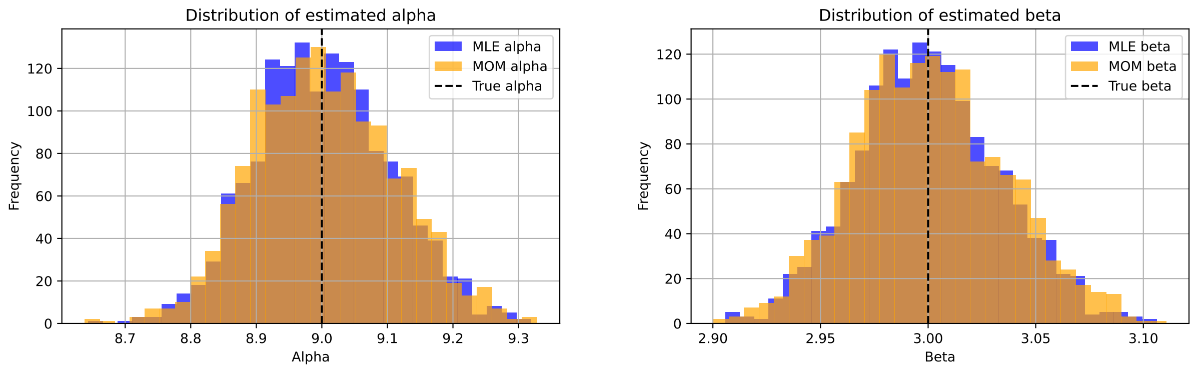
11 Finally, let us compare both estimation methods (MLE and MOM). To this end, set the number of simulations equal to $Nsim = 1500$.

(a) Use the previous implementation for the MOM and use the function `ebeta` contained in the package `EnvStats` for the MLE estimator for the beta distribution to create histograms illustrating the distribution of each of the shape parameters.

We conducted $N_{sim} = 1500$ independent simulations, each generating $n = 15{,}000$ samples from a Be$(9, 3)$ distribution. For each simulated dataset, we estimated the shape parameters α and β using two methods:

- Maximum likelihood estimation (MLE): Using the `beta.fit` function from the `scipy.stats` library, which estimates parameters by maximizing the likelihood function.
- Method of moments (MOM): Based on equating the theoretical moments of the Beta distribution to the sample moments.

The histograms (Figure 9) illustrate the empirical distribution of the estimated parameters across the 1500 simulations.



(a) Estimated α values (b) Estimated β values

Figure 9: Histograms of estimated Beta distribution parameters across 1500 simulations.

- Both the MLE and MOM estimators for α and β are approximately unbiased, with distributions centered near the true parameter values.
- The spread of the histograms indicates the variability of the estimators. Visual inspection suggests that the MLE estimators have slightly less variability than the MOM estimators.
- The overlap in histograms indicates strong agreement between the two estimation methods for large samples.

(b) Evaluate the efficiency of each method in terms of bias and variance. Compare the results.

Using the simulation results from 1,500 independent datasets of size n = 15,000 drawn from a Be$(9, 3)$ distribution, we evaluated the bias and variance of the shape parameter estimators for both Maximum Likelihood Estimation (MLE) and Method of Moments (MOM).

The results are summarized in Table 3:

Table 3: Estimated bias and variance of the Beta distribution shape parameters over 1500 simulations.

Parameter	**Bias**	**Variance**
α (MLE)	0.002573	0.010503
α (MOM)	0.003619	0.011518
β (MLE)	0.000461	0.001087
β (MOM)	0.000858	0.001211

- Both MLE and MOM estimators exhibit very small biases, indicating nearly unbiased estimation of the parameters.
- The MLE estimators have slightly lower variance for both α and β, suggesting higher efficiency compared to the MOM estimators.
- Overall, both methods provide accurate and reliable parameter estimates for large samples, with MLE being marginally preferable in terms of variability.

(c) Illustrate the asymptotic normality of the obtained MLE estimators.

The asymptotic normality property is fundamental in statistics and describes how, as the sample size n grows large, the distribution of the maximum likelihood estimator (MLE) $\hat{\theta}$ approaches a normal distribution centered around the true parameter θ_{true}.

Mathematically, this is expressed as:

$$\sqrt{n}(\hat{\theta} - \theta_{\text{true}}) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2_{\hat{\theta},\text{true}}\right),$$

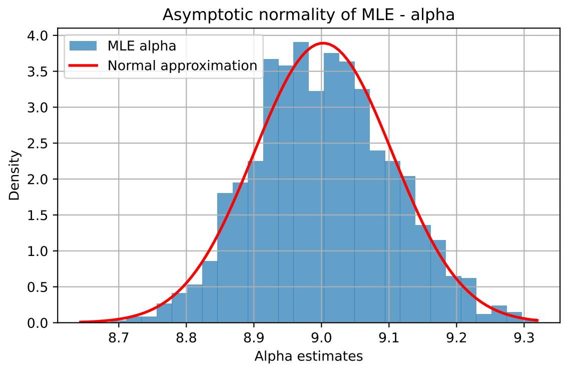
where $\xrightarrow{d}$ denotes convergence in distribution, and $\sigma^2_{\hat{\theta},\text{true}}$ is the asymptotic variance of the estimator.

In this context:

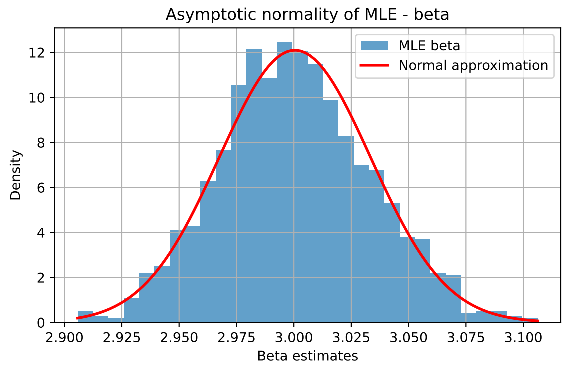
- $\sqrt{n}$ represents the rate at which the distribution converges.
- $\hat{\theta}$ is the estimator obtained from the data.
- θ_{true} is the true (unknown) parameter value.

Using $N_{sim} = 1500$ simulations of sample size $n = 15,000$ from a Be$(9, 3)$ distribution, we computed MLE estimates for the shape parameters α and β.

Figure 10 and Figure 11 illustrate that the distribution of the scaled estimation errors $\sqrt{n}(\hat{\theta} - \theta_{\text{true}})$ approximates a normal distribution centered around zero, which confirms the asymptotic normality of the MLE estimators.



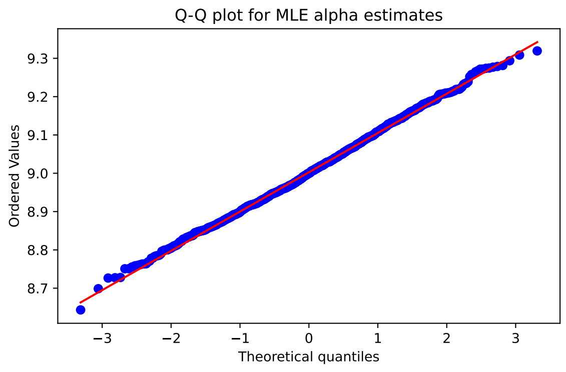
(a) Histogram of MLE α estimates with normal overlay



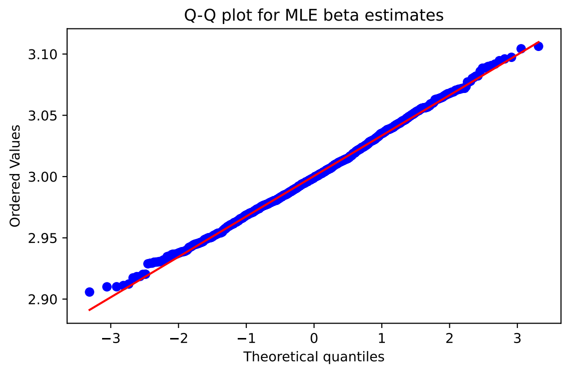
(b) Histogram of MLE β estimates with normal overlay

Figure 10: Histograms illustrating the approximate normality of MLE estimators for Beta parameters α and β.

The empirical distribution of the MLE estimates for α and β closely resemble normal distributions (red curves) with means and variances matching the sample estimates.



(a) Q-Q plot for MLE α estimates



(b) Q-Q plot for MLE β estimates

Figure 11: Q-Q plots comparing empirical quantiles of MLE estimators to theoretical normal quantiles, showing good agreement.

The quantile-quantile plots compare the quantiles of the MLE estimates to those of a theoretical normal distribution. The points fall approximately along the diagonal line, indicating that the empirical distributions are well-approximated by normal distributions.

These visualizations clearly demonstrate that, as the number of simulations increases, the distribution of estimation errors tends toward a normal distribution centered at zero, consistent with the asymptotic normality property of MLE estimators.

A Appendix

A.1 Question 2

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

def plot_beta_distribution(alpha_val, beta_val, filename=None):
    """
    Plot a Beta distribution with given alpha and beta parameters, then save as a PDF.

    Parameters:
    - alpha_val (float): Shape parameter alpha of the Beta distribution.
    - beta_val (float): Shape parameter beta of the Beta distribution.
    - filename (str): Name of the file to export the plot (should end with .pdf).
    """
    x = np.linspace(0, 1, 100)
    y = beta.pdf(x, alpha_val, beta_val)

    plt.figure(figsize=(4, 3))
    plt.plot(x, y, label=f"Beta({alpha_val}, {beta_val})")
    plt.title(f"Be({alpha_val}, {beta_val})")
    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.grid(True)
    plt.tight_layout()

    if filename is None:
        os.makedirs("figures", exist_ok=True)
        filename = f"figures/q1-beta_distr-alpha_{alpha_val}-beta_{beta_val}.pdf"

    plt.savefig(filename, format='pdf')
    plt.show()

# Symmetric
plot_beta_distribution(0.5, 0.5) # U-shaped
plot_beta_distribution(1,1) # Uniform distribution
plot_beta_distribution(2, 2) # Bell-shaped
plot_beta_distribution(5, 5) # More peaked

# Left-skewed
plot_beta_distribution(1,3) # Left-skewed
plot_beta_distribution(5,2) # More left-skewed

# Right-skewed
plot_beta_distribution(3,1) # Right-skewed
plot_beta_distribution(2,5) # More right-skewed

# Edge case
plot_beta_distribution(10000,10000) # Very peaked around 0.5
plot_beta_distribution(100000,100000) # Very peaked around 0.5
```

A.2 Question 3

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

# Parameters
a = 3
b = 9
n_samples = 15000

# Generate uniform random numbers
u = np.random.uniform(0, 1, size=n_samples)

# Apply the inverse CDF (quantile function)
samples = beta.ppf(u, a, b)

# Plot the histogram of the generated samples
plt.figure(figsize=(8, 4))
plt.hist(samples, bins=50, density=True, color='skyblue', edgecolor='black',
    alpha=0.7, label='Simulated (Inverse transform)')

# Plot the true Beta(3,9) density for comparison
x = np.linspace(0, 1, 1000)
plt.plot(x, beta.pdf(x, a, b), 'r-', lw=2, label='True PDF')

plt.title('Beta(3, 9) - Inverse transform sampling')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q2-beta_inverse_transform_sampling.pdf")

plt.show()
```

A.3 Question 4

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

# Parameters
a = 3
b = 9
n_samples = 15000

# Generate U_ij ~ Uniform(0, 1)
U = np.random.uniform(0, 1, size=(n_samples, a + b))
log_U = -np.log(U)

# Compute sum of first 'a' logs and sum of all a + b logs for each row
Sa = np.sum(log_U[:, :a], axis=1)
Sab = np.sum(log_U, axis=1)

# Apply the transformation
samples = Sa / Sab

# Plot histogram of generated samples
plt.figure(figsize=(8, 4))
plt.hist(samples, bins=50, density=True, color='lightgreen', edgecolor='black',
    alpha=0.7, label='Simulated (Transformation)')

# Plot the true Beta(3,9) density
x = np.linspace(0, 1, 1000)
plt.plot(x, beta.pdf(x, a, b), 'r-', lw=2, label='True PDF')

plt.title('Beta(3, 9) - Transformation sampling')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q3-beta_transformation_sampling.pdf")

plt.show()
```

A.4 Question 5

(b)

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gamma, beta

# Parameters
a = 3
b = 9
n_samples = 15000

# Generate Y1 ~ Gamma(alpha, 1) and Y2 ~ Gamma(beta, 1)
Y1 = np.random.gamma(shape=a, scale=1.0, size=n_samples)
Y2 = np.random.gamma(shape=b, scale=1.0, size=n_samples)

# Compute X = Y1 / (Y1 + Y2)
samples = Y1 / (Y1 + Y2)

# Plot histogram of generated samples
plt.figure(figsize=(8, 4))
plt.hist(samples, bins=50, density=True, color='lightcoral', edgecolor='black',
    alpha=0.7, label='Simulated (Gamma ratio)')

# Plot the true Beta(3, 9) PDF
x = np.linspace(0, 1, 1000)
plt.plot(x, beta.pdf(x, a, b), 'r-', lw=2, label='True PDF')

plt.title('Beta(3, 9) - Gamma ratio sampling')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q5-beta_gamma_ratio_sampling.pdf")

plt.show()
```

A.5 Question 6

```python
import os
import time
import numpy as np
import pandas as pd
from scipy.stats import beta, kstest
import matplotlib.pyplot as plt

# Parameters
a = 3
b = 9
sample_sizes = [10, 100, 1000, 10000, 100000, 1000000]
n_runtime = 15000
n_experiments = 200

# Sampling methods
def method_uniform_log(n, alpha_val, beta_val):
    U = np.random.uniform(0, 1, size=(n, alpha_val + beta_val))
    log_U = -np.log(U)
    Sa = np.sum(log_U[:, :alpha_val], axis=1)
    Sab = np.sum(log_U, axis=1)
    return Sa / Sab

def method_gamma_ratio(n, alpha_val, beta_val):
    Y1 = np.random.gamma(alpha_val, 1, size=n)
    Y2 = np.random.gamma(beta_val, 1, size=n)
    return Y1 / (Y1 + Y2)

def method_builtin_beta(n, alpha_val, beta_val):
    return beta.rvs(alpha_val, beta_val, size=n)

# Containers for averaging
runtime_results = {'uniform_log': [], 'gamma_ratio': [], 'builtin': []}
ks_results = {'uniform_log': {n: [] for n in sample_sizes},
              'gamma_ratio': {n: [] for n in sample_sizes},
              'builtin': {n: [] for n in sample_sizes}}

# Measure runtime (n = 15000)
for _ in range(n_experiments):
    start = time.time()
    method_uniform_log(n_runtime, a, b)
    runtime_results['uniform_log'].append(time.time() - start)

    start = time.time()
    method_gamma_ratio(n_runtime, a, b)
    runtime_results['gamma_ratio'].append(time.time() - start)

    start = time.time()
    method_builtin_beta(n_runtime, a, b)
    runtime_results['builtin'].append(time.time() - start)

# Measure K-S statistic for all sample sizes
for n in sample_sizes:
    for i in range(n_experiments):
        samples = method_uniform_log(n, a, b)
        ks_results['uniform_log'][n].append(kstest(samples, 'beta', args=(a, b))
    [0])

        samples = method_gamma_ratio(n, a, b)
        ks_results['gamma_ratio'][n].append(kstest(samples, 'beta', args=(a, b))
    [0])

        samples = method_builtin_beta(n, a, b)
        ks_results['builtin'][n].append(kstest(samples, 'beta', args=(a, b))[0])

# Average results
avg_runtime = {method: np.mean(times) for method, times in runtime_results.items
    ()}
avg_ks = {method: [np.mean(ks_results[method][n]) for n in sample_sizes] for
    method in ks_results.keys()}

# Create DataFrames
df_runtime = pd.DataFrame(avg_runtime, index=[n_runtime])
df_runtime.index.name = 'Sample Size (n)'

df_ks = pd.DataFrame(avg_ks, index=sample_sizes)
df_ks.index.name = 'Sample Size (n)'

print("Average runtime results (seconds) for n=15,000:")
print(df_runtime)

print("\nAverage K-S statistic results (lower is better):")
print(df_ks)

# plot K-S statistic
plt.figure(figsize=(8, 3))
for method, ks_values in avg_ks.items():
    plt.plot(sample_sizes, ks_values, marker='o', label=method)

plt.xlabel('Sample size (n) (log scale)')
plt.ylabel('Average K-S statistic (log scale)')
plt.title('Average K-S statistic vs sample size for Beta sampling methods')
plt.xscale('log')
plt.yscale('log')
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q6-beta_sampling_methods_ks_statistic.pdf")

plt.show()
```

A.6 Question 7

(b)

```python
from scipy.stats import beta

# Parameters for Beta(3.3, 9.5)
a = 3.3
b = 9.5

# Mode formula for alpha > 1, beta > 1
def beta_mode(alpha, beta):
    return (alpha - 1) / (alpha + beta - 2)

# Compute the analytical limiting constant C using the mode
mode = beta_mode(a, b)
C_analytical = beta.pdf(mode, a, b)

print(f"Analytical limiting constant C: {C_analytical}, Mode: {mode}")
```

(c)

```python
import numpy as np
from scipy.stats import beta

# Parameters
a = 3.3
b = 9.5

# Compute limiting constant C using the Beta mode
def beta_mode(alpha, beta):
    return (alpha - 1) / (alpha + beta - 2)

C = beta.pdf(beta_mode(a, b), a, b)
n_trials = 15000

# Accept-Reject simulation with fixed number of proposals
accepted = []
rejected = []

for _ in range(n_trials):
    x = np.random.uniform(0, 1)
    u = np.random.uniform(0, 1)
    fx = beta.pdf(x, a, b)
    if u < fx / C:
        accepted.append(x)
    else:
        rejected.append(x)

# Convert to arrays
accepted = np.array(accepted)
rejected = np.array(rejected)
```

(d)

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

# Output acceptance rate
acceptance_rate = len(accepted) / (len(accepted) + len(rejected))

```

```python
print(f"Acceptance rate: {acceptance_rate}")

# Prepare plot values
x_vals = np.linspace(0, 1, 1000)
true_beta_pdf = beta.pdf(x_vals, a, b)
uniform_pdf = np.ones_like(x_vals)

# Plotting
plt.figure(figsize=(8, 4))
plt.hist(rejected, bins=100, density=True, alpha=0.5, color='red', label='
    Rejected samples')
plt.hist(accepted, bins=100, density=True, alpha=0.5, color='green', label='
    Accepted samples')
plt.plot(x_vals, true_beta_pdf, 'k-', lw=2, label='True Beta PDF')
plt.plot(x_vals, uniform_pdf, 'b--', lw=2, label='Instrumental Uniform PDF')
plt.title(r'Accept-reject simulation: Be(3.3, 9.5)')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q7d-beta_accept_reject.pdf")

plt.show()
```

(e)

```python
# Recompute theoretical acceptance probability
theoretical_acceptance_rate = 1 / C_analytical

# Compare to empirical
comparison = {
    "Theoretical acceptance rate (1/C)": theoretical_acceptance_rate,
    "Empirical acceptance rate": acceptance_rate,
    "Absolute difference": abs(theoretical_acceptance_rate - acceptance_rate)
}

print(comparison)
```

(f)

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta

# Parameters
a = 3.3
b = 9.5
target_n = 15000

accepted_f = []
total_trials_f = 0

while len(accepted_f) < target_n:
    x = np.random.uniform(0, 1)
    u = np.random.uniform(0, 1)
    fx = beta.pdf(x, a, b)
    if u < fx / C:
        accepted_f.append(x)
    total_trials_f += 1

# Convert to NumPy array
accepted_f = np.array(accepted_f)

# Compute updated acceptance rate
acceptance_rate_f = target_n / total_trials_f

print(f"Final acceptance rate (exact 15,000 accepts): {acceptance_rate_f}")

# Plot histogram of accepted samples vs. true PDF
plt.figure(figsize=(8, 4))
plt.hist(accepted_f, bins=100, density=True, alpha=0.6, label='Simulated samples
    ', color='green')
plt.plot(x_vals, true_beta_pdf, 'k-', lw=2, label='True Beta(3.3, 9.5) PDF')
plt.title('Simulated vs. true distribution (exact 15,000 accepts)')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q7f-beta_fixed_15k_accepts.pdf")

plt.show()
```

A.7 Question 8

(b)

```python
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta
from scipy.special import beta as beta_func

# Parameters
alpha_target = 3.3
beta_target = 9.5
a_instrument = 3
b_instrument = 9

# Compute limiting constant C for f/g
def compute_beta_ratio_max(alpha, beta, a, b):
    B_ab = beta_func(a, b)
    B_alpha_beta = beta_func(alpha, beta)
    x_mode = (alpha - a) / ((alpha - a) + (beta - b))
    ratio = (B_ab / B_alpha_beta) * \
            (x_mode ** (alpha - a)) * ((1 - x_mode) ** (beta - b))
    return ratio, x_mode

C_beta, x_max = compute_beta_ratio_max(alpha_target, beta_target, a_instrument,
    b_instrument)

# Accept-Reject simulation using Beta(3, 9) as instrumental distribution
n_target = 15000
accepted_beta = []
rejected_beta = []
total_trials_beta = 0

while len(accepted_beta) < n_target:
    x = np.random.beta(a_instrument, b_instrument)
    u = np.random.uniform(0, 1)
    fx = beta.pdf(x, alpha_target, beta_target)
    gx = beta.pdf(x, a_instrument, b_instrument)
    if u < fx / (C_beta * gx):
        accepted_beta.append(x)
    else:
        rejected_beta.append(x)
    total_trials_beta += 1

acceptance_rate_beta = len(accepted_beta) / total_trials_beta

print(f"Acceptance rate: {acceptance_rate_beta}, Limiting constant C: {C_beta},
    Mode: {x_max}")

# Prepare plot
x_vals = np.linspace(0, 1, 1000)
true_pdf = beta.pdf(x_vals, alpha_target, beta_target)
instrument_pdf = beta.pdf(x_vals, a_instrument, b_instrument)

plt.figure(figsize=(8, 6.5))
plt.hist(rejected_beta, bins=100, density=True, alpha=0.4, color='red', label='
    Rejected samples')
plt.hist(accepted_beta, bins=100, density=True, alpha=0.6, color='green', label=
    'Accepted samples')
plt.plot(x_vals, true_pdf, 'k-', lw=2, label='True Beta(3.3, 9.5) PDF')
plt.plot(x_vals, instrument_pdf, 'b--', lw=2, label='Instrumental beta(3, 9) PDF
    ')
plt.title('Accept-Reject using Beta(3, 9) instrumental')
plt.xlabel('x')
plt.ylabel('Density')
plt.legend()
plt.yticks(ticks=np.arange(0, plt.ylim()[1] + 0.5, 0.5))
plt.grid(True, which='major', axis='both')
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q8b-beta_instrumental.pdf")

plt.show()
```

A.8 Question 10

(d)

```python
import numpy as np

# Parameters
a = 9
b = 3
n_samples = 15000

# Generate samples from Beta(9, 3)
samples = np.random.beta(a, b, size=n_samples)

# Calculate sample mean and variance
sample_mean = np.mean(samples)
sample_var = np.var(samples, ddof=1)  # unbiased estimator

# Use method of moments formulas to estimate alpha and beta
common_factor = (sample_mean * (1 - sample_mean) / sample_var) - 1
alpha_est = sample_mean * common_factor
beta_est = (1 - sample_mean) * common_factor

print(f"Estimated alpha: {alpha_est}")
print(f"Estimated beta: {beta_est}")
```

A.9 Question 11

(a)

```python
import numpy as np

# Convert lists to numpy arrays for easier computations
mle_alpha_arr = np.array(mle_alpha)
mle_beta_arr = np.array(mle_beta)
mom_alpha_arr = np.array(mom_alpha)
mom_beta_arr = np.array(mom_beta)

# True parameter values
alpha_true = 9
beta_true = 3

# Calculate bias and variance for alpha
bias_mle_alpha = np.mean(mle_alpha_arr) - alpha_true
var_mle_alpha = np.var(mle_alpha_arr, ddof=1)

bias_mom_alpha = np.mean(mom_alpha_arr) - alpha_true
var_mom_alpha = np.var(mom_alpha_arr, ddof=1)

# Calculate bias and variance for beta
bias_mle_beta = np.mean(mle_beta_arr) - beta_true
var_mle_beta = np.var(mle_beta_arr, ddof=1)

bias_mom_beta = np.mean(mom_beta_arr) - beta_true
var_mom_beta = np.var(mom_beta_arr, ddof=1)

# Print results
print("Alpha parameter:")
print(f"MLE Bias: {bias_mle_alpha:.6f}, Variance: {var_mle_alpha:.6f}")
print(f"MOM Bias: {bias_mom_alpha:.6f}, Variance: {var_mom_alpha:.6f}")

print("\nBeta parameter:")
print(f"MLE Bias: {bias_mle_beta:.6f}, Variance: {var_mle_beta:.6f}")
print(f"MOM Bias: {bias_mom_beta:.6f}, Variance: {var_mom_beta:.6f}")
```

(b)

```python
import numpy as np

# Convert lists to numpy arrays
mle_alpha_arr = np.array(mle_alpha)
mle_beta_arr = np.array(mle_beta)
mom_alpha_arr = np.array(mom_alpha)
mom_beta_arr = np.array(mom_beta)

# Parameters
alpha_true = 9
beta_true = 3

# Calculate bias and variance for alpha
bias_mle_alpha = np.mean(mle_alpha_arr) - alpha_true
var_mle_alpha = np.var(mle_alpha_arr, ddof=1)

bias_mom_alpha = np.mean(mom_alpha_arr) - alpha_true
var_mom_alpha = np.var(mom_alpha_arr, ddof=1)

# Calculate bias and variance for beta
bias_mle_beta = np.mean(mle_beta_arr) - beta_true
var_mle_beta = np.var(mle_beta_arr, ddof=1)

```

```python
bias_mom_beta = np.mean(mom_beta_arr) - beta_true
var_mom_beta = np.var(mom_beta_arr, ddof=1)

print("Alpha parameter:")
print(f"MLE Bias: {bias_mle_alpha:.6f}, Variance: {var_mle_alpha:.6f}")
print(f"MOM Bias: {bias_mom_alpha:.6f}, Variance: {var_mom_alpha:.6f}")

print("\nBeta parameter:")
print(f"MLE Bias: {bias_mle_beta:.6f}, Variance: {var_mle_beta:.6f}")
print(f"MOM Bias: {bias_mom_beta:.6f}, Variance: {var_mom_beta:.6f}")
```

(c)

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

# Mean and std of MLE estimates
mle_alpha_mean = np.mean(mle_alpha_arr)
mle_alpha_std = np.std(mle_alpha_arr, ddof=1)
mle_beta_mean = np.mean(mle_beta_arr)
mle_beta_std = np.std(mle_beta_arr, ddof=1)

x_alpha = np.linspace(min(mle_alpha_arr), max(mle_alpha_arr), 100)
x_beta = np.linspace(min(mle_beta_arr), max(mle_beta_arr), 100)

# Normal PDFs for overlay
normal_pdf_alpha = stats.norm.pdf(x_alpha, mle_alpha_mean, mle_alpha_std)
normal_pdf_beta = stats.norm.pdf(x_beta, mle_beta_mean, mle_beta_std)

# Plot histogram + normal overlay for alpha
plt.figure(figsize=(6, 4))
plt.hist(mle_alpha_arr, bins=30, density=True, alpha=0.7, label='MLE alpha')
plt.plot(x_alpha, normal_pdf_alpha, 'r-', lw=2, label='Normal approximation')
plt.title('Asymptotic normality of MLE - alpha')
plt.xlabel('Alpha estimates')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q11c-alpha_mle_histogram.pdf")


plt.show()

# Plot histogram + normal overlay for beta
plt.figure(figsize=(6, 4))
plt.hist(mle_beta_arr, bins=30, density=True, alpha=0.7, label='MLE beta')
plt.plot(x_beta, normal_pdf_beta, 'r-', lw=2, label='Normal approximation')
plt.title('Asymptotic normality of MLE - beta')
plt.xlabel('Beta estimates')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q11c-beta_mle_histogram.pdf")

plt.show()

```

```
# Q-Q plot for alpha
plt.figure(figsize=(6, 4))
stats.probplot(mle_alpha_arr, dist="norm", plot=plt)
plt.title('Q-Q plot for MLE alpha estimates')
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q11c-alpha_mle_qqplot.pdf")

plt.show()

# Q-Q plot for beta
plt.figure(figsize=(6, 4))
stats.probplot(mle_beta_arr, dist="norm", plot=plt)
plt.title('Q-Q plot for MLE beta estimates')
plt.tight_layout()

os.makedirs("figures", exist_ok=True)
plt.savefig("figures/q11c-beta_mle_qqplot.pdf")

plt.show()
```