Chapter 12

Leaving conjugates behind: Markov Chain Monte Carlo

12.1 A fairground game

At a fairground a man advertises a gambling game that allows participants the chance to win a monetary prize, if they pay an entrance fee. The game sequence goes like this:

- You pay £X.
- The man flips a fair coin (i.e. with an equal chance of the coin landing heads or tails up),
- If the coin lands tails up, the game ends and you walk away with nothing.
- If the coin lands heads up, he flips the coin a further two times and you receive the total number of heads across these latter two flips, H. So if the coin lands heads up twice, you receive £2; if once, you receive £1; if zero, you receive £0.
- Your winnings are given by £(H - X).

Problem 12.1.1. Calculate the expected value of your winnings W if you participate, and hence determine the fair price of the game.

The expected value of winnings W is given by,

$$\mathbb{E}(W) = (1/2) \times 0 + (1/2) \times 2 \times (1/2) - X = (1/2) - X \tag{12.1}$$

So the fair price is just £0.50.

Problem 12.1.2. Create an R function that simulates a single run of the game, and use this to estimate the expected value of your winnings.

Hint: use R's `rbinom` and `ifelse` functions.

A function that does this is shown below,

```
fGame <- function(){
  Y <- rbinom(1, 1, 0.5)
  Z <- ifelse(Y == 0, 0, rbinom(1, 2, 0.5))
  return(Z)
}
```

which can then be run over 10,000 iterations, and its mean calculated

```
mean(sapply(seq(1, 10000, 1), function(i) fGame()))
```

which should roughly be 0.5.

Problem 12.1.3. Suppose that you pay £1 for each game, and start with £10 in your pocket. By using your previously-created function, or otherwise, determine the expected number of games you can play before going broke.

A function that implements this is,

```
fGamblersRuin <- function(Wealth, Price){
  numGames <- 0
  while(Wealth > 0){
    Win <- fGame() - Price
    Wealth <- Wealth + Win
    numGames <- numGames + 1
  }
  return(numGames)
}
```

Using the above function to simulate 10,000 runs,

```
mean(sapply(seq(1, 10000, 1), function(i) fGamblersRuin(10, 1)))
```

which should be close to 20. This makes sense intuitively, since the expected loss per game is £0.50, and so it takes 20 games for our initial wealth to erode!

Problem 12.1.4. Suppose you start with £10, and play the game 100 times (stopping only if your wealth is below the price of entry), each time paying £0.49. You want to insure against the risk of losing all your wealth. What is the fair price to pay for such an insurance scheme?

The fair price to pay is the probability that you go bust times the loss that causes you, i.e. £10. To determine this risk I use sampling, creating a function that determines the number of plays before you go broke.

```
fGamblersRuinN <- function(N, Wealth, Price){
  numGames <- 0
  for(i in 1:N){
    if(Wealth >= Price){
      Win <- fGame() - Price
      Wealth <- Wealth + Win
      numGames <- numGames + 1
    } else{
      break
    }
  }
  return(numGames)
}
```

which we then use to simulate 10,000 runs, and calculate the proportion of times that you lose all your wealth,

```
lData <- sapply(seq(1, 10000, 1), function(i) fGamblersRuinN(100, 10, 0.49))
mean(lData < 100)
```

which should be around 12-13%. So the fair amount to pay for such a scheme is about £1.20-1.30.

12.2 Independent sampling

An analysis results in a posterior with the following probability density function:

$$f(x) = \begin{cases} \frac{1}{1.33485}\frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}, & \text{if } x < 0.9735. \\ 0.186056, & \text{if } 0.9735 \leq x \leq 5. \\ 0, & \text{otherwise.} \end{cases} \tag{12.2}$$

Problem 12.2.1. Verify that this is a valid PDF (hint: see R's numerical integration function).

This is straightforwardly done by integrating the function over the range,

```
fPDF <- function(X){
  y <- ifelse(X < 0.9735, (1 / 1.33485) * (1 / sqrt(2 * pi)) * exp(-X ^ 2 / 2),
                          ifelse(X <= 5, 0.186056, 0))
  return(y)
}
integrate(fPDF,0,8)
```

Problem 12.2.2. Using *independent* sampling estimate the mean and variance of this distribution.

There are a large number of methods here, of which I describe two (importance sampling is discussed below). The first is **rejection sampling** where we generate two (pseudo-)random continuous points: $x \in (0,5)$ and $y \in (0, \frac{1}{1.335\sqrt{2\pi}}$. We *accept* the point as a sample from our distribution iff $y \leq f(x)$.

```
fReject <- function(N){
  count <- 1
  lSamples <- vector(length=N)
  while(count <= N){
    X <- runif(1, 0, 5)
    Y <- runif(1, 0, (1 / 1.335) * (1 / sqrt(2 * pi)))
    if(Y < fPDF(X)){
      lSamples[count] <- X
      count <- count + 1
    }
  }
  return(lSamples)
}
mean(fReject(10000), 100)
var(fReject(10000))
```

yields a mean of around 2.35, and a variance of about 2.24.

The second is **inverse transform sampling**. To do this we need first to find an approximate CDF function, by creating a function that integrates from 0 to a given point x. We can then generate $(CDF[x], x)$ for $x \in (0,5)$, which we use to create an interpolating function that represents the inverse CDF. Then use R's uniform random number generator on (0,1) to generate a series of 'CDF' samples, to which the inverse CDF is applied to give actual samples,

```
fIntegrator <- function(X){
  return(integrate(fPDF, 0, X)[[1]])
}

lCDF <- sapply(seq(0, 5, 0.1), fIntegrator)
fICDF <- approxfun(lCDF, seq(0, 5, 0.1))

fInverseTransform <- function(N){
  lCDF <- runif(N, 0, 1)
  return(sapply(lCDF, fICDF))
}
mean(fInverseTransform(100000))
var(fInverseTransform(100000))
```

Problem 12.2.3. Construct uncertainty intervals around your estimates of the mean.

This needn't be done with too much precision. Essentially all you need to do is repeat the above process a reasonable number of times, and examine the quantiles of this distribution.

Problem 12.2.4. Verify your previous answer by calculating the mean and variance of this distribution.

This can be done using R's numerical integration functions,

```
aMean <- integrate(function(x) x * fPDF(x), 0, 5)[[1]]
aVar <- integrate(function(x) x ^ 2 * fPDF(x), 0, 5)[[1]] - aMean ^ 2
```

which yields a mean of about 2.35 and a variance of 2.24.

Problem 12.2.5. On the basis of the equation:

$$\begin{aligned} \mathrm{E(X)} &= \int x f(x) \mathrm{d}x \\ &= \int x \frac{f(x)}{g(x)} g(x) \mathrm{d}x \end{aligned}$$

provide another way to estimate the mean.

If we can generate independent samples $x \sim g(x)$, then we can estimate the mean using:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \frac{f(x_i)}{g(x_i)} \tag{12.3}$$

Since we know that:

$$\begin{aligned} \mathbb{E}_g(\hat{\mu}) &= \mathrm{E_g}\left(\mathrm{x}\frac{\mathrm{f(x)}}{\mathrm{g(x)}}\right) \\ &= \int x \frac{f(x)}{g(x)} g(x) \mathrm{d}x \\ &= \int x f(x) \mathrm{d}x \\ &= \mathrm{E_f(X)} = \mu \end{aligned}$$

In this case choosing $g(x)$ to be a continuous uniform distribution over (0,5) is a reasonable first choice.

Problem 12.2.6. Using the above method where g is the continuous uniform distribution between 0 and 5 find an estimate of the mean.

This can be done with the following code,

```
fImportance <- function(N){
  lX <- runif(N, 0, 5)
  lF <- sapply(lX, fPDF)
  lG <- rep(1 / 5, N)
  lRatio <- lX * lF / lG
  mean(lRatio)
}

fImportance(10000)
```

Problem 12.2.7. How should we choose $g(x)$ to yield estimators with the lowest variance? (Difficult.)

Essentially we want an estimator with a low variance:

$$\begin{aligned} Var_g(\hat{\mu}) &= \frac{1}{n^2}\sum_{i=1}^{n} Var_g\left[x_i \frac{f(x_i)}{g(x_i)}\right] \\ &= \frac{1}{n}\left[\int \frac{x^2 f(x)^2}{g(x)^2} g(x)\mathrm{d}x - \mu^2\right] \\ &= \frac{1}{n}\left[\int \frac{x^2 f(x)^2}{g(x)}\mathrm{d}x - \mu^2\right] \\ &= \frac{1}{n}\int \frac{(xf(x) - g(x)\mu)^2}{g(x)}\mathrm{d}x \end{aligned}$$

To minimise the above expression we should choose $g(x) = \frac{xf(x)}{\mu}$, or more generally $g(x) \propto xf(x)$. (For clarity we obtained the bottom line of the above because of the following trick:)

$$\begin{aligned} \int \frac{(xf(x) - g(x)\mu)^2}{g(x)}\mathrm{d}x &= \int \frac{x^2 f(x)^2 - 2xf(x)g(x)\mu + g(x)^2\mu^2}{g(x)}\mathrm{d}x \\ &= \int \frac{x^2 f(x)^2}{g(x)}\mathrm{d}x - 2\mu\int xf(x)\mathrm{d}x + \mu^2\int g(x)\mathrm{d}x \\ &= \int \frac{x^2 f(x)^2}{g(x)}\mathrm{d}x - \mu^2 \end{aligned}$$

In this example this means that perhaps using a triangular distribution would yield the lowest variance, although this isn't easy to sample from.

12.3 Integration by sampling

Calculate the following integrals by sampling.

Problem 12.3.1.

$$\int_{-\infty}^{\infty} \frac{x^6}{\sqrt{2\pi}} \times exp(-\frac{x^2}{2})\mathrm{d}x \tag{12.4}$$

This is equivalent to calculating $\mathbb{E}(X^6)$ from a standard normal. So all you do is sample from a standard normal and raise each sample to the power 6. If you then take the mean of these you get an estimator for the above. The actual answer here is 15 (Figure 12.1).

Problem 12.3.2.

$$\int_{1}^{\infty} \frac{x^3}{\sqrt{2\pi}} \times exp(-\frac{x^2}{2})\mathrm{d}x \tag{12.5}$$

I can generate samples from a truncated normal using Mathematica. Suspect that the same could be done in R or Matlab. Alternatively, just use rejection sampling! The actual answer here is about 47 (Figure 12.1).

Problem 12.3.3.

$$\int_{1}^{\infty} \frac{x^6}{\sqrt{2\pi}} \times exp(-\frac{x^2-4x}{2})\mathrm{d}x \tag{12.6}$$

The trick here is to notice that we can rewrite the above as:

$$\frac{x^6}{\sqrt{2\pi}}exp(-\frac{x^2-4x}{2}) = x^6exp(-2x) \times \frac{1}{\sqrt{2\pi}}exp(-\frac{x^2}{2}) \tag{12.7}$$

So we are trying to determine $\mathbb{E}\left(x^6exp(-2x)\right)$ for a truncated normal. The method then proceeds as above, with an answer of about 0.617 (Figure 12.1).

Problem 12.3.4.

$$\int_{1}^{10} x^6 \frac{e^{-\frac{x^4}{2}}}{\sqrt{2\pi}}\mathrm{d}x \tag{12.8}$$

This is a bit of a trick. It looks like we could use the normal distribution here, but you can't easily due to the x^4 in the exponent. A better way to approach this is to sample from a continuous uniform distribution on (1,10), and then take each sample x and evaluate $9\frac{e^{-\frac{x^4}{2}}x^6}{\sqrt{2\pi}}$. **Note:** we need a 9 there to account for the fact that the continuous uniform density is equal to $\frac{1}{9}$! If we then take the mean of the transformed samples we get the result of about 0.2665 (Figure 12.1).

Problem 12.3.5. What is the approximate sampling distribution in using independent sampling to evaluate integrals?

Using the (Lindberg-Lévy) central limit theorem (that applies for independent samples) we get:

$$\bar{X} \approx \mathcal{N}(\mu, \frac{\sigma}{\sqrt{n}}) \tag{12.9}$$

where n is the sample size, and $\sigma = \mathrm{E}[(\mathrm{i} - \mathrm{I})^2]$, where i is a sample estimate of the integral and I is the true value.

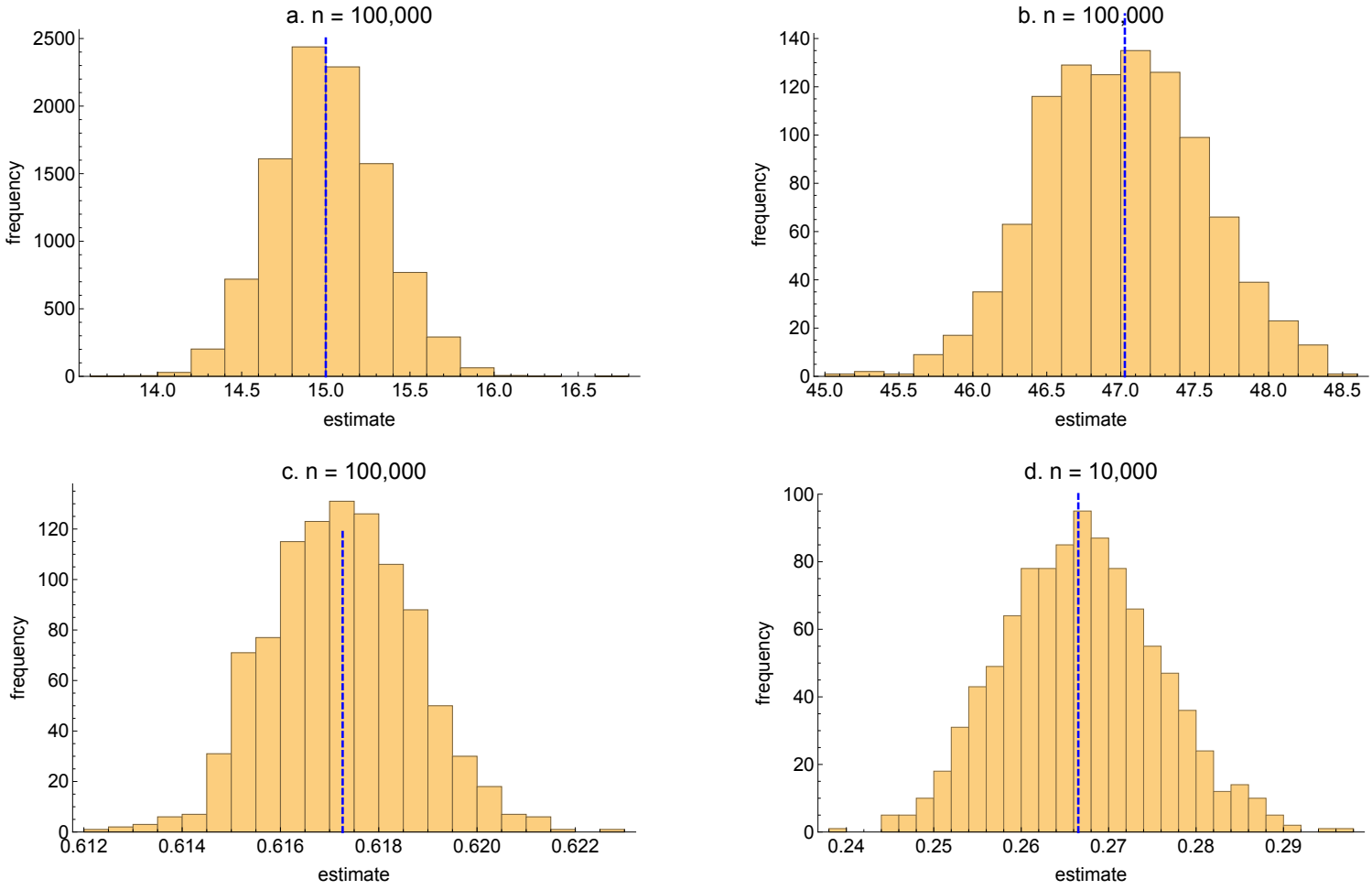

Figure 12.1: Sampling distributions for the estimators of the integrals in the text.

12.4 Markovian coin

Consider a type of coin for which the result of the next throw (heads or tails) can depend on the result of the current throw. In particular if a "heads" is thrown then the probability of obtaining a "heads" on the next throw is $(\frac{1}{2} + \epsilon)$; if instead a tails is thrown then the probability of obtaining a "tails" on the next throw is $(\frac{1}{2} + \epsilon)$. To start, we assume $0 \leq \epsilon \leq \frac{1}{2}$. The random variable $X = 0, 1$ if the coin lands "tails-up" or "heads-up" on a given throw.

Problem 12.4.1. Find the mean of the coin supposing it starts with probability $\frac{1}{2}$ on each side.

Considering the first throw after the initial one:

$$\begin{aligned}\mathbb{E}(X_1|X_0) &= Pr(X_1 = 1|X_0) \times 1 + Pr(X_1 = 0|X_0) \times 0 \\ &= Pr(X_1 = 1|X_0)\end{aligned}$$

By the law of iterated expectations:

$$\begin{aligned}\mathbb{E}(X_1) &= \mathbb{E}\left[\mathbb{E}(X_1|X_0)\right] \\ &= Pr(X_0 = 0) \times Pr(X_1 = 1|X_0 = 0) + Pr(X_0 = 1) \times Pr(X_1 = 1|X_0 = 1) \\ &= \frac{1}{2} \times (\frac{1}{2} - \epsilon) + \frac{1}{2} \times (\frac{1}{2} + \epsilon) \\ &= \frac{1}{2}\end{aligned}$$

Therefore we know that $\mathbb{E}(X_2) = \frac{1}{2}$, and ... $\mathbb{E}(X_k) = \frac{1}{2}$.

So this property is exactly the same as a fair coin.

Problem 12.4.2. Computationally estimate the mean of the coin by simulating 10, 20, and 100 throws for $\epsilon = 0$.

The results of using sampling to estimate the mean of the coin are shown in Figure 12.2. The error (via the Monte Carlo Central Limit Theorem) decreases as $\frac{1}{\sqrt{n}}$.

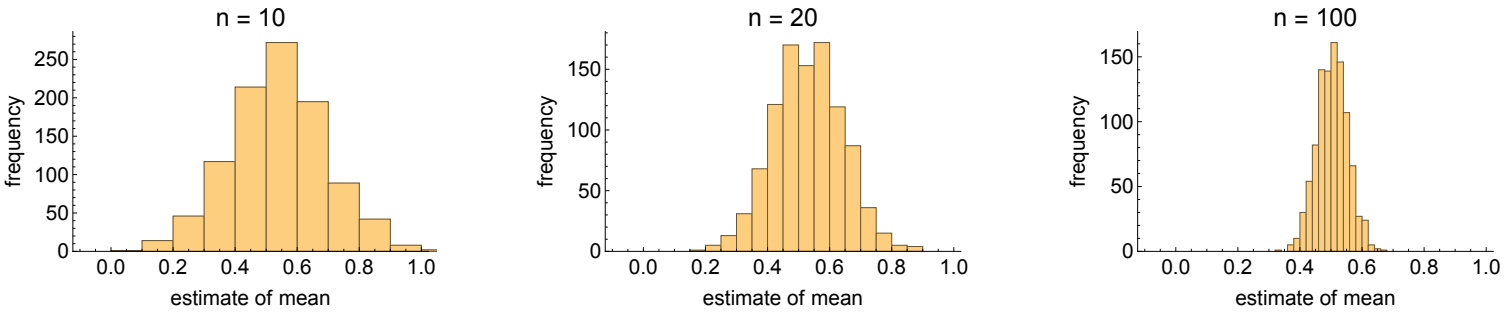

Figure 12.2: Estimating the mean of a Markovian coin (where $\epsilon = 0$, i.e. it is fair), using sampling across three sample sizes. In all cases 1,000 iterates were used to estimate the sampling distribution.

Problem 12.4.3. As $\epsilon \uparrow$ how does the error in estimating the mean change, and why?

As $\epsilon \uparrow$ there is increased *dependence*, meaning that the information garnered from each incremental sample is less than for the purely *independent* case (Figure 12.3).

Problem 12.4.4. When $\epsilon = \frac{9}{20}$ calculate the effective sample size of an actual sample size of 100. How does the effective sample size depend on ϵ?

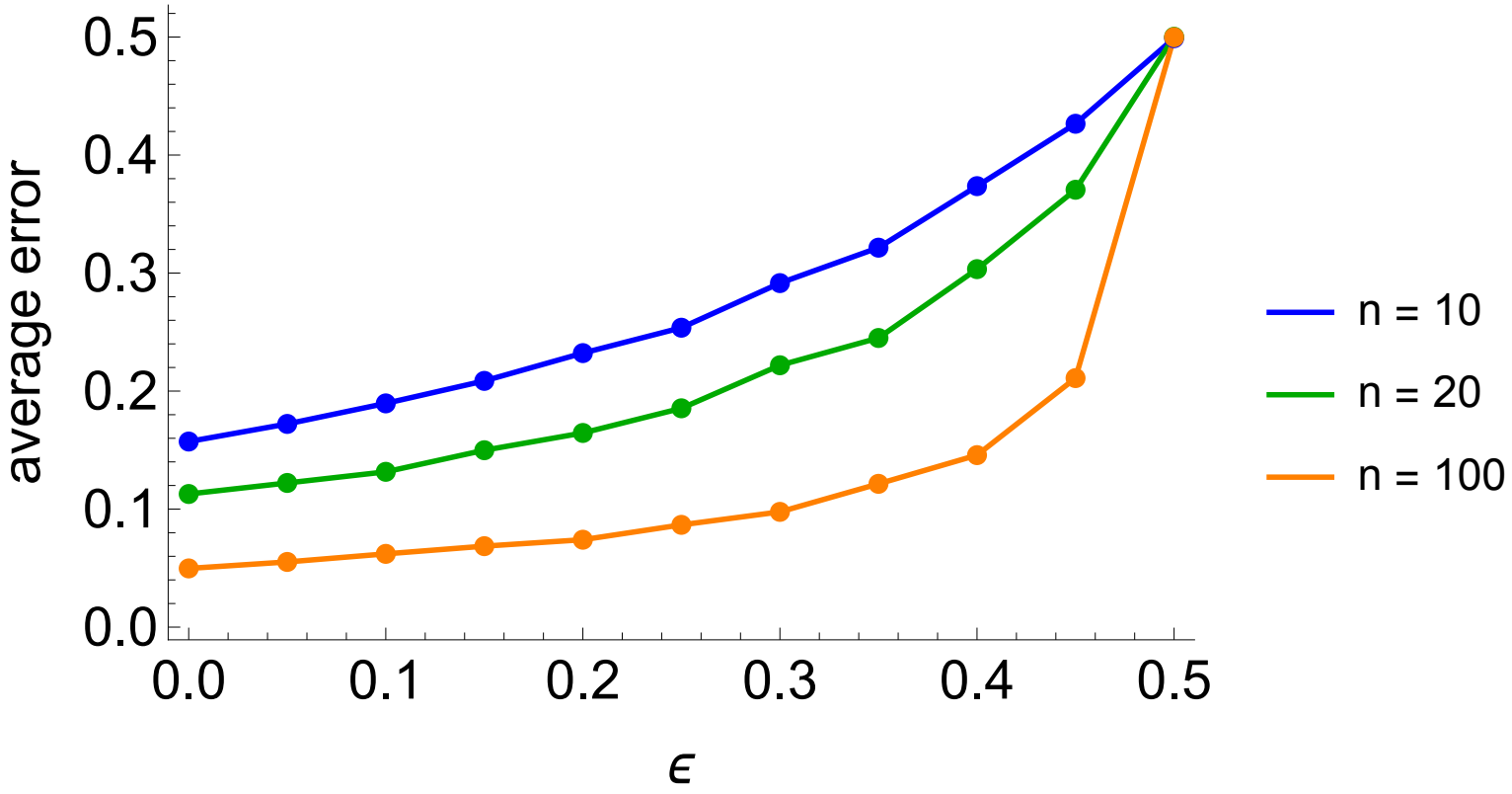

Figure 12.3: Standard error in estimating the mean of the Markovian coin, across different levels of ϵ.

The equivalent sample size for an *independent* coin, for another with $\epsilon = \frac{9}{20}$ and 100 samples is about 6 (Figure 12.4)! This effect is due to dependence.

Problem 12.4.5. Now assume that $\epsilon = -\frac{9}{20}$. what is the effective sample size of an actual sample size of 100? Explain your result.

This is antithetic sampling, where there is a negative correlation between the values of the sampler at each time step. Consider a sample $\frac{X_1+X_2}{2}$. Calculating its variance:

$$var\left(\frac{X_1+X_2}{2}\right) = \frac{1}{4}\left(var(X_1) + var(X_2) - 2Cov(X_1, X_2)\right) \tag{12.10}$$

If $Cov(X_1, X_2) = 0$ as in independent sampling, then we get $var(\bar{X}) = \frac{1}{n}\sigma^2$, where σ^2 is the variance of one sample. However, if $Cov(X_1, X_2) < 0$, then we can achieve a variance $var(\bar{X}) < \frac{1}{n}\sigma^2$. So antithetic sampling beats independent! Intuitively this is because an antithetic sampler can visit the state space much more efficiently than an independent sampler.

By sampling we have an effective sample size of about 1600 for an antithetic sample size of 100 (Figure 12.5).

I am sure this can be proved semi-analytically, since the series of autocovariances follow a geometric

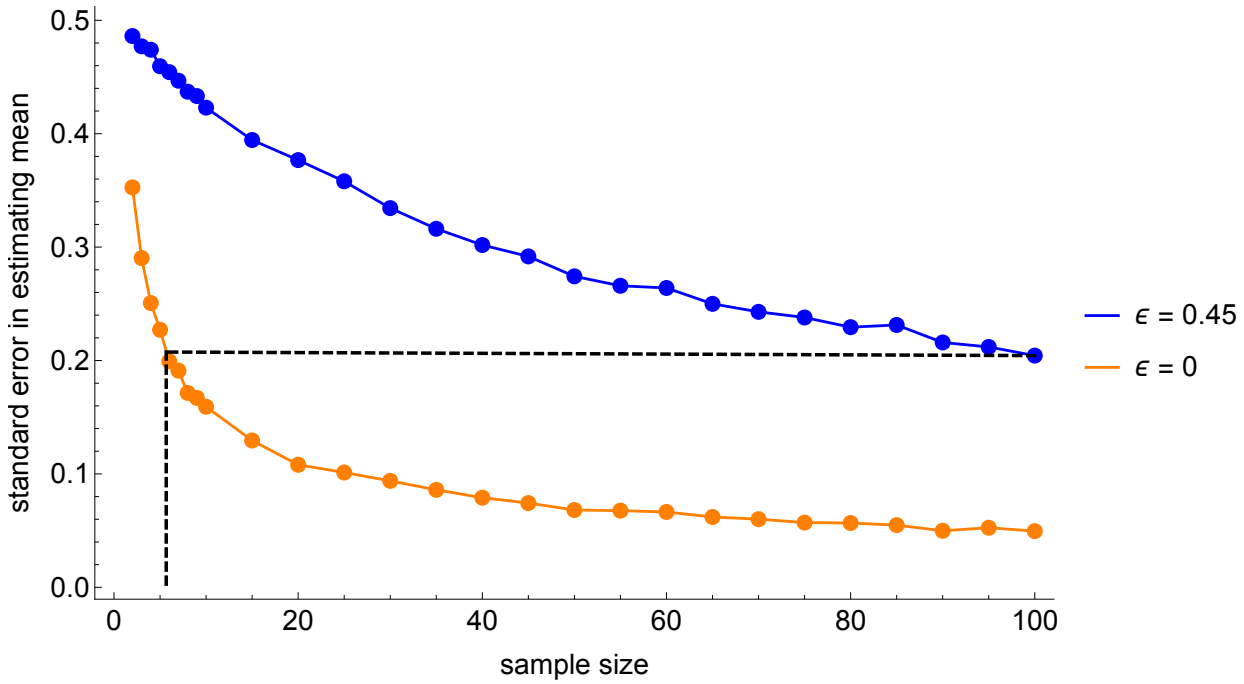

Figure 12.4: Calculating the effective sample size for $\epsilon = \frac{9}{20}$ for an actual sample size of 100.

series $(\epsilon/2, \epsilon^2, 2\epsilon^3, ...)$ (see Mathematica file). However, finding the sums is a bit tricky, and I'm not convinced it will simplify greatly!

12.5 Markovian die

Consider a type of die whose next value thrown can depend on the current value. The degree of dependence is specified by a parameter $0 \leq \epsilon \leq 1$ (see Figure 12.6). If $\epsilon = 0$ then each separate throw of the die can be considered independent of the previous value. Another way of saying this is that each number has an equal probability of being thrown irrespective of the current value. If $\epsilon = 1$ then there is strong dependence from one throw to the next, where from a given number on a throw only neighbouring numbers are possible on the next. So $1 \to (6, 2)$, $2 \to (1, 3)$ etc. If $0 < \epsilon < 1$ we suppose that there is preference towards consecutive numbers, with the preference increasing in ϵ.

For all values of ϵ we assume that both the forward and backward steps are equally likely, so $1 \to 2$ and $1 \to 6$ are of the same probability. If $0 < \epsilon < 1$, we suppose that those transitions that are not neighbours are all of the same probability (which is less than the probability of consecutive numbers).

Specifically, we define ϵ in the following way:

$$Pr(X_{n+1}|X_n) = \frac{1}{6}(1-\epsilon) + \frac{\epsilon}{2}1_{X_{n+1}\in\mathcal{C}(X_n)}$$

Where $1_{X_{n+1}\in\mathcal{C}(X_n)}$ is an indicator function which is equal to 1 if the next value of the die, X_{n+1} is

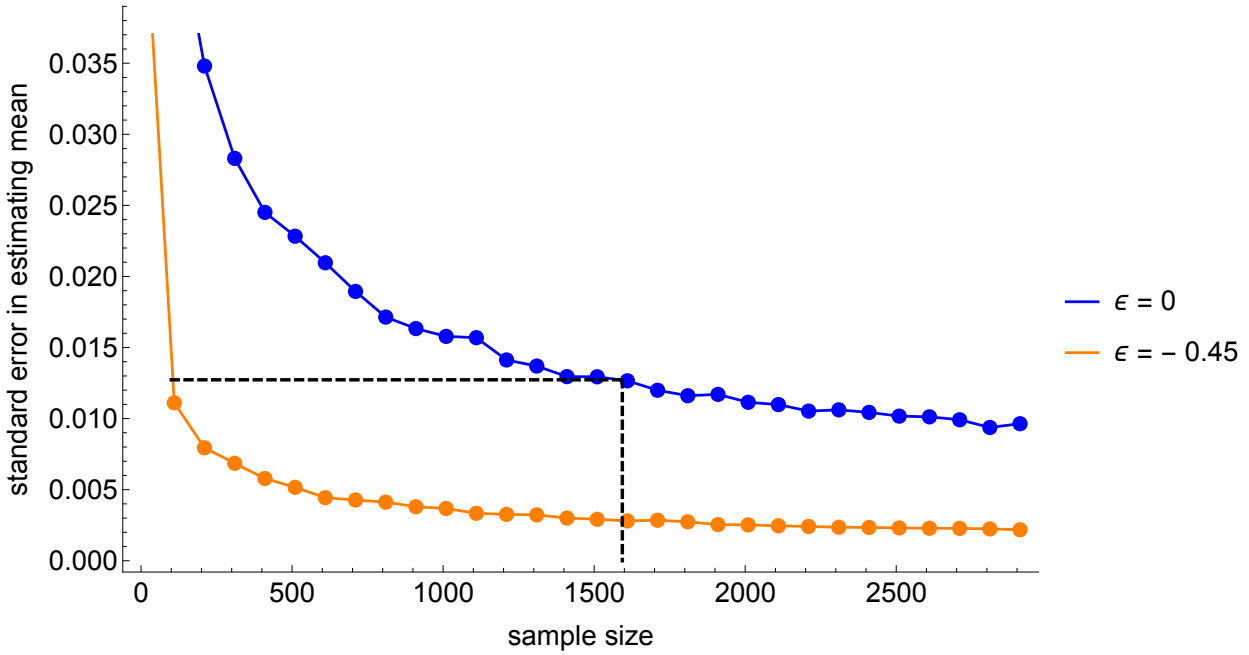

Figure 12.5: Calculating the effective sample size for $\epsilon = -0.45$ for an actual sample size of 100.

in the neighbour set $\mathcal{C}(X_n)$ of the current value, X_n. (The above is just a fancy way of saying that we increase the probability of neighbours by an amount $\frac{\epsilon}{2}$ relative to the non-neighbours.)

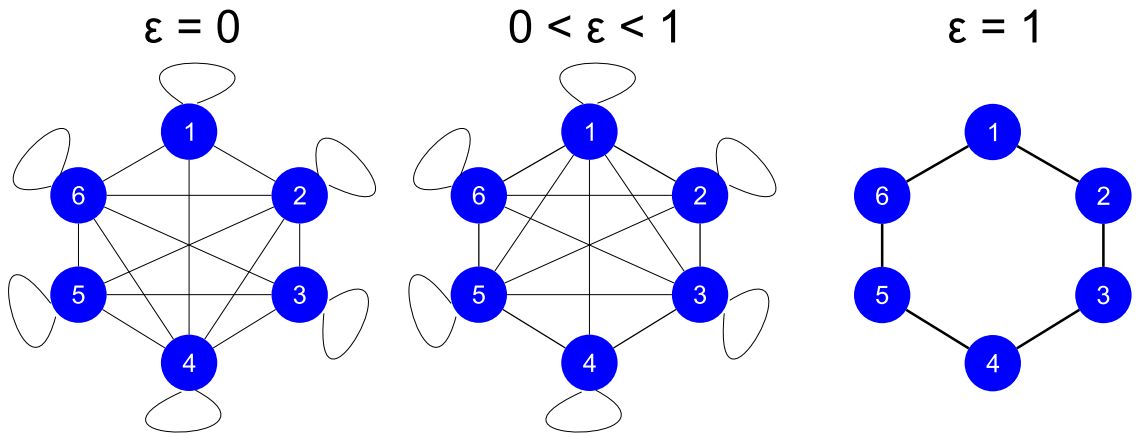

Figure 12.6: A Markovian die where ϵ determines the degree of dependence between throws of the die.

Problem 12.5.1. Find the mean of the die across all values of ϵ assuming it starts on a randomly-selected side.

Since it starts on a random side this means all numbers of equally likely (on the first throw), so its mean is just that of a typical die $\mu = \frac{7}{2}$. From then onwards we have the following conditional expectation:

$$\begin{aligned}\mathbb{E}(X_{n+1}|X_n) &= \sum_{i=1}^{6} Y_i \frac{1}{6}(1-\epsilon) + \sum_{i\in\mathcal{C}(X_n)} Y_i \\ &= \frac{7}{2}(1-\epsilon) + \sum_{i\in\mathcal{C}(X_n)} Y_i\end{aligned}$$

The only part of the above expression that depends on X_n is the second bit concerning the neighbours. So all we need to do is calculate this term for all possible neighbour combinations, and average over them yielding $\frac{7}{2}\epsilon$. Using this we can then find the *unconditional* expectation:

$$\mathbb{E}(X_{n+1}) = \frac{7}{2} \tag{12.11}$$

So the dependent die has the same mean as a typical fair die.

Problem 12.5.2. By simulating throws of the die, find an estimator of its mean.

The following Mathematica code does this simulation.

```
fPreferredSelector[aNumber_, n_Integer] :=
  If[aNumber == 1, {n, 2},
  If[aNumber == n, {n - 1, 1}, {aNumber - 1, aNumber + 1}]]

fPreferredBinary[aNumber_, n_Integer] :=
  If[# == 1, 1, 0] & /@ Mod[Abs[Range@n - aNumber], n - 2]

fProposalProbabilities[aNumber_, n_Integer, epsilon_] :=
  Module[{lPreferredSelector =
    fPreferredBinary[aNumber, n]},
        (0.5 - (epsilon/2)) lPreferredSelector + (epsilon / n)
          ConstantArray[1, n]]

toPiecewise[wts_, x_] :=
  Piecewise[MapIndexed[{#1, x == #2[[1]]} &, wts]]

fSelectNext[aNumber_, n_Integer, epsilon_] :=
  Module[{wts = fProposalProbabilities[aNumber, n, epsilon], f},
    f = ProbabilityDistribution[toPiecewise[wts, x], {x, 1, n, 1}];
    RandomVariate[f, {1}]][[1]]

fGenerateSamples[numSamples_, aStartNumber_, n_Integer, epsilon_] :=
  NestList[fSelectNext[#, n, epsilon] &, aStartNumber, numSamples]

fGenerateTransitions[maxSamples_Integer, n_Integer, epsilon_] :=
  Table[RandomVariate[ProbabilityDistribution[
    toPiecewise[fProposalProbabilities[i, n, epsilon], x], {x, 1, n,
```

```
1}], {maxSamples}], {i, 1, n, 1}]

fChain[numChains_Integer, numSamples_Integer, n_Integer, epsilon_] :=
  Module[{aNewEpsilon = 1 - epsilon, lAllTransitions, lTotal, lSteps},
    lAllTransitions = fGenerateTransitions[numChains numSamples,
                                                                    n, aNewEpsilon
    lSteps = Range@(numChains numSamples);
    lTotal = FoldList[fStep[#1, #2, lAllTransitions] &,
    RandomInteger[{1, n}], lSteps]; Partition[lTotal, numSamples]]

fStep[aCurrentNumber_Integer, aStepNumber_Integer, lSamples__] :=
  lSamples[[aCurrentNumber, aStepNumber]]

fVarianceEstimator[n_Integer, numChains_Integer, numSamples_, epsilon_] :=
  Variance[Mean /@ fChain[numChains, numSamples, n, epsilon]]
```

Problem 12.5.3. Compute the error in estimating the mean as ϵ is varied at a sample size of 5, 10,and 100.

The results of this are shown in Figure 12.7. As the sample size increases, the effect of dependence is less evident.

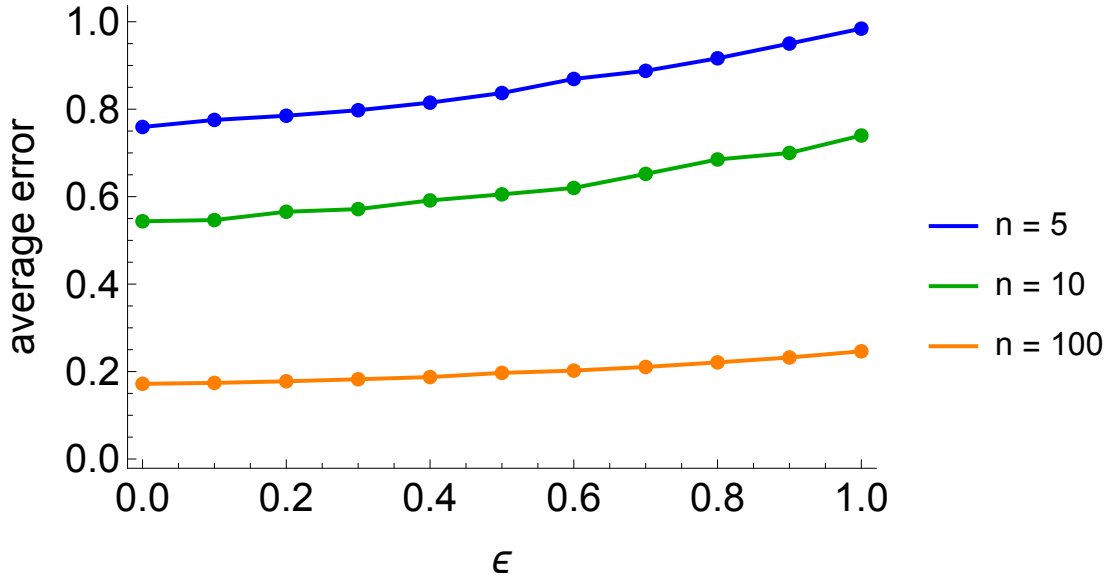

Figure 12.7: The effect of varying ϵ on the standard error in estimating the mean of a six-sided die.

Problem 12.5.4. Find the effective sample size of 100 throws (when estimating the mean) for a die where $\epsilon = 1$. Comment on the effect of dependence on sampling.

First you need to find the average error for an independent die, which I find to be about 0.17. To make an equivalent error on a die with $\epsilon = 1$, I find about 210 samples are necessary. So

dependence means that we need more samples to get the same quality of estimator. This problem is compounded as the size of the state space increases.

Problem 12.5.5. Now suppose that the die starts always on side 2. Find the expectation of the die (not the running total, just the current value) at each time step. (Difficult.)

The key to doing this is to set up the problem as a discrete Markov chain. To do this we create a transition matrix $\boldsymbol{P}$:

$$\boldsymbol{P} = \begin{pmatrix} \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} \\ \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} \\ \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} \\ \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} \\ \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} \\ \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{1}{6}-\frac{\epsilon}{6} & \frac{\epsilon}{3}+\frac{1}{6} & \frac{1}{6}-\frac{\epsilon}{6} \end{pmatrix}$$

If we assume the die starts in state $\boldsymbol{s(0)} = (0, 1, 0, 0, 0, 0)$ (in other words on the number 2), then we can calculate the probabilities of each state at each time step $\boldsymbol{s}$:

$$\boldsymbol{s(t)} = \boldsymbol{s(0)}\boldsymbol{P}^t \tag{12.12}$$

Since we know the probability of each state as a function of time, we can calculate the expectation:

$$\boldsymbol{s(t)} = \boldsymbol{s(0)}\boldsymbol{P}^t \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \tag{12.13}$$

Problem 12.5.6. Following on from the last question find how long we need to leave the die before we are confident we are sampling from its *unconditional* distribution. (By "unconditional" here, we mean its probability distribution disregarding its start point.) (Difficult.)

Figure 12.8 shows the conditional expectation as a function of time for different ϵ values. For $\epsilon << 1$ we see that we get a steady state expectation after about 10-15 samples. We see that for $\epsilon = 1$ we never get a constant expectation, but we do approach (periodic) stationarity in the distribution after a sample of 15. Both of these cases indicate that we have reached the *unconditional* distribution, where we have essentially forgotten the starting value.

Problem 12.5.7. Carry out the above investigations but for a die with n sides. How does n affect the results?

As $n \uparrow$ the effective sample size decreases for a given ϵ. Figure 12.9 also shows that we get increased sensitivity to ϵ for a 20 sided die; intuitively this sensitivity should increase with n.

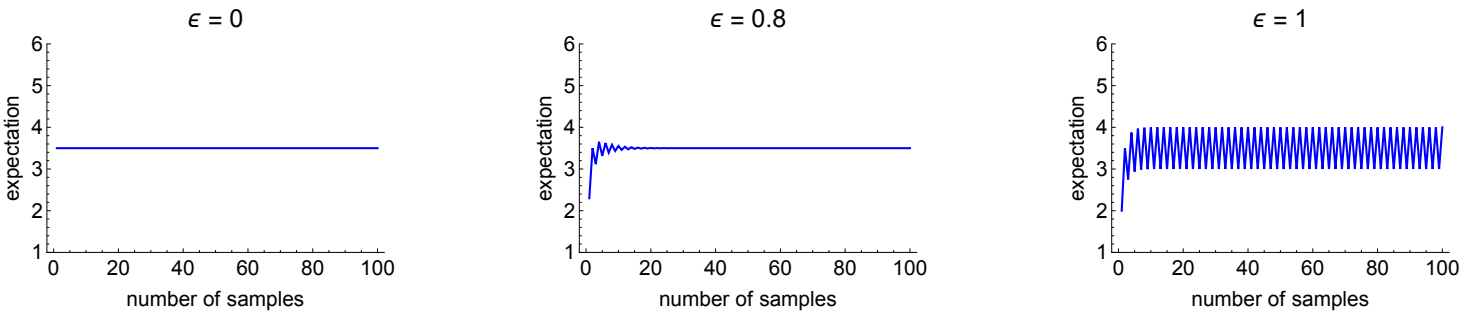

Figure 12.8: The effect of varying ϵ on the conditional expectation.

12.6 Turning a coin into a random-number generator

Suppose you have one coin that has equal probability of landings heads up versus tails up.

Problem 12.6.1. How can you use this coin to create a random variable X that has $Pr(X = 1) = 1/3$ and $Pr(X = 0) = 2/3$? (Hint: use rejection sampling.)

If you flip the coin twice and record its value Y on each flip you obtain,

$$Pr(00) = 1/4 \tag{12.14}$$
$$Pr(10) = 1/4 \tag{12.15}$$
$$Pr(01) = 1/4 \tag{12.16}$$
$$Pr(11) = 1/4 \tag{12.17}$$

so there are four outcomes, all of equal chance. What you want to do is convert something that has four outcomes into another thing that has three. So you assign: $00 \rightarrow X = 0$, and or$(10, 01) \rightarrow X = 1$. If the coin lands heads up twice, you just reject that sample, and repeat the exercise until you get one of the other three outcomes. This gives you the 1/3 to 2/3 probabilities of each outcome as you needed.

Problem 12.6.2. In R use a computational fair coin (i.e. a bernoulli distribution with $\theta = 0.5$) to create a random variable that is *approximately* distributed as a standard normal.

Use the central limit theorem,

$$\sqrt{n}(\overline{X} - 0.5) \xrightarrow{p} \mathcal{N}(0, \sigma) \tag{12.18}$$

So if we take enough samples the distribution of $\overline{X} \approx \mathcal{N}(0.5, \frac{\sigma}{\sqrt{n}})$. Doing this in R,

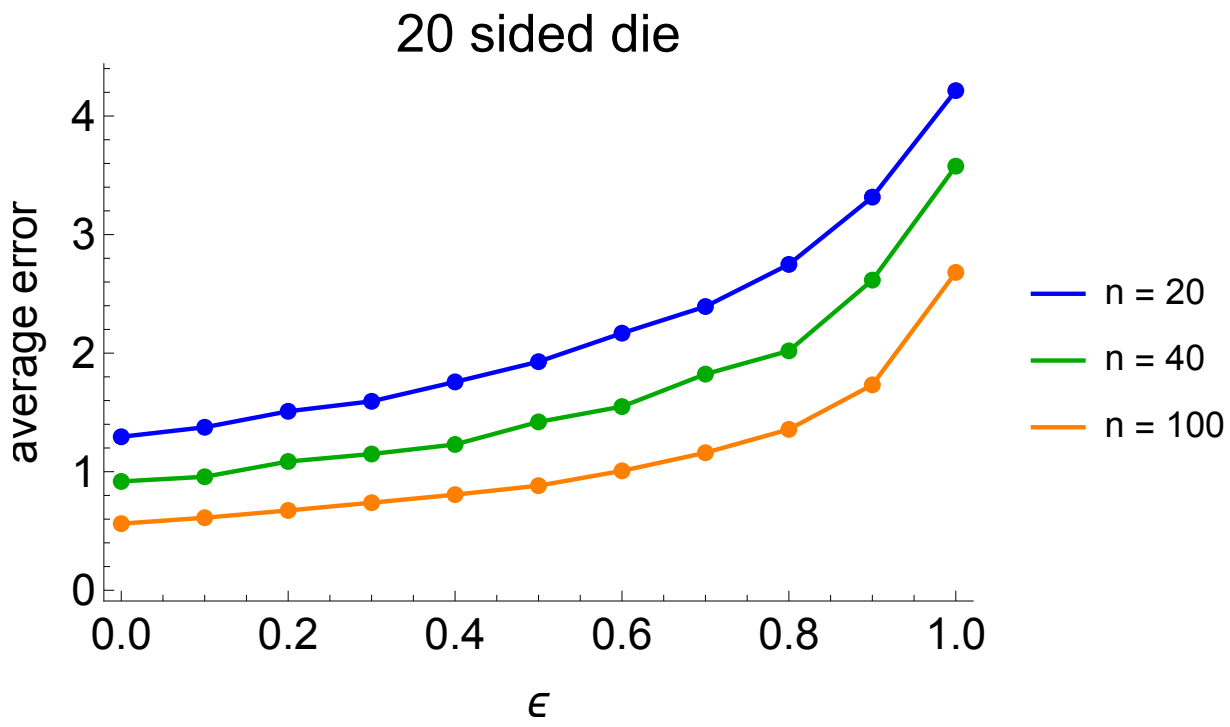

Figure 12.9: The effect of varying ϵ on the average error in estimating the mean for a 20-sided die.

```
fCLT <- function(numSamples){
  X <- sapply(seq(1, numSamples, 1), function(i) rbinom(1, 1, 0.5))
  return(mean(X))
}

hist(sapply(seq(1, 1000, 1), function(i) fCLT(1000)), 100)
```

How to convert this into a standard normal we need to know $\sigma^2 = 0.5(1-0.5) \implies \sigma = 0.5$. Therefore the following should yield a standard normally-distributed rv,

```
fCLT1 <- function(numSamples){
  X <- sapply(seq(1, numSamples, 1), function(i) rbinom(1, 1, 0.5))
  return(sqrt(numSamples) * (mean(X) - 0.5) / 0.5)
}

lData <- sapply(seq(1, 1000, 1), function(i) fCLT1(1000))
hist(lData, 100)
sd(lData)
```

where the sd is close to 1.

Problem 12.6.3. Using the answer to the previous question create a variable that is approximately uniformly distributed between 0 and 1.

Just use the previous answer to generate CDF values from the theoretic $\mathcal{N}(0.5, \frac{\sigma}{\sqrt{n}})$ distribution,

```
fNormalQuantiles <- function(numSamples, numPerSample){
  X <- sapply(seq(1, numSamples, 1), function(i) fCLT(numPerSample))
  lCDF <- pnorm(X, 0.5, sqrt(0.5 * (1 - 0.5)) / sqrt(numPerSample))
  return(lCDF)
}
lCDF <- fNormalQuantiles(10000, 1000)
hist(lCDF)
```

12.7 Pseudo-random-number generators

Problem 12.7.1. A particular pseudo-random-number generator is known as the linear congruential generator which generates a sequence of pseudo-randomised numbers using the relation,

$$s_t = as_{t-1} + b \bmod M \tag{12.19}$$

where a, b and M are suitably chosen positive integers. What is the maximum period that such a sequence can have?

Since the sequence is bounded above my M this is the maximum number of unique values this sequence can take.

Problem 12.7.2. Write a function that implements the above recurrence relation and hence show that when $a = 2$, $b = 3$ and $M = 10$, where we begin with $s_0 = 5$ (the seed), the series has a period of 4.

Implementing this function in Mathematica we have the following,

```
fLinearCongruential[seed_Integer, a_Integer, b_Integer, M_Integer] :=
  Mod[a seed + b, M]

fLinearCongruentialSeries[numSamples_Integer, seed_Integer, a_Integer,
    b_Integer, M_Integer] :=
  NestList[fLinearCongruential[#, a, b, M] &, seed, numSamples]

fLinearCongruentialSeries[10, 5, 2, 3, 10]

{5, 3, 9, 1, 5, 3, 9, 1, 5, 3, 9}
```

which has repeats itself every fourth element.

Problem 12.7.3. Create a new function that has a maximum of one and a minimum of zero.

Simply divide the above output by M,

```
fLinearCongruentialUniform[numSamples_Integer, seed_Integer, a_Integer,
                           b_Integer, M_Integer] :=
  fLinearCongruentialSeries[numSamples, seed, a, b, M] / M
```

Problem 12.7.4. Use your newly created function with $a = 1229$, $b = 1$ and $M = 2048$ where we begin with $s_0 = 1$ (the seed) to generate 10,000 numbers between zero and one. Draw a histogram of the resultant sample. What sort of distribution does this look like?

It looks like a continuous uniform distribution (Figure 12.10).

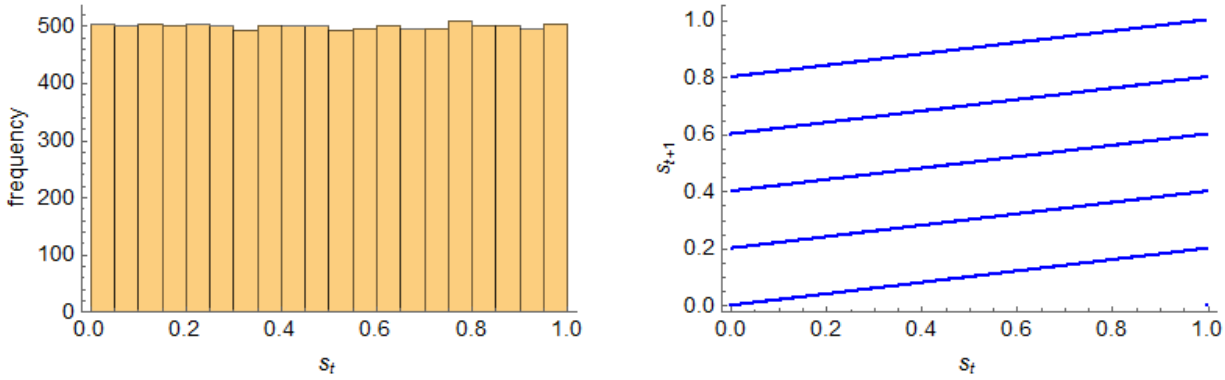

Figure 12.10: Left: the histogram and right: pairwise consecutive samples for the linear congruential generator with $a = 1229$, $b = 1$ and $M = 2048$ where we begin with $s_0 = 1$.

Problem 12.7.5. Draw a scatter plot of pairs of consecutive samples for the previously-generated series. Does this series look random?

No there is definitely patterning (Figure 12.10).

Problem 12.7.6. Now generate a series with $a = 1597$, $b = 51749$ and $M = 244944$, beginning with $s_0 = 1$, to generate 10,000 numbers between zero and one. Draw a histogram of the resultant sample. What sort of distribution does this look like? Does a scatter plot of consecutive pairs look random?

Again it looks like a continuous uniform distribution, and this time the consecutive pairs look more random (Figure 12.11).

Problem 12.7.7. Prove that inverse transform sampling works.

First we need to state the method itself mathematically. If $u \sim U(0, 1)$ then we suppose that $y = F^{-1}(u) \sim F$. To prove this,

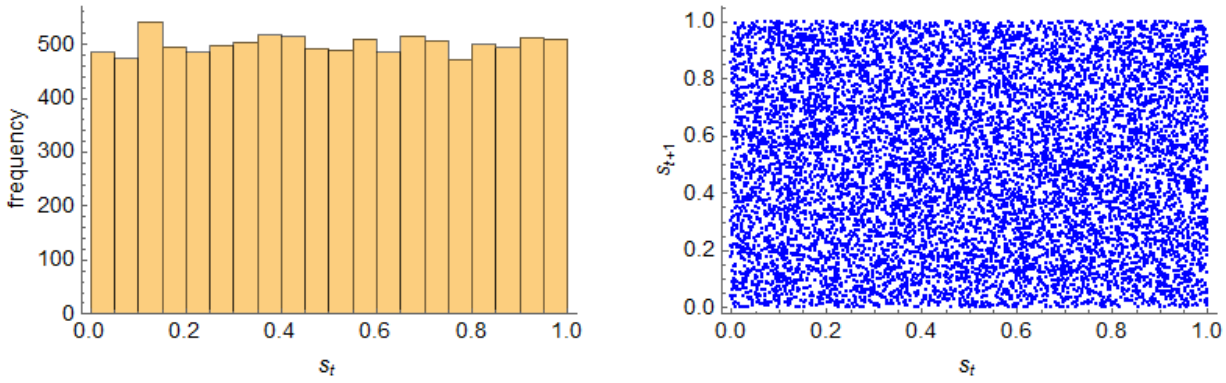

Figure 12.11: Left: the histogram and right: pairwise consecutive samples for the linear congruential generator with $a = 1597$, $b = 51749$ and $M = 244944$ where we begin with $s_0 = 1$.

$$\begin{aligned} Pr(y \leq x) &= Pr(F^{-1}(u) \leq x) && (12.20)\\ &= Pr(FF^{-1}(u) \leq F(x)) && (12.21)\\ &= Pr(u \leq F(x)) && (12.22)\\ &= F(x) && (12.23) \end{aligned}$$

This means that $y \sim F$.

Problem 12.7.8. Use your most recent sequence of numbers from the linear congruential generator along with inverse transform sampling to generate pseudo-independent samples from the following density $F(x) = 1 - exp(-\sqrt{x})$.

We need to find the inverse function meaning,

$$u = 1 - exp(-\sqrt{x}), \tag{12.24}$$

which yields,

$$x = \left[\log\left(1 - u\right)\right]^2. \tag{12.25}$$

Applying the above function to our sequence we obtain samples that look as if they come from the requisite distribution (remember to differentiate the CDF to yield the PDF).

Problem 12.7.9. Using the inverse transform method or otherwise use your sequence linear congruential generator to generate samples from a standard normal distribution.

To do with we need to find the inverse CDF for a standard normal. There is no analytic solution here and so numeric integration is the only approach. Once you have a numerical CDF versus a range of x you can create a interpolating function that returns the inverse-CDF.

An alternative approach is to use the Box-Muller transforms.

Bibliography