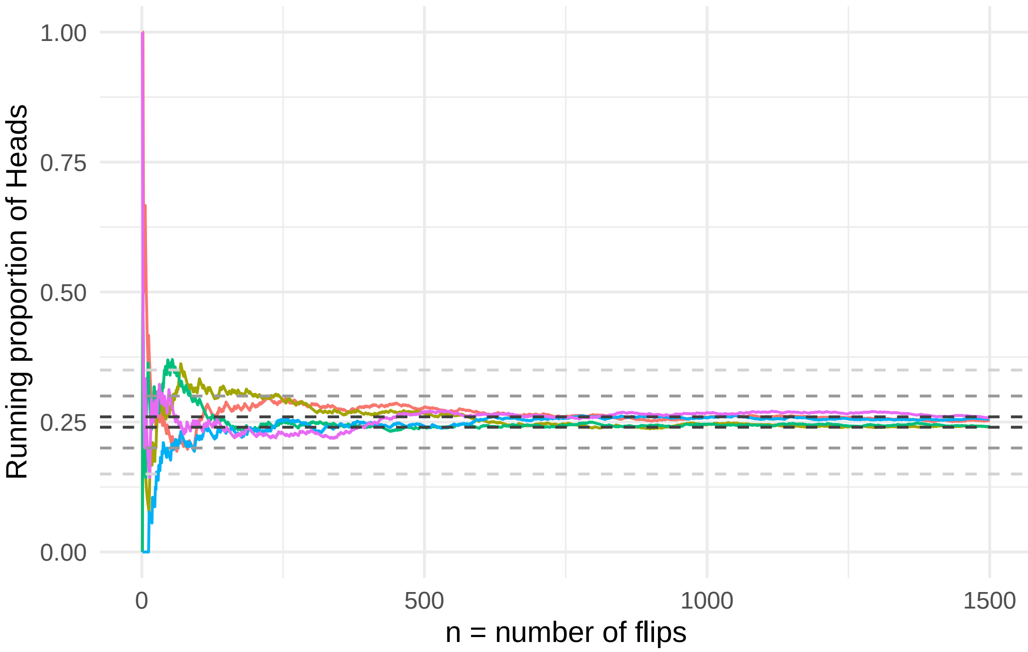
Law of Large Numbers

Coin flips

$X_1, X_2, ...$ are IID Bern(1/4) random variables, where X_i is indicator of Heads. Then $\bar{X}_n$ is the proportion of Heads in our n tosses.

The WLLN tells us that as n increases, then for any $\epsilon > 0$, the probability that $\bar{X}_n$ is more than ϵ away from its mean of 1/4 can be made arbitrarily small.

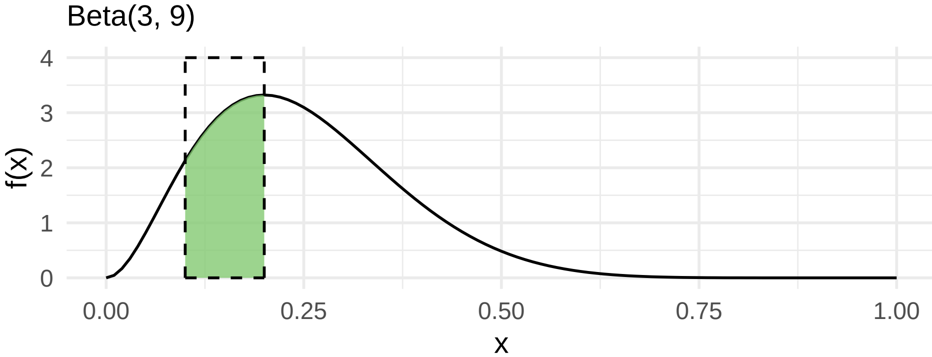
In the following plot, we simulate 5 different running means, represented by the colors of the lines. The dashed horizontal bands represent varying ϵ.



Monte carlo integration

Estimating Beta probability

Suppose $X \sim \text{Beta}(3, 9)$ and I want to approximate $P(0.1 < X < 0.2)$ (i.e. the shaded area below). I know I can evaluate the function/PDF of X using `dbeta()` or by hand, but suppose I do not have access to `pbeta()` for probabilities.



Monte Carlo will help us! We have to come up with a bounding region to sample uniform points from. Well, we want $x \in (0.1, 0.2)$. For the y-axis, we see that any c greater than roughly 3.25 will do the trick. We will use $c = 4$. Our bounding rectangle is depicted above.

```
# simulate uniformly within bounding rectangle
set.seed(1)
B <- 5000
x <- runif(B, 0.1, 0.2)
y <- runif(B, 0, 4)

# obtain proportion that fall into area under curve
under <- mean(y <= dbeta(x, 3, 9))

# multiply by bounding rectangle area
mc_approx <- under * 4 * (0.2 - 0.1)
```

So our Monte Carlo approximation to the integral is 0.29024.

Suppose we suddenly have access to the CDF function `pbeta()`! Let's compare our MC approximation to the truth:

```
pbeta(0.2, 3, 9) - pbeta(0.1, 3, 9)
```

```
[1] 0.2930366
```

Estimating *pi*

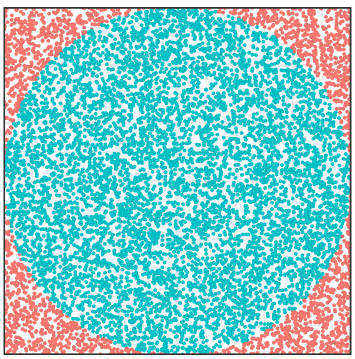
```
set.seed(10)

# choose area of circle/dart board
r <- 3

# sample uniformly within square
n <- 10000
x <- runif(n, -r, r)
y <- runif(n, -r, r)

# determine which points are in the circle
in_circle <- x^2 + y^2 <= r^2

# mc approximation
mc_pi <- 4 * sum(in_circle)/n
```



Our estimate of π is 3.1432, which is pretty good!