Lab 4

Senan Hogan-H.

5 April 2018

Completed with James Kinney, around 50/50 work completed between us.

Question 1.

a. Define a random walk with no use of the MH ratio.

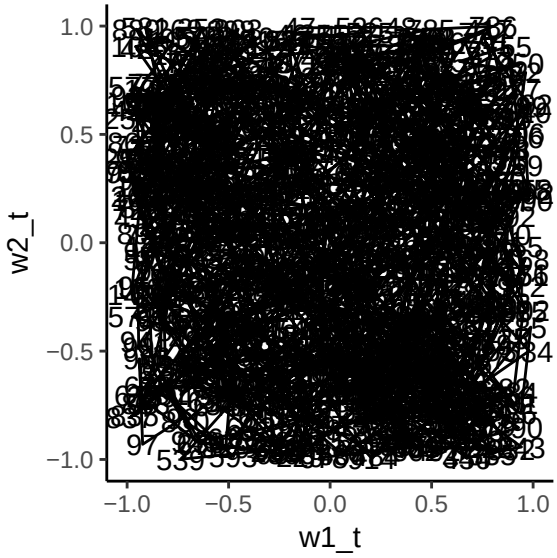
```
# Define function that returns a dataframe of a random walk in 2 dimensions

random_walker <- function(w1_0, w2_0, epsilon, n){
  t <- c(0:n) # define chain
  w1_t <- c(w1_0) # define chain
  w2_t <- c(w2_0) # define chain
  for (i in c(1:n) ){
    # First coordinate
    error1 <- runif(1, -epsilon , epsilon)
    w1_move <- w1_t[i] + runif(1, -epsilon , epsilon)
    while (w1_move > 1 |  w1_move < -1){ # reject of not in stationary support
      error1 <- runif(1, -epsilon , epsilon)
      w1_move <- w1_t[i] + runif(1, -epsilon , epsilon)
    }

    # Second coordinate
    error2 <- runif(1, -epsilon , epsilon)
    w2_move <- w2_t[i] + runif(1, -epsilon , epsilon)
    while (w2_move > 1 |  w2_move < -1){ # reject of not in stationary support
      error2 <- runif(1, -epsilon , epsilon)
      w2_move <- w2_t[i] + runif(1, -epsilon , epsilon)
    }

    # Move with 100% probability to new coordinates.
    w1_t[i+1] <- w1_move
    w2_t[i+1] <- w2_move
  }
  return(data.frame(w1_t, w2_t, t))
}

# Define function that returns a graph of a random walk in 2 dimensions
# input is a dataframe created by first function

random_walker_graph <- function(data){
  graph <- data %>% ggplot(aes(x = w1_t, y = w2_t)) +
    geom_text(aes(label = t)) +
    geom_path() +
    coord_cartesian(xlim = c(-1, 1), ylim = c(-1, 1)) +
    theme_classic()
  return(graph)
}
```

```r
# Define function that returns an estimate of pi from a random walk in 2 dimensions
# input is a dataframe created by first function

random_walker_pi <- function(data){
  t <- c(1:nrow(data))
  w1 <- w2 <- pi_estimate <- c()
  for (i in t){
    w1 <- c(w1, data$w1_t[i])
    w2 <- c(w2, data$w2_t[i])
    w <- w1^2 + w2^2
    pi_estimate <- c(pi_estimate, 4*length(w[w < 1])/length(w))
  }
  graph <- data.frame(t, pi_estimate) %>%
    ggplot(aes(x=t, y = pi_estimate)) +
    geom_point() + geom_line() +
    geom_hline(yintercept=pi, linetype='dashed') +
    coord_cartesian(ylim = c(2, 4)) +
    theme_classic()
  return(graph)
}
```

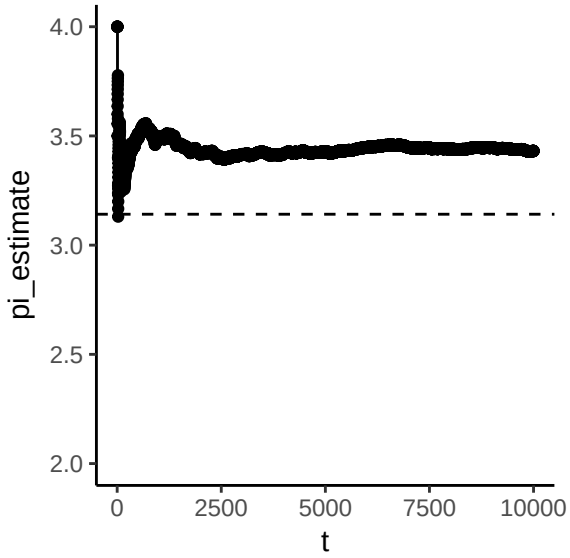
There is a problem here because the chain spends less time near the edges of the support. This is because a point in the chain near the edge of the support is more likely to choice a next point in the random walk outside of the support, and thus have this next point rejected until a point inside the support picked for the next point. This leads to a tendency for the chain to wander pack to the centre of the support if it deviates to the edges.

This leads to an ver-estimate of π in the unit square. A quarter of the unit square would be an even more distorted picture, because the chain would spend a dissproportionate amount of time in the centre of the [0,1] square, providing an unclear amount of bias for estimating the area of the 90 degree slice of the unit circle. See the following for a demonstration of this.

```r
data <- random_walker(0, 0, 0.5 , 1000)

random_walker_graph(data) # area around the edges is more sparse than centre.
```



```
data <- random_walker(0, 0, 0.5 , 10000)

random_walker_pi(data) # Over-estimates pi.
```

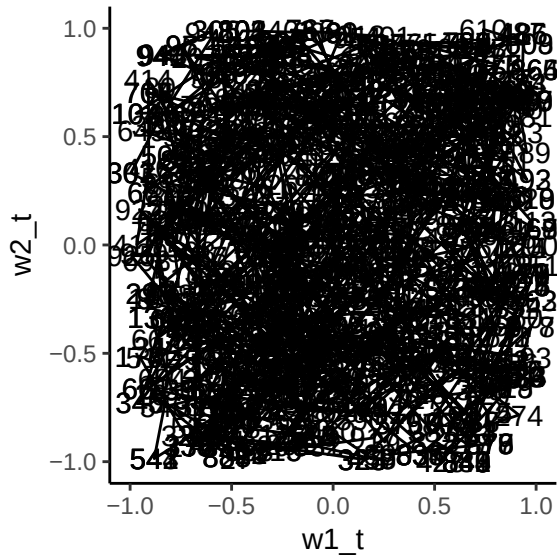


d. The following adds a burn-in, making the next point in the chain the same as the previous if a draw is drawn outside of the support. This gives greater density to the edges of the square, and leads to a better estimate of π.

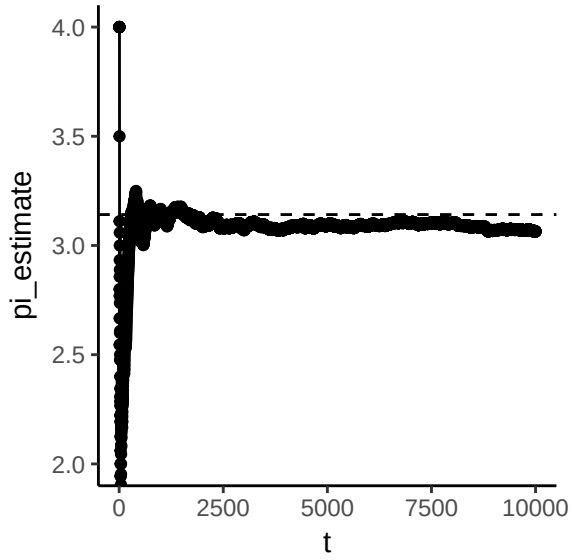
```
# Define function that returns a dataframe of a random walk in 2 dimensions
# Now considers MH ratio.

random_walker <- function(w1_0, w2_0, epsilon, n){
  t <- c(0:n) # define chain
  w1_t <- c(w1_0) # define chain
  w2_t <- c(w2_0) # define chain
  for (i in c(1:n) ){
    # First coordinate
    error1 <- runif(1, -epsilon , epsilon)
    w1_move <- w1_t[i] + runif(1, -epsilon , epsilon)



    # Second coordinate
    error2 <- runif(1, -epsilon , epsilon)
    w2_move <- w2_t[i] + runif(1, -epsilon , epsilon)

    # Move with to new coordinates only if inside the unit square.
    w1_t[i+1] <- w1_t[i]
    w2_t[i+1] <- w2_t[i]

    if (abs(w1_move) < 1 & abs(w2_move) < 1){
      w1_t[i+1] <- w1_move
      w2_t[i+1] <- w2_move
    }
```

```
  }
  return(data.frame(w1_t, w2_t, t))
}

# Define function that returns a graph of a random walk in 2 dimensions
# input is a dataframe created by first function

random_walker_graph <- function(data){
  graph <- data %>% ggplot(aes(x = w1_t, y = w2_t)) +
    geom_text(aes(label = t)) +
    geom_path() +
    coord_cartesian(xlim = c(-1, 1), ylim = c(-1, 1)) +
    theme_classic()
  return(graph)
}

# Define function that returns an estimate of pi from a random walk in 2 dimensions
# input is a dataframe created by first function

random_walker_pi <- function(data){
  t <- c(1:nrow(data))
  w1 <- w2 <- pi_estimate <- c()
  for (i in t){
    w1 <- c(w1, data$w1_t[i])
    w2 <- c(w2, data$w2_t[i])
    w <- w1^2 + w2^2
    pi_estimate <- c(pi_estimate, 4*length(w[w < 1])/length(w))
  }
  graph <- data.frame(t, pi_estimate) %>%
    ggplot(aes(x=t, y = pi_estimate)) +
    geom_point() + geom_line() +
    geom_hline(yintercept=pi, linetype='dashed') +
    coord_cartesian(ylim = c(2, 4)) +
    theme_classic()
  return(graph)
}
```

```
data <- random_walker(0, 0, 0.5 , 1000)

random_walker_graph(data) # area around the edges is no more sparse than centre.
```

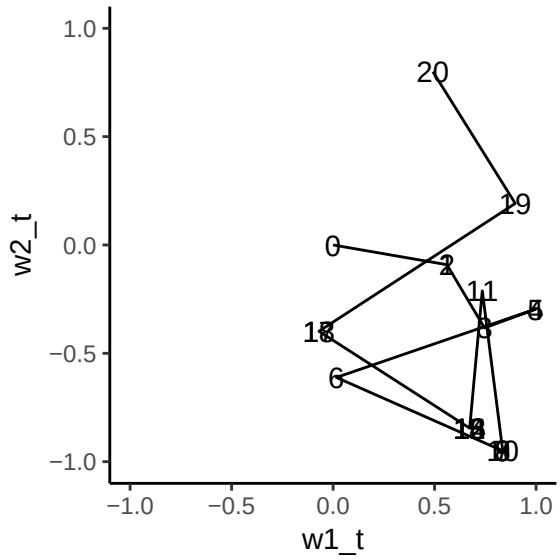



```
data <- random_walker(0, 0, 0.5 , 10000)

random_walker_pi(data) # Estimates pi well.
```

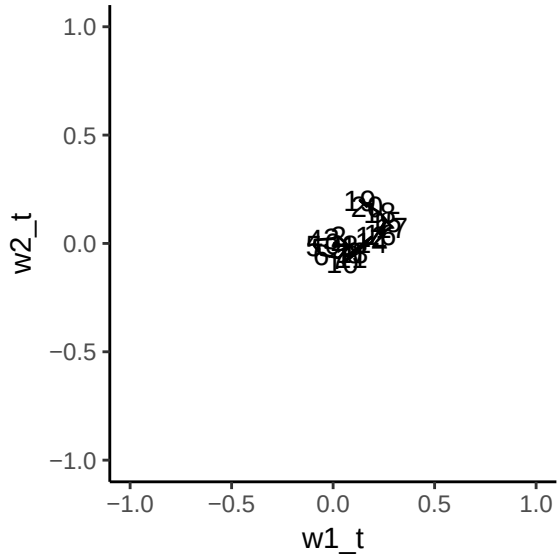


i. Choice of ϵ. If *epsilon* is exceptionally small, then the random walk explores less of the space. This gives a much worse estimate of the uniform distribution as the chain stays around a much smaller area in the space, and thus a worse estimate of π.

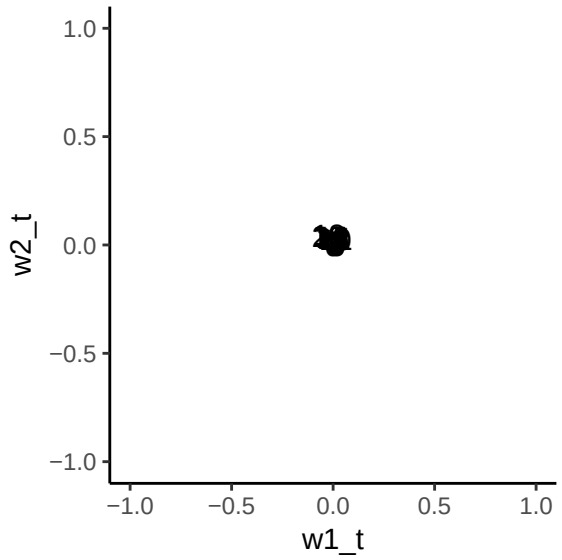
```
data <- random_walker(0, 0, 1 , 20)

random_walker_graph(data)
```



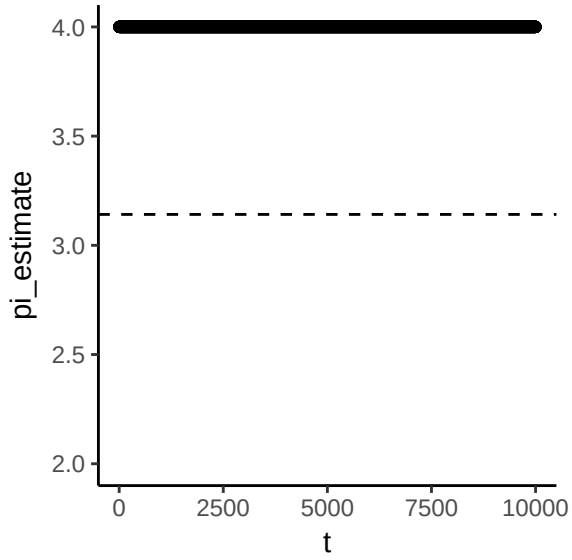

```
data <- random_walker(0, 0, 0.1 , 20)

random_walker_graph(data)
```




```
data <- random_walker(0, 0, 0.01 , 20)

random_walker_graph(data)
```

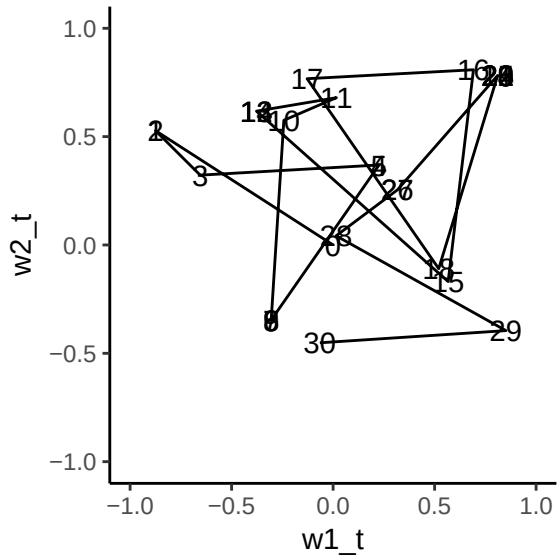



```
data <- random_walker(0, 0, 0.001 , 10000)

random_walker_pi(data)
```

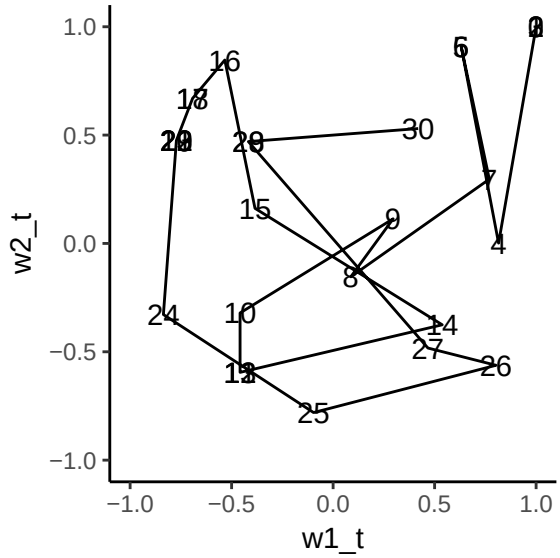


ii. Choice of w_0. w_0 generally gets washed by a long enough chain, though this does require a large enough value of epsilon. So that choice of w_0 does not theoretically matter, and different values of w_0 gives very similar estimates of pi.

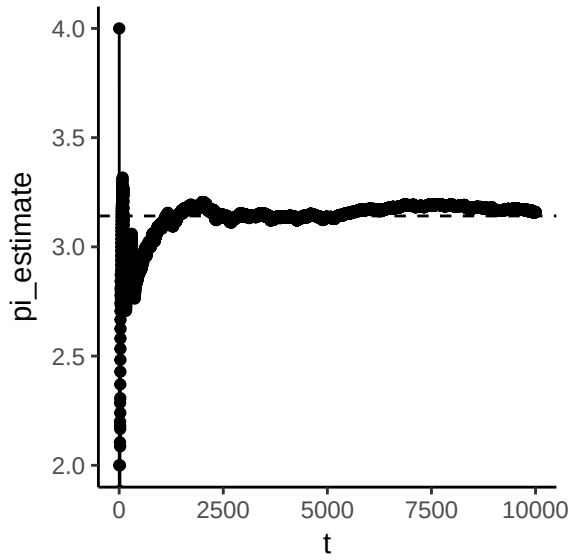
```
data <- random_walker(0, 0, 1 , 30)

random_walker_graph(data)
```



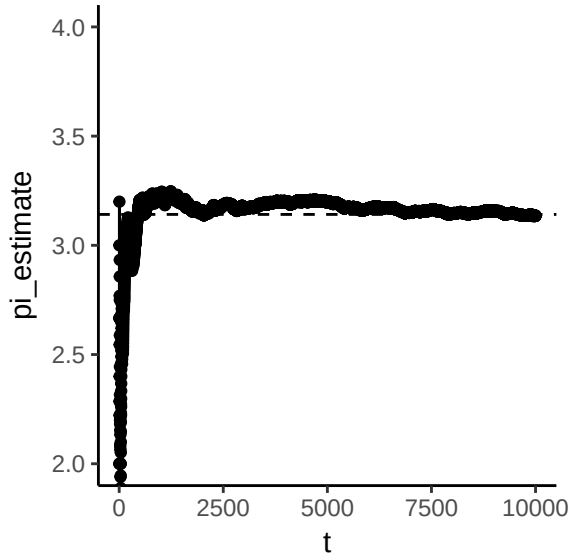

```
data <- random_walker(1, 1, 1 , 30)

random_walker_graph(data)
```




```
data <- random_walker(0, 0, 1 , 10000)

random_walker_pi(data)
```

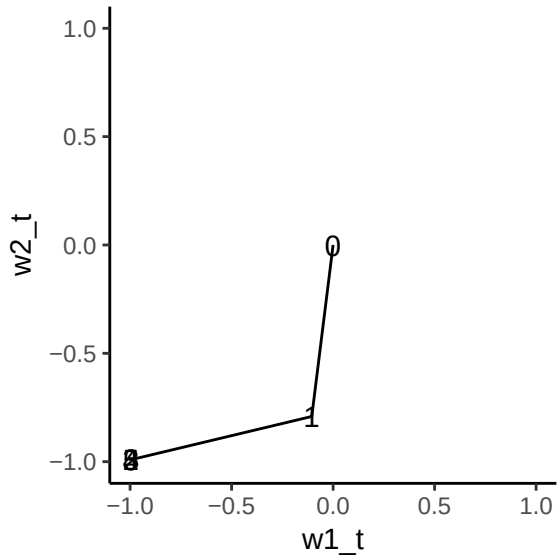



```
data <- random_walker(1, 1, 1 , 10000)

random_walker_pi(data)
```

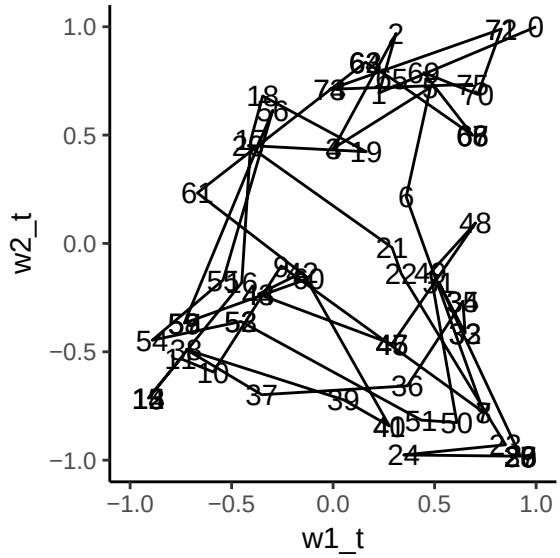


iii. Choice of chain length, n. Longer chains give more a convergent estimate of π, i.e. low values of n give variable estimates, which makes sense given the law of of large numbers. Also, shorter chains do not have chance to explore the sample space as much as longer chains do.

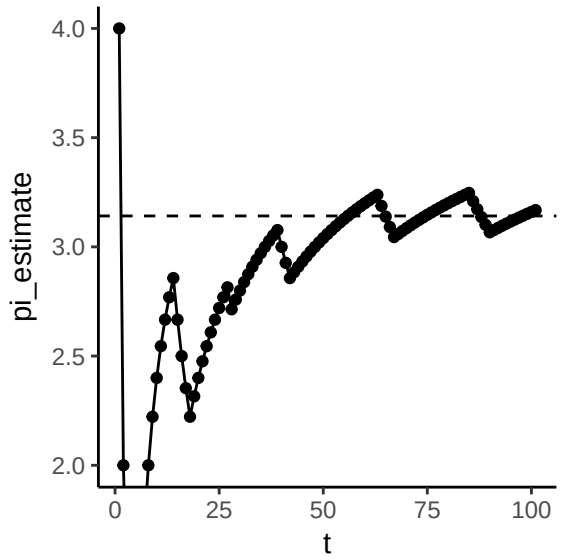
```
data <- random_walker(0, 0, 1 , 5)

random_walker_graph(data)
```



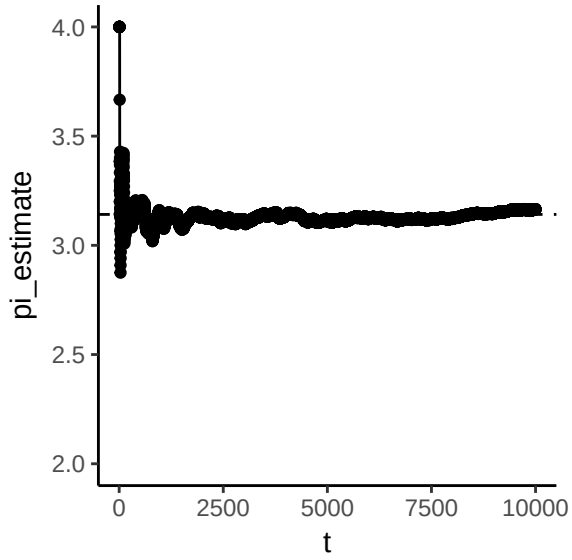

```
data <- random_walker(1, 1, 1 , 75)

random_walker_graph(data)
```




```
data <- random_walker(0, 0, 1 , 100)

random_walker_pi(data)
```




```
data <- random_walker(0, 0, 1 , 10000)

random_walker_pi(data)
```



Question 2.

Let's suppose our data is coming from a binomial with parameters $n = 100$ and $p = \frac{1}{4}$.

Then a good prior would be a beta centered around 0.25. So $\frac{\alpha}{\alpha+\beta} = .25$ and thus $.75\alpha = .25\beta \implies 3\alpha = \beta$. For our good prior, let $alpha_g = 10$ and $\beta_g = 30$. For our bad prior, let's swap the two so that the mean is .75. Let $\alpha_b = 30$ and $\beta_b = 10$.

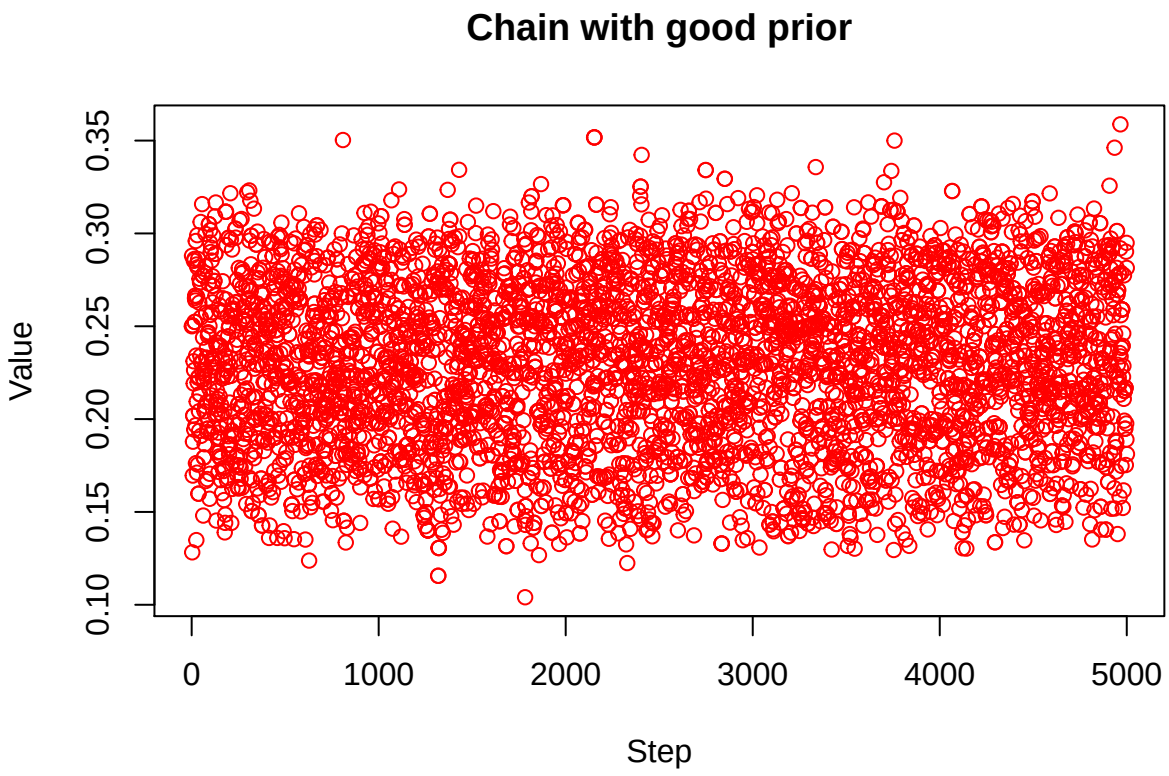
The function we want to estimate is the posterior. The function we use as our g is the prior. The actual posterior with the good prior is $p_g(x) = \text{Beta}(X + 10, 100 - X + 30)$ where X is the result of the binomial experiment. The bad posterior is $p_b(x) = \text{Beta}(X + 30, 100 - X + 10)$. We want to estimate the mean, medial and 95% credible interval for these posteriors using the M-H ratio and a random walk.

Let $p(x)$ be the posteiror of x and let $g(x)$ be the prior of x. So the M-H ratio is $R(x,y) = \min\{1, \frac{p(y)g(x|y)}{p(x)g(y|x)}\}$.

```
#Random walk with good prior
betaWalk = function(Xbinom, xNot, n, a, b){
  steps = c() # A vector to store our locations
  x = xNot; steps[1] = xNot #A starting place
  for(i in 2:(n+1)){
    y = rbeta(1, a, b) #Simulate a y from the beta distribution
    MH = min(1, dbeta(y, a + Xbinom, 100-Xbinom+b))#Compute the M-H ratio
    if(runif(1,0,1) < MH){ #With prob M-H, move to y
      x=y
    }
    steps[i]=x
  }
  return(steps)
}

#Simulate X~binom(.25, 100)
X=rbinom(1, 100, p=.25)
goodPrior = betaWalk(X, .25, n=5000, a=10, b=30)
goodMean = mean(goodPrior)
goodMedian = median(goodPrior)
goodCredInt = quantile(goodPrior, probs = c(.025, .975))
print(paste("With good prior: Posterior Mean = ", round(goodMean,3), " Posterior Median = ", round(goodM
```

```
## [1] "With good prior: Posterior Mean =  0.23  Posterior Median =  0.231  95% Credible Interval = ( 0
```

```
plot(goodPrior, col="red", lwd=1, main = "Chain with good prior", xlab = "Step", ylab = "Value")
```

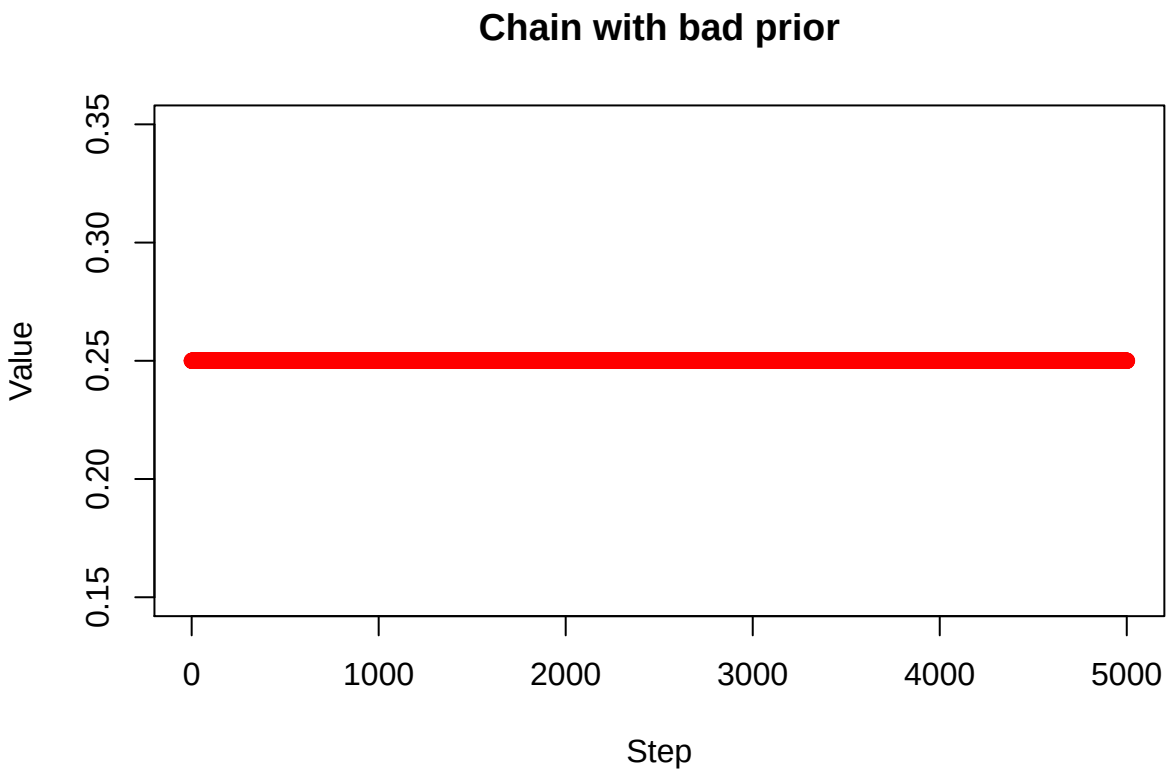


We see that this chain with a good prior mixes well as expected. We also get a credible interval that contains the actual $p = 0.25$ and a good estimate of the mean and median.

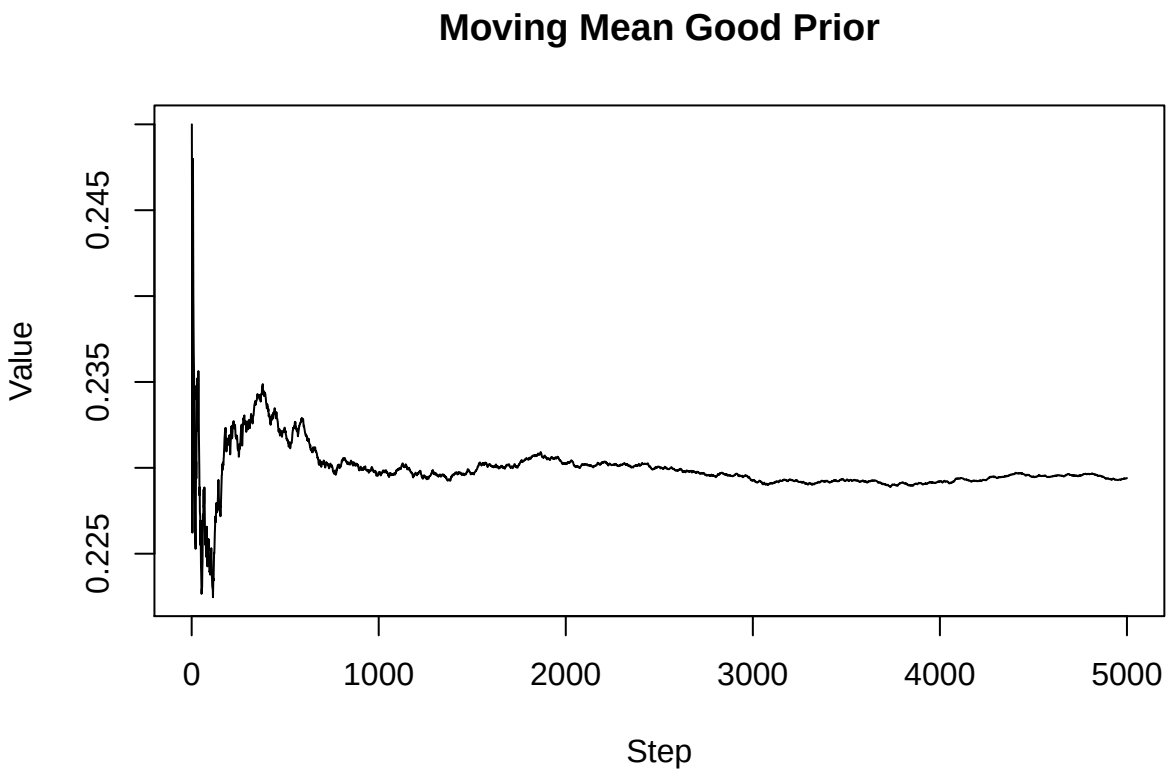
```
badPrior = betaWalk(X, 0.25, n=5000, a=30, b=10)
badMean = mean(badPrior)
badMedian = median(badPrior)
badCredInt = quantile(badPrior, probs = c(0.025, 0.975))
print(paste("With bad prior: Posterior Mean = ", round(badMean,3), " Posterior Median = ", round(badMedi
```

```
## [1] "With bad prior: Posterior Mean =  0.25  Posterior Median =  0.25  95% Credible Interval = ( 0.25
```

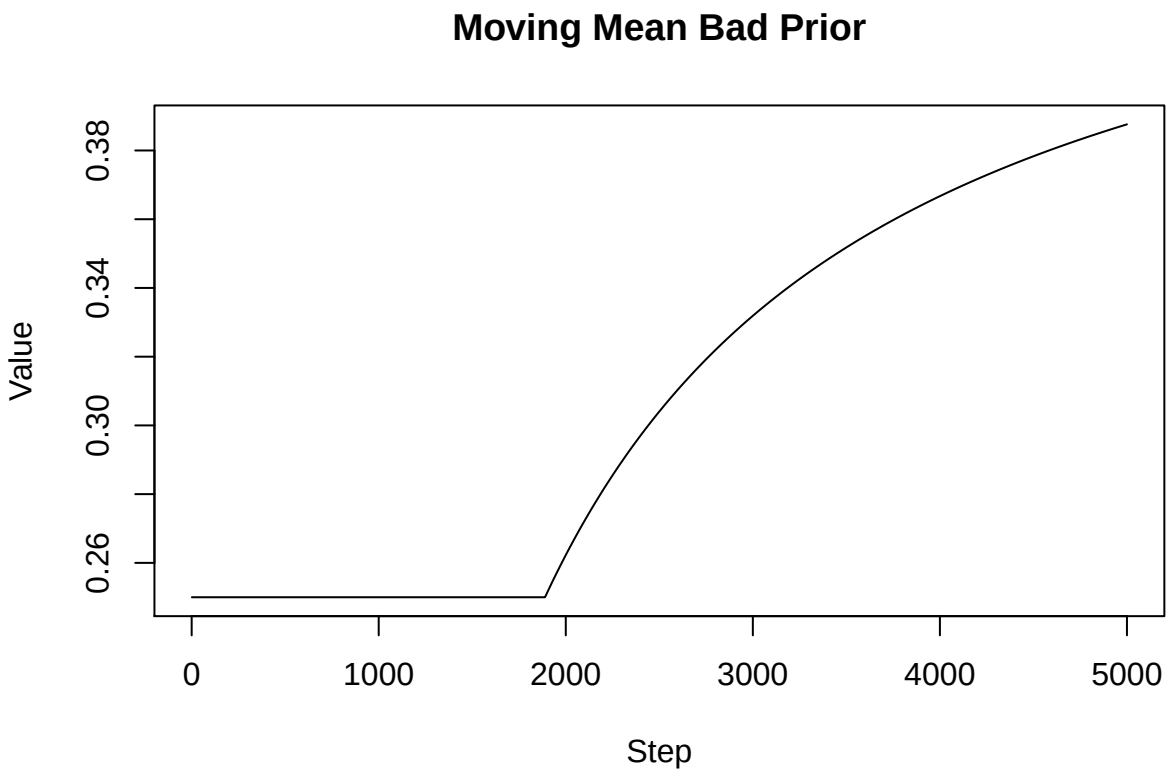
```
plot(badPrior, col="red", lwd=1, main = "Chain with bad prior", xlab = "Step", ylab = "Value")
```



The chain with a bad prior mixes terribly. It gets stuck at points because it is not offered values with high probabilty often. As a result, we get a bad interval and bad estimates of the mean and median.



Here we se that the estimate of the posterior mean with a good prior settles to ≈ 0.25 after 1000 iterations.


Moving Mean Bad Prior
Value
0.26
0.30
0.34
0.38
0
1000
2000
3000
4000
5000
Step