Lab 4

Senan Hogan-H.

5 April 2018

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

Question 1.

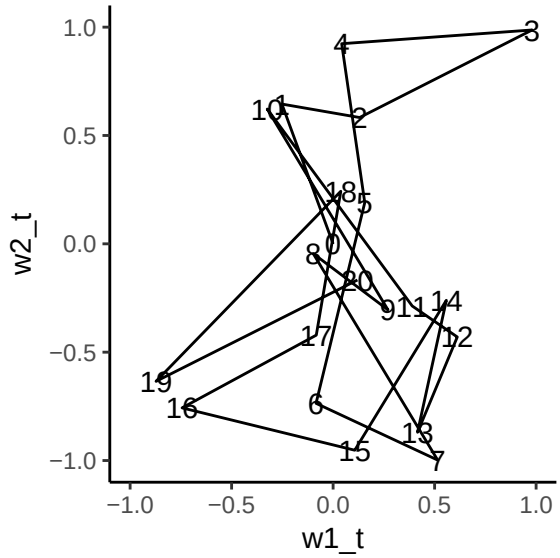
a. Define a random walk with no use of the MH ratio.

```
# 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){
      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){
      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)
```

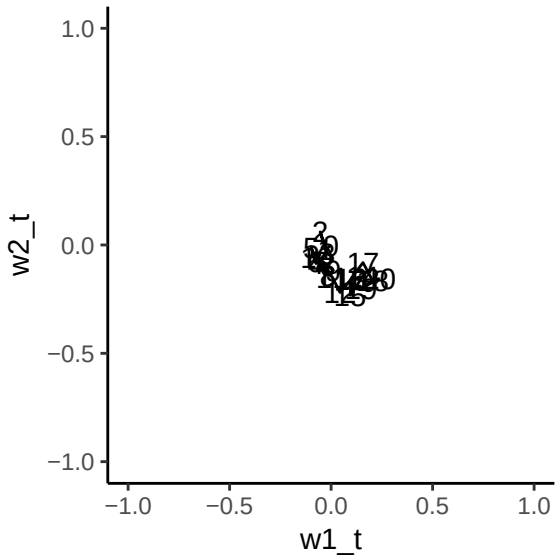
```
}

# 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))
  return(graph)
}
```

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.

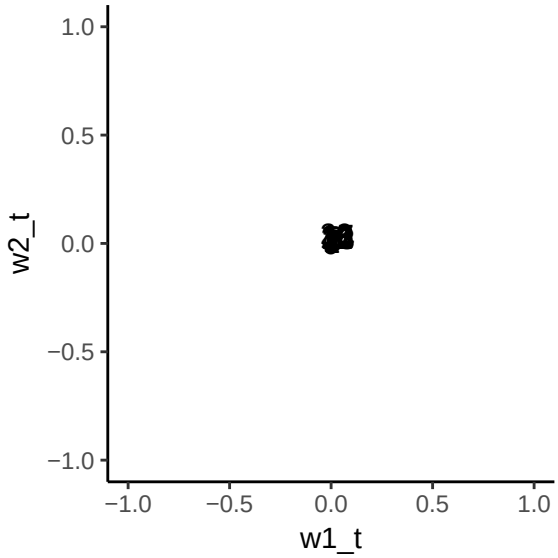
```
data <- random_walker(0, 0, 1 , 20)

random_walker_graph(data)
```



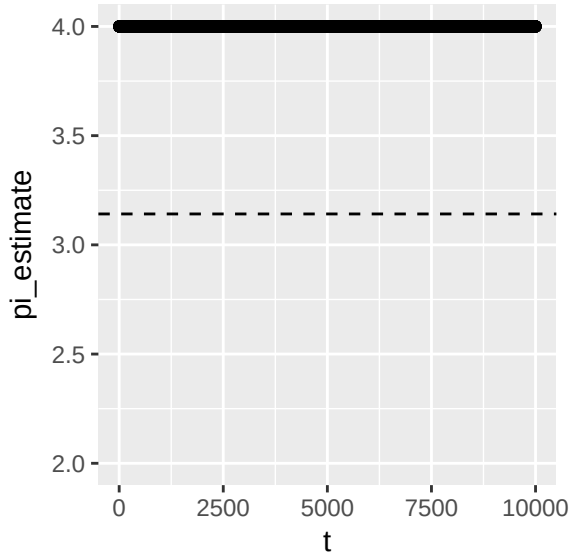

```
data <- random_walker(0, 0, 0.1 , 20)

random_walker_graph(data)
```




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

random_walker_graph(data)
```

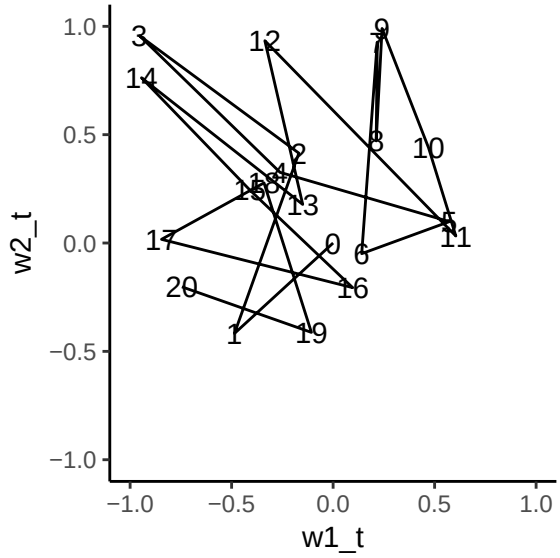



```
data <- random_walker(0, 0, 0.001 , 10000)

random_walker_pi(data)
```

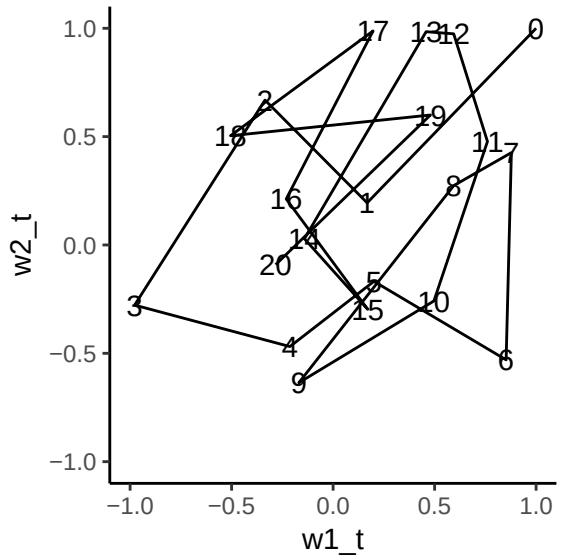


ii. Choice of w_0. w_0 generally gets washed by a long enough chain, though this does require a large enough value of epsilon. So that choice of w_0 does not theoretically matter, and different values of w_0 gives very similar estimates of pi.

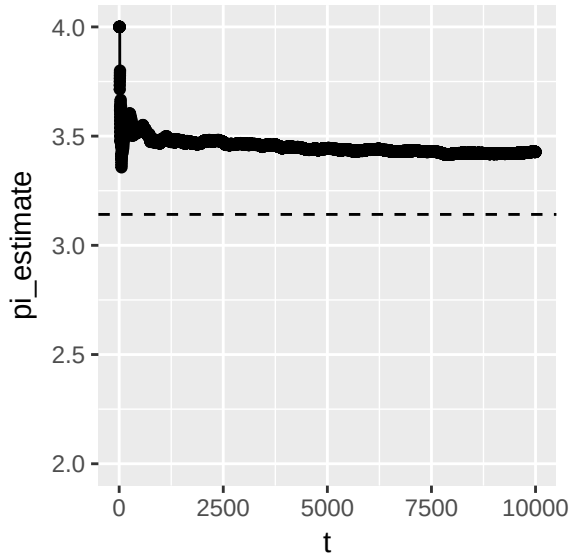
```
data <- random_walker(0, 0, 1 , 20)

random_walker_graph(data)
```



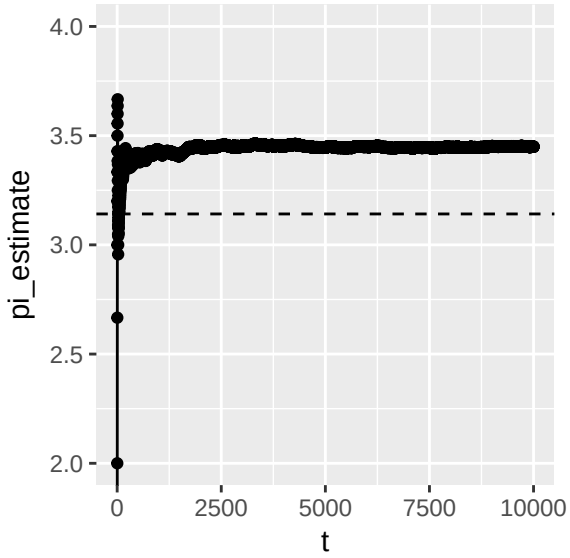

```
data <- random_walker(1, 1, 1 , 20)

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 variables estimates, which makes sense given the law of of large numbers.

iv. The above gives overestimates of π regardless of parameters. The chain must be updated to take account of the M-H ratio, given by:

$$r(x, y) = \min\{1, \frac{f(y)g(x|y)}{f(x)g(y|x)}\}$$

```
# 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)
    while (w1_move > 1 |  w1_move < -1){
      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){
      error2 <- runif(1, -epsilon , epsilon)
      w2_move <- w2_t[i] + runif(1, -epsilon , epsilon)
    }
    u <- runif(1) # generate a probability in (0,1)

    # compute MH ratio
    r <- 1 # dnorm(y, 3, 1)/dnorm(x[i-1], 3, 1)*dnorm(x[i-1])/dnorm(y)
```

```
    if (u < r){
      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))
  return(graph)
}
```

Question 2.

Good prior and bad prior.