Tarea 6: Ejercicios 6.7, 6.15, 7.9 y 7.12

Santiago Robatto, Sofía Terra

Exercise 6.7 (Normal-Normal grid approximation)

Parte A:

Como primer paso definimos la grilla de valores posibles de μ, entre 5 y 15 by 1, dado que es el recorrido indicado en la letra.

```
 [1]  5  6  7  8  9 10 11 12 13 14 15
```

Para cada valor de μ evaluaremos el prior y la verosimilitud.

Para el prior utilizaremos el dato de letra que $\mu \sim \mathcal{N}(10, 1.2^2)$ y evaluaremos eso.

A su vez, creamos un vector con el valor de las observaciones $(Y_1, Y_2, Y_3, Y_4) = (7.1, 8.9, 8.4, 8.6)$ para poder hallar la verosimilitud.

```
   mu_grid        prior   likelihood
1        5 5.646917e-05 1.889473e-08
2        6 1.285232e-03 1.267972e-05
3        7 1.460692e-02 7.979256e-04
4        8 8.289762e-02 4.708682e-03
5        9 2.349266e-01 2.605676e-03
6       10 3.324519e-01 1.352152e-04
7       11 2.349266e-01 6.579829e-07
8       12 8.289762e-02 3.002533e-10
9       13 1.460692e-02 1.284828e-14
10      14 1.285232e-03 5.155686e-20
11      15 5.646917e-05 1.940045e-26
```

Finalmente ya podemos aproximar el posterior, haciendo el producto de likelihood x prior y dividiendo entre su suma. Hacer el cociente lo que hace es asegurarnos de obtener una probabilidad, dado que estamos normalizando la funcion.

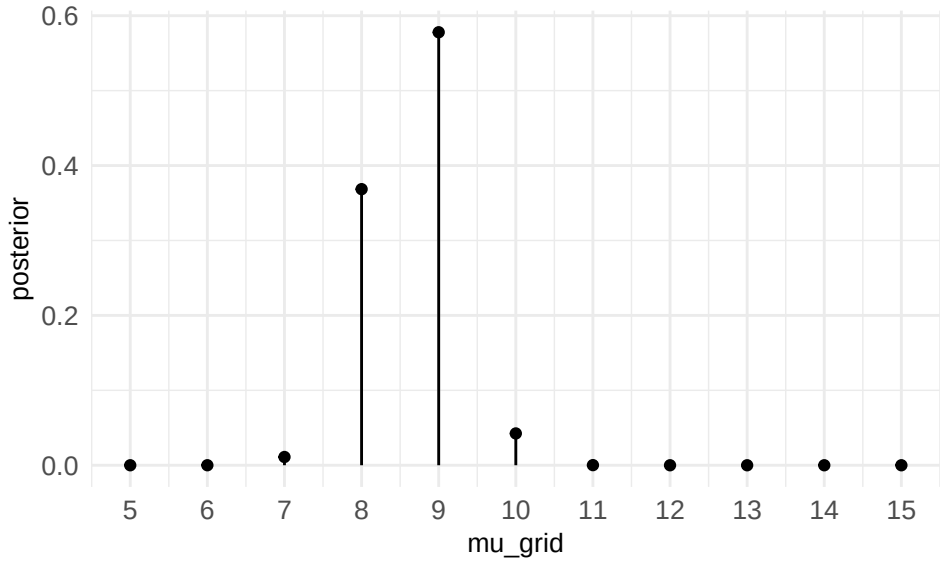
$$p(\mu_j \mid y) = \frac{p(y \mid \mu_j)\, p(\mu_j)}{\sum_k p(y \mid \mu_k)\, p(\mu_k)}$$

	mu_grid	prior	likelihood	unnormalized	posterior
1	5	5.646917e-05	1.889473e-08	1.066970e-12	1.007279e-09
2	6	1.285232e-03	1.267972e-05	1.629638e-08	1.538469e-05
3	7	1.460692e-02	7.979256e-04	1.165523e-05	1.100319e-02
4	8	8.289762e-02	4.708682e-03	3.903385e-04	3.685013e-01
5	9	2.349266e-01	2.605676e-03	6.121424e-04	5.778965e-01
6	10	3.324519e-01	1.352152e-04	4.495254e-05	4.243770e-02
7	11	2.349266e-01	6.579829e-07	1.545777e-07	1.459299e-04
8	12	8.289762e-02	3.002533e-10	2.489028e-11	2.349781e-08
9	13	1.460692e-02	1.284828e-14	1.876738e-16	1.771745e-13
10	14	1.285232e-03	5.155686e-20	6.626255e-23	6.255553e-20
11	15	5.646917e-05	1.940045e-26	1.095528e-30	1.034239e-27

El siguiente paso es solo un checkeo que haya funcionado bien todo. El hecho de que la suma de la variable posterior sume 1 nos asegura que esta efectivamente es una probabilidad.

	sum(unnormalized)	sum(posterior)
1	0.00105926	1

Graficamos el posterior obtenido en grid data, el cual nos devolvera los valores mas probables de μ. De esta manera, a partir de un priori normal (continuo) y 4 observaciones, generamos una distribucion a posteriori discreta para μ. Los valores mas probables son 8 y 9, pero al usar solo 10 valores para modelar μ perdimos mucha precision. Es por ello que repetiremos el experimento en la parte b, pero usando 201 valores posibles para μ.



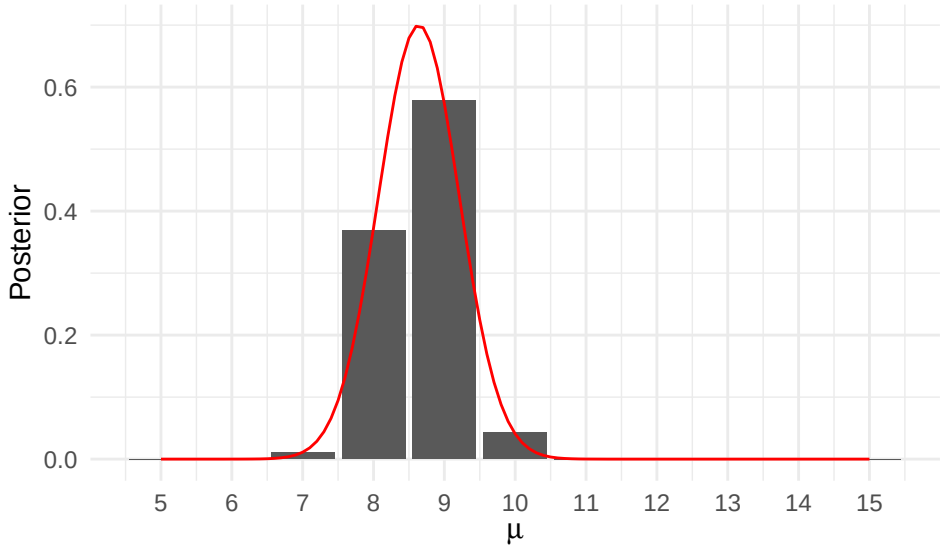
Ademas, podemos samplear a partir de lo obtenido anteriormente y compararlo contra el posterior real:

```
      model     mean     mode       var        sd
1     prior 10.00000 10.00000 1.4400000 1.2000000
2 posterior  8.64698  8.64698 0.3266577 0.5715398
```

Por ende sabemos que el posterior teorico del ejemplo distribuye $\mu \sim \mathcal{N}(8.64698, 0.571^2)$

Para el sampleo:

```
 mu_grid     n percent
       7   112  0.0112
       8  3687  0.3687
       9  5770  0.5770
      10   430  0.0430
      11     1  0.0001
   Total 10000  1.0000
```



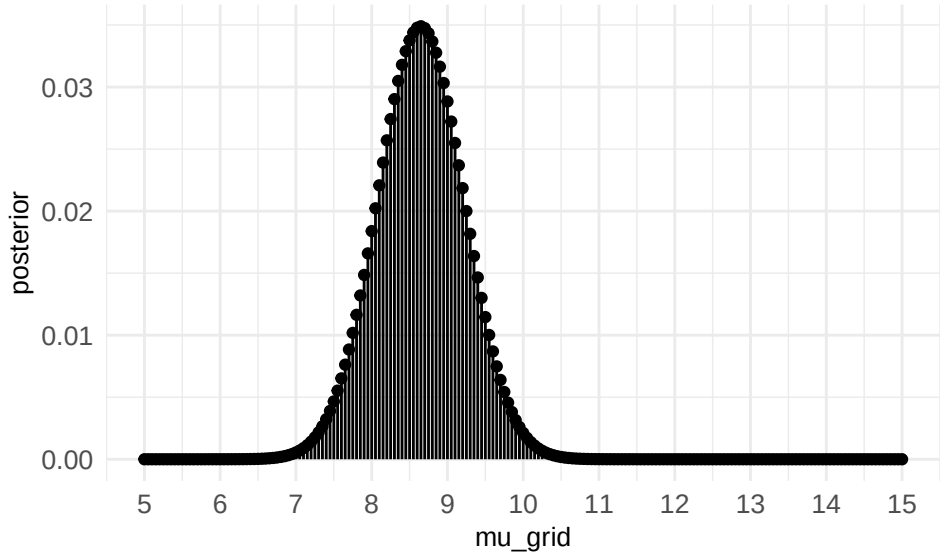
De esta manera vemos que el histograma de aproximacion por grilla es sumamente similar al posterior real.

Sin embargo, como ya vimos antes, al haber elegido pocos valores posibles para μ la aproximacion no es precisa.

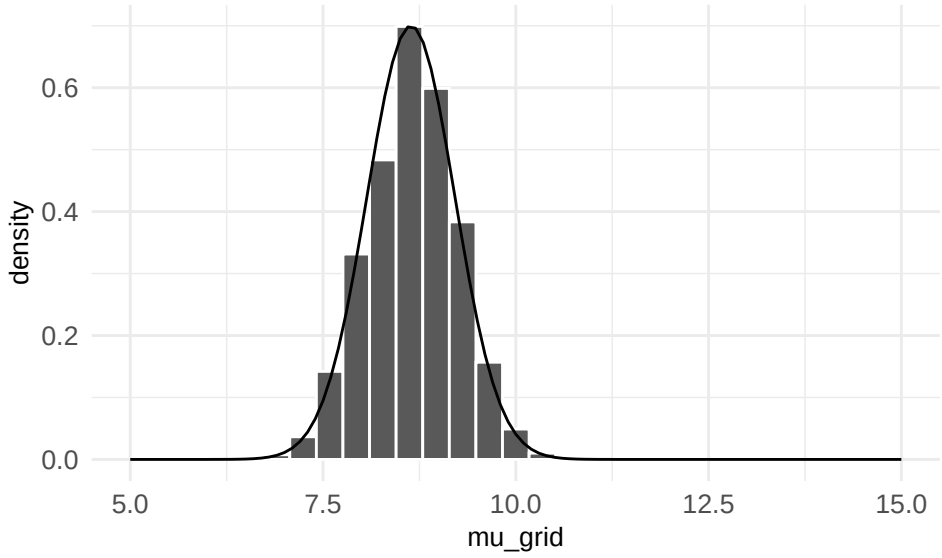
Parte B:

Replicaremos el codigo realizado en la parte anterior, con la unica diferencia que la secuencia para valores de μ sera de a 0.05 en vez de a 1.

Ahora, al haber graficado para 201 casos, la aproximacion por grilla se aproxima de manera casi perfecta a la normal del posteriori que calculamos teoricamente. Al tener la variable discreta, pero ser la diferencia entre valores muy pequena, se aproxima en mayor proporcion a una normal.



El muestreo a partir del posterior discreto produce datos cuya distribución prácticamente coincide con la densidad normal teórica del posterior, lo que sugiere que nuestra aproximación por grilla es adecuada para modelarˇ



Exercise 6.15 (MCMC with RStan: Gamma-Poisson)

El primer paso que debemos hacer es definir el modelo. Para ello utilizaremos la estructura base de stan: definir el recorrido de los datos, los parametros y por ultimo el modelo. Para definir los datos utilizaremos Y[3], que representa el vector de observaciones Y.

En los parametros usaremos lambda y lo definiremos en $\mathbb{R}^+$, dado que es el recorrido de cualquier distribucion poisson. Finalmente definimos la distribucion de Y y lambda, con el modelo gamma-poisson. Sabemos que los datos siguen una distribucion Poisson (λ) y que λ~Gamma(20,5) La definicion del modelo por ende es la siguiente:

```
gp_model <- "
  data {
    array[3] int<lower=0> Y;
  }
  parameters {
    real<lower = 0 > lambda;
  }
  model {
    Y ~ poisson (lambda);
    lambda ~ gamma (20,5);
  }
"
```

El siguiente paso es utilizar la funcion stan para simular las cadenas:

Esto no esta corriendo y ni se porque

Para hacer la traza de las cadenas utilizaremos la funcion mcmc_trace, que viene ya inclida en el paquete:

Exercise 7.9: One iteration with a Uniform proposal model

Utilizaremos un algoritmo de Metropolis Hastings para realizar la iteracion. En este caso particular, dado que tomaremos un proposal simetrico, el algoritmo tiene otro nombre y se denomina Metropolis alghortim.

Este algorito tiene propiedades que simplifican las reglas de aceptacion de moverse o no a la nueva locacion propuesta de μ.

Tras analizar tal como se hace en el libro (reescribiendo y desestimando la constante de normalizacion), la regla de decision se simplifica a comparar si el posterior evaluado en μ es mayor que en μ'.

$$\alpha = \min\left\{1, \frac{f(\mu' \mid y)}{f(\mu \mid y)}\right\}.$$

Cuando $f(\mu' \mid y) > f(\mu \mid y)$ entonces el minimo es 1 y nos movemos a la nueva locacion.

Cuando $f(\mu' \mid y) < f(\mu \mid y)$ podriamos aceptarlo o no, y eso lo desarrollaremos en el algoritmo con un sorteo aleatorio con probabilidad α de aceptacion y su opuesto de rechazar.

Parte a: W=0.01

```
  proposal    alpha next_stop
1  2.99531 0.987027   2.99531
```

En primer lugar, como w es muy pequeno (0.01), los valores del proposal estan entre 2.99 y 3.01. El nuevo proposal pertenece al intervalo, lo que es correcto.

α, la probabilidad de aceptacion del nuevo proposal, es casi 1. Por ende muy probablemente aceptaremos el valor propuesto. Es la probabilidad que le cargaremos al sorteo aleatorio.

Como era muy probable, aceptamos la nueva locacion para el proposal propuesto. Por ende next_stop es igual al proposal.

Con un w tan chiquito casi siempre aceptaremos el proposal, pero a cambio daremos pasos muy pequenos, recorreremos el recorrido de manera muy lenta.

Parte B: W=0.5

```
  proposal alpha next_stop
1 3.072853     1  3.072853
```

El proposal pertenece al intervalo determinado por current=3 $\pm$ w=

En este caso, se acepto automaticamente, por lo que sabemos que $f(\mu' \mid y) > f(\mu \mid y)$. Como α es exactamente igual a 1 no hubo sorteo.Esto significa que el proposal mejora la plausibilidad respecto al valor anterior. Por ende la cadena se movio automaticamente al nuevo valor.

Parte c: w=1

```
  proposal     alpha next_stop
1 2.403364 0.1162826  2.403364
```

En este caso el valor propuesto salta a 2.897 y fue obtenido dentro del intervalo [2,4] En este caso, el valor es menos plausible que el actual, por lo que α es menor a 1 y se debe sortear. Se realiza dicho sorteo con probabilidad α=0.7402203 y fue aceptado, por lo que el valor de la cadena se actualiza a $'$=2.897. En las pruebas que realizamos en clase, vimos que para este current, w=1 es un buen valor, que tiene un equilibrio dentro de estabilidad y avance en el recorrido. Lo utilizaremos para comparar en el siguiente ejercicio.

Parte D: w=3.

```
  proposal      alpha next_stop
1 5.668052 0.08411669         3
```

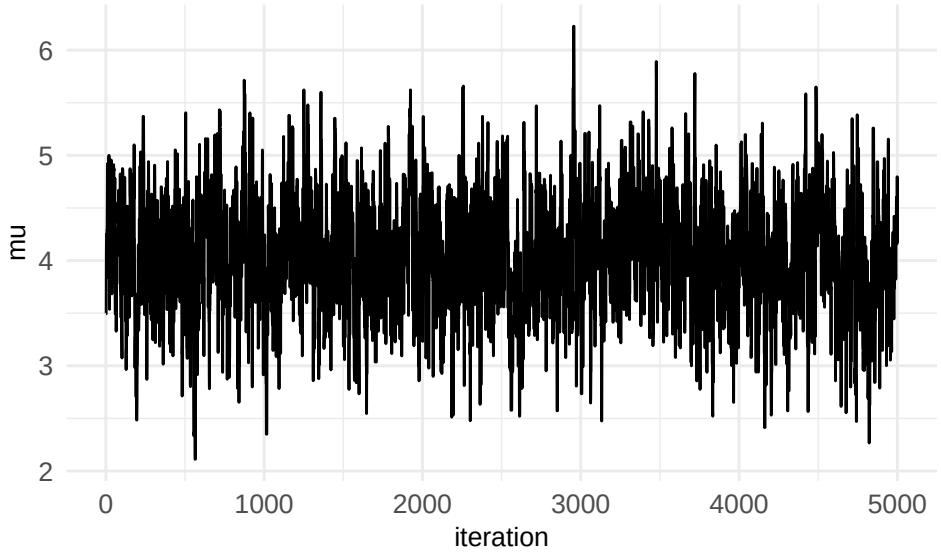
En este caso el valor del proposal es 4.0474 y pertenece al intervalo 3 ± 3, es decir [0,6]. En este caso α=1, lo que significa que la plausibilidad (posterior en el nuevo punto) fue mayor o igual a la del punto actual(3). En este caso, la probabilidad de aceptación es $1 \rightarrow$ siempre aceptamos la propuesta.

Ejercicio 7.10: (An entire tour with a Uniform proposal model)

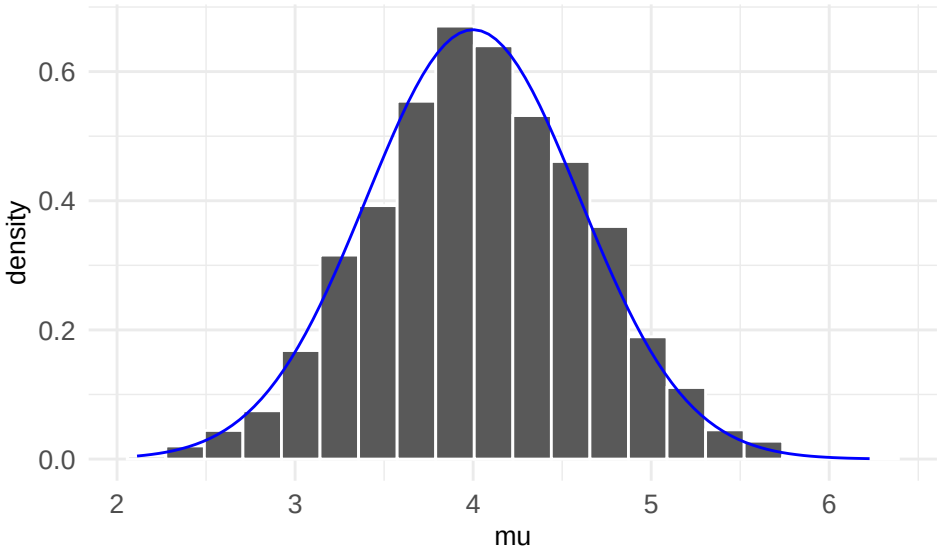
Modificaremos la funcion mh_tour de manera de hacerla mas universal. Para ello le pasaremos el valor de current al llamar la funcion y no dentro de ella, como se realizo en el ejercico en clase. De este modo, cada vez que llamemos a current debemos pasarle 3 datos: N=cantidad de iteraciones, w=ancho del intervalo para la uniforme y current=valor inicial.

Por ejemplo, tal como vimos en clase, un buen valor para realizar las iteraciones es w=1 y un n suficientemente grande, por ejemplo n=5000, dado que la traza recorre los distintos valores plausibles para μ, pero con mayor probabilidad los cercanos a 3, y al graficar el histograma nos aproximamos mucho a una normal.

De esta manera, el gráfico de traza mantendrá su mayor concentracion de ruido alrededor del μ teorico del posterior (4), oscilando y recorriendo distintos valores . Se observa que en ningun momento se estanca (no genera lineas constantes) y tanto para arriba como para abajo recorre el recorrido de μs posibles. Ademas aparenta ser estable, no tiene patron a crecer ni a achicarse. Tampoco se observan valores demasiado alejados ni saltos descontrolados.



Esta simulacion generara una muy similar a la normal calculada de manera teorica, por lo qjue decimos que el algoritmo ajusta bien a la posterior que queriamos calcular. Por lo tanto, la simulación MCMC reproduce correctamente la forma de la distribución teórica.

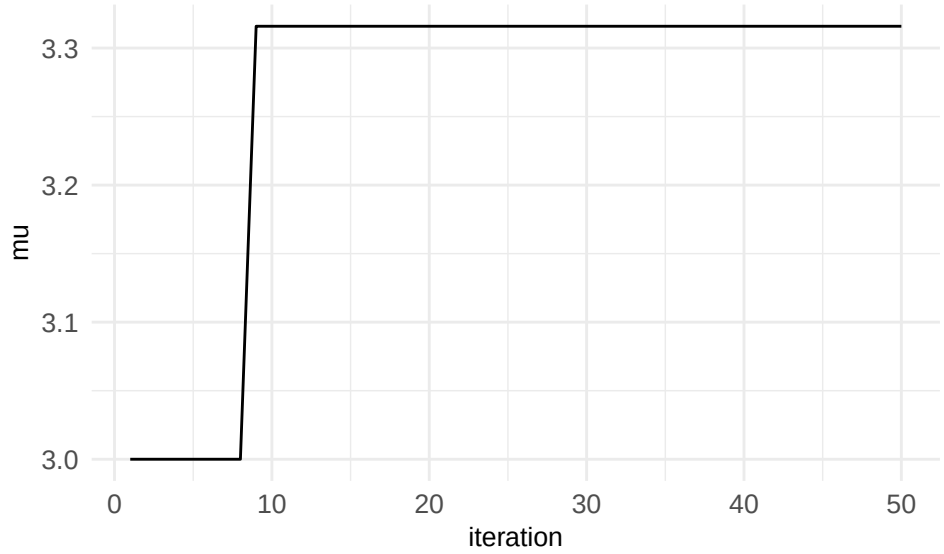


Parte A: 50 iterations, W=50.

Al iterar 50 veces con un w tan grande, los valores posibles para la uniforme perteneceran a [-47,53]. Este intevalo es sumamente amplio si lo comparamos con la normal que estamos intentando muestrear. Esto generara que los valores de proposal esten muy dispersos respecto

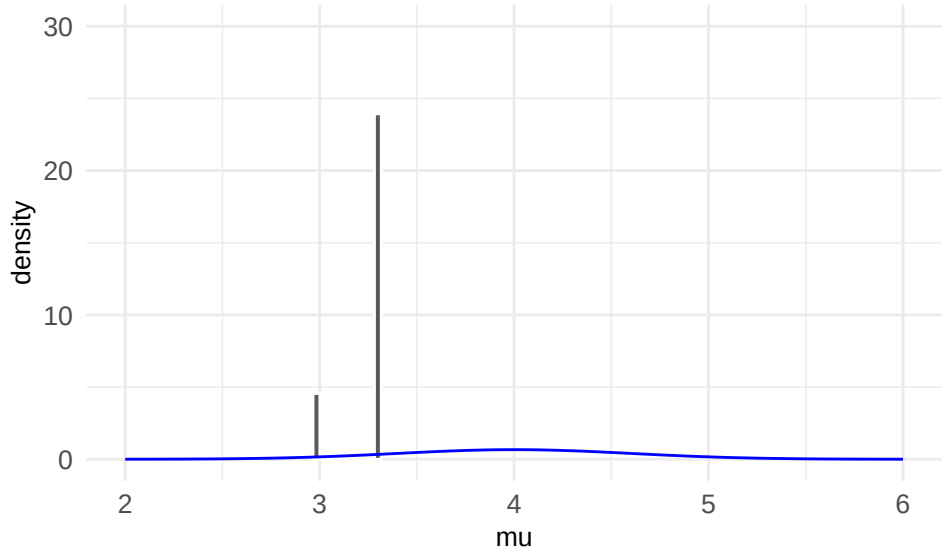
a 3, por ende al evaluar la funcion en proposal tendera a cero, haciendo muy probable que los α de cada iteracion sean muy pequenos y por ende la probabilidad de hacer el salto en el sorteo sea muy baja. Por ende, seria esperable ver pocos saltos de valor, es decir que nos costara salir del current inicial y a su vez nos costara salir del nuevo valor que tomemos.

Respecto al histograma, es de esperar tener pocos valores con muchas observaciones cada uno.



El grafico de la traza muestra que se rechaza para los 8 primeros valores, es decir que los valores fueron menos plausibles y el sorteo fue negativo. En el caso del valor 9 lo acepta. Finalmente sigue rechazando durante toda la iteracion, habiendo quedado fijo en aproximadamente 3.3.

Esto generara un histograma con dos barras, dado que la cadena tuvo solo dos valores: 3 (inicial) y 3.3, que fue el aceptado en el caso numero 9.

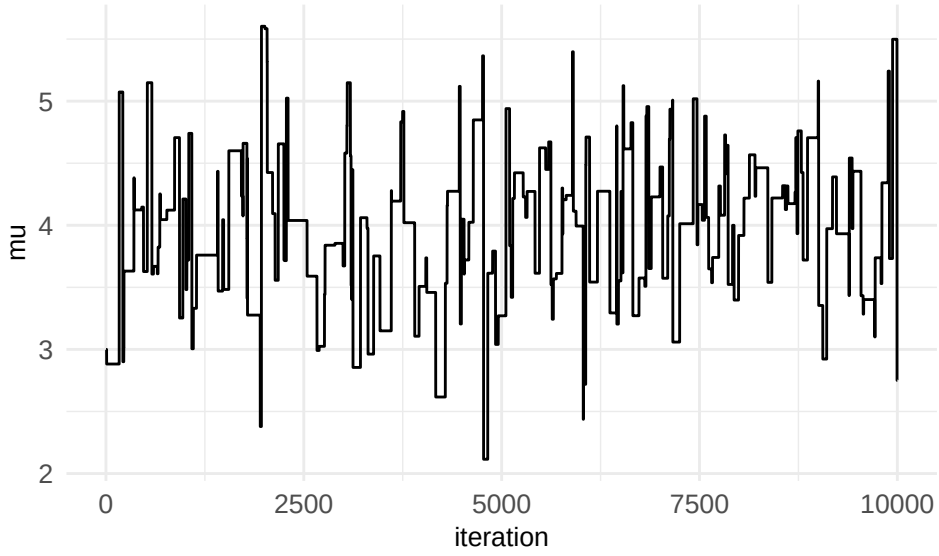


Si bien el grafico a primera vista parece incorrecto, se ve raro por una cuestion de escala. Se observa correctamente graficada la normal que representa el posterior teorico. A su vez, se generaron las dos columnas mencionadas antereiormente en el historgrama, las cuales son sumamente altas dado que se repitio mucas veces el 3 y el 3.3. Es sumamente evidente que con n=50 y w=50, no se ajusta bien.

Parte c: Que pasa si agrandamos a n=10.000?

El grafico de traza en esta oportunidad:

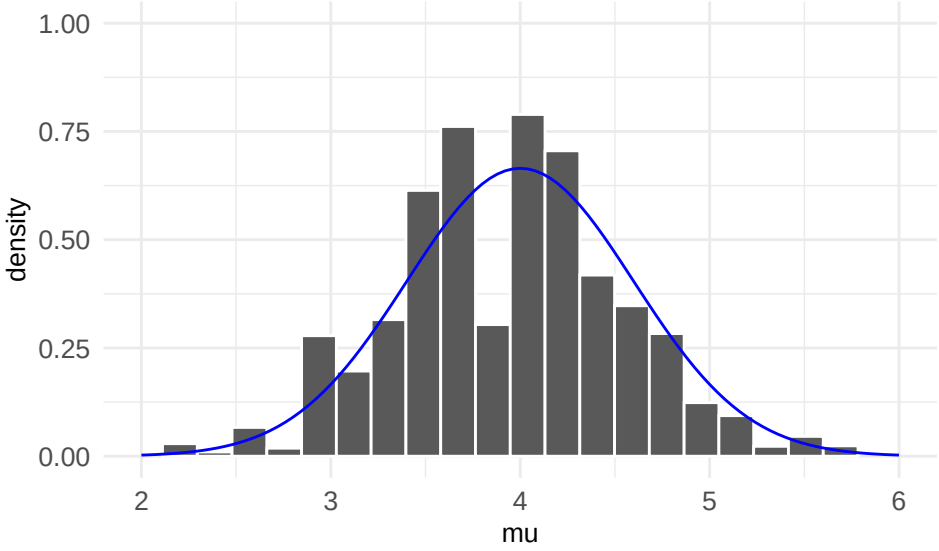
- Muestra muchas mesestas, lo que nos hace pensar en que se estanco varias veces en muchos valores. Un w tan grande genera que se tranque mucho la cadena, por el tema del sorteo, que ya mencionamos anteriormente,
- Muestra varios saltos de valores bajos a altos y viceversa, lo que nos hace pensar que en alguno de los sorteos tuvimos exito a pesar de tener baja probabilidad de que esto ocurra.
- Por lo tanto, a pesar de tener un w tan proporcionalmente grande, con $n = 10000$ logramos compensar parte del efecto de los estancamientos para poder ver todo el recorrido.
- Probablemente genere un histograma sesgado, o con columnas mas altas de lo que deberian en algunos valores.



En el histograma se observa que, a pesar de no usar el w ideal, logramos mejorar bastante la aproximacion a la normal. Si bien a la aproximacion le falta mucho para ser buena, empieza a verse reflejada la forma.

Los valores mas extremos, empieza a verse reflejados de buena manera, acompasando la curva de la normal. En el centro se observa una serie de valores entorno a 4 que no lograron tener buena densidad.

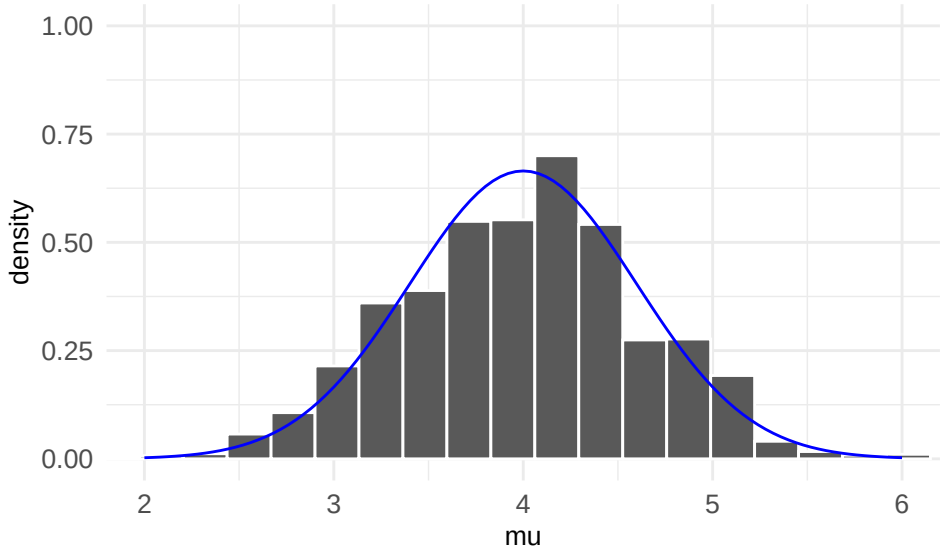
De este modo, podemos pensar que agrandando suficientemente la N, logramos compensar gran parte del efecto de haber elegido un w tan grande.



Si volvemos a repetir la simulacion, pero esta vez con n=50.000, logramos aproximar de mucha mejor manera la normal teorica del posterior.

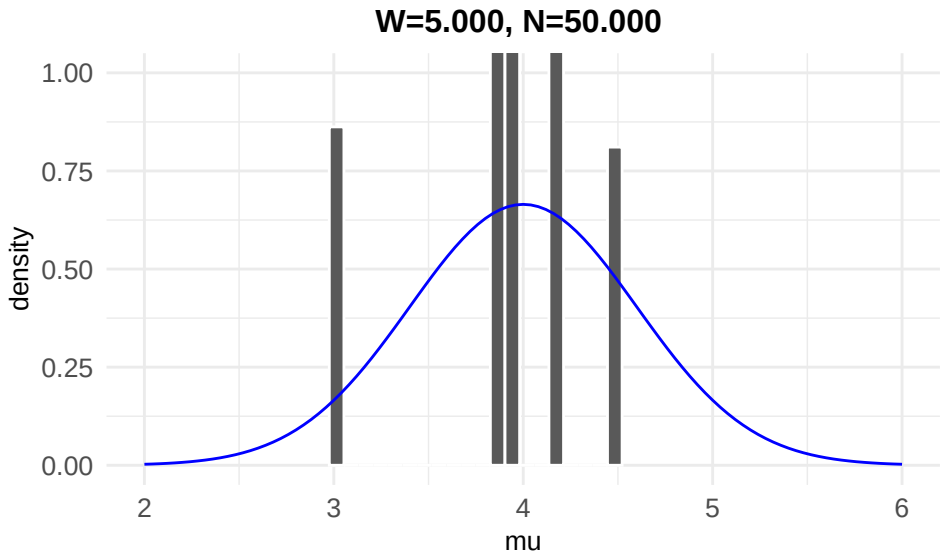
Aun no llega a ser perfecta, pero cada vez es mejor.

Esto nos da lugar a pensar que era cierta nuestra teoria de que con un n suficientemente grande, compensamos el efecto de un mal w.



Computacionalmente podemos verlo como que estamos sacrificando eficiencia a cambio de haber elegido mal los parametros. Se muestra que con un tamaño de muestra suficientemente grande podemos compensar un tuning ineficiente, aunque a costa de un uso mucho mayor de recursos computacionales. Se intento volver a reproducir con n=500.000, pero la computadora no tuvo capacidad de soportarlo.

Por lo tanto, no todo tuning ineficiente sera compensable.



Como prueba de que no todo tuning eficiente sera compensable, si hubieramos elegido un w considerablemente peor que 50, por ejemplo 5000, el efecto de n no alcanza para arreglarlo. La grafica se parece al ejemplo inicial de w=50 y n=50, donde le cuesta muchisimo salir de cada valor. Necesitariamos un n inmensamente superior para compensarlo, lo que computacionalmente no siemrpe sera viable.

De esta manera, concluimos que es mas eficaz mejorar el w antes que seguir agrandando la simulacion.