Trabajo

Francisco Herrera Barajas

2025-05-08

1. Introducción

La estadística bayesiana ha sido utilizada con gran éxito en la búsqueda de personas u objetos desaparecidos. Algunos ejemplos son su uso en desastres aéreos o marítimos, así como para personas desaparecidas en alta montaña. Un caso concreto es el vuelo MH370 de Malaysian Airlines, en el cual se usaron modelos basados en el enfoque bayesiano.

Una de las ventajas que tiene usar la estadística bayesiana en estos escenarios es que permite sistematizar el proceso de búsqueda. Podemos fijar probabilidades a priori sobre las localizaciones del objeto desaparecido y, con las evidencias que vamos recogiendo, ir actualizando dichas probabilidades. Esto lleva a una búsqueda que va guiada por los datos y nos puede servir de herramienta para organizar la búsqueda.

En este trabajo voy a simular una búsqueda bayesiana iterativa de estas características. Para ello, primero nos vamos a poner en situación: imaginemos que un barco que llevaba una preciada mercancía se ha hundido en mitad del océano Atlántico. Sabemos de forma aproximada la localización de la última comunicación que se tuvo con el barco. Esta ya nos permite empezar a buscar en una zona determinada. Pese a ello, esta zona es un cuadrado de 100x100 km, demasiado grande para buscar de forma exhaustiva kilómetro por kilómetro. Sin embargo, para nuestra suerte, la mercancía emite una señal. Dicha señal la podemos captar y medir su intensidad. Pese al ruido, esta nos puede indicar la ubicación de la mercancía.

Tras esta breve narración, vamos a pasar a ver el modelo de forma detallada.

2. Modelo

El modelo se basa en el teorema de Bayes, que permite actualizar las probabilidades sobre la localización del objeto a medida que se recogen nuevas observaciones. Cada observación aporta evidencia que, combinada con la información previa (la distribución a priori), genera una distribución posterior que refleja el conocimiento actualizado. A medida que el modelo acumula más observaciones, la posterior se va concentrando en las regiones más compatibles con los datos. En este caso debido a las limitaciones de rstan la prior siempre va a ser una uniforme que se va a ir acotando.

Los supuestos básicos del modelo son:

1) Buscamos un objeto único que se encuentra estático en una posición determinada.
2) El objeto se encuentra en un plano 100x100 casillas,con coordenadas x e y.
3) El objeto emite una señal con una intensidad determinada la cual se puede medir en cada observación.
4) Cada observación es independiente de las demás.
5) Al modelo le basta con que estos puntos observados estén en este rango, $x_{\text{obs}} \in [x-3,\ x+3]$, $\quad y_{\text{obs}} \in [y-3,\ y+3]$, para que puedan encontrar el objeto.

6) El modelo no puede repetir la misma combinación de coordenadas x e y.

7) Los puntos de observación iniciales son bastante determinantes para el modelo, por lo que todas las simulaciones tienen la misma combinación de cinco puntos iniciales. Solo después de que se metan estas cinco observaciones se empieza a modificar la prior.

El modelo en términos matemáticos:

$$f(x, y \mid I) \propto f(x, y) \cdot f(I \mid x, y)$$

-$f(x, y)$: La distribución a priori, Asumimos que tanto x como y siguen una distribución uniforme en el intervalo (0,100).

-$f(I \mid x, y)$: Verosimilitud de los puntos (x, y) dada la intensidad observada. Sigue una distribución normal $I_i \sim \mathcal{N}(\mu_i, \sigma^2)$ (los parámetros que tiene los explicaré más adelante)

-$f(x, y \mid I)$: La distribución a posteriori. Se calcula multiplicando la función de verosimilitud por la distribución a priori. Posteriormente, discretizamos la distribución a posteriori para determinar un nuevo intervalo de búsqueda, sobre el cual definimos de nuevo una distribución a priori uniforme. De este modo, aunque la distribución posterior ya no es uniforme, en cada iteración volvemos a utilizar una prior uniforme, pero restringida al subespacio más probable según la evidencia acumulada.

Parámetros de la función de verosimilitud.

Si desarollamos más $f(I \mid x, y)$:

$$\prod_{i=1}^{n} \mathcal{N}\left(I_i \mid \mu_i(x, y), \sigma^2\right)$$

* Utilizamos el producto por que cada observación de la señal es indenpendiente de las demás.

Asimismo μ_i estaría dada por:

$$\mu_i = A \cdot \exp\left(-\frac{d_i}{\lambda}\right)$$

-A: Amplitud máxima de la señal.

-λ: Parámetro de decaimiento espacial, es decir cuanto se difumina la señal.

(Ambos parámetros son fijos y y arbitrarios en nuestro caso)

-d_i: Es la distancia euclídea entre los puntos de observación y donde se estima que esta verdaderamente el objeto. $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$

Finalmente,

σ^2: Representa la varianza del ruido asociada a la medición de la intensidad de la señal

3. Guión conceptual de como ejecutamos el modelo.

1) Fijamos parámetros: Las coordenadas x e y del objeto escondido, el parámetro A, el parámetro lambda y sigma.

2) Realizamos cinco observaciones en coordenadas concretas y calculamos la intensidad. El hecho de que sean estas cinco coordenadas para todas las simulaciones es a razón de que sobre todo para simulaciones con mucho ruido, estas suelen ser bastante determinantes.

3) A partir de las observaciones ya se podrá ir actualizando las distribuciones a priori. Estas siempre tendrán forma de uniformes solo acotaremos sus límites para ir centrándolas alrededor del objeto perdido, en nuestro caso el barco hundido

4) Simularemos el modelo Stan y extraeremos las muestras, estas las redondearemos para convertirlas discretas. Sé que esto añade ruido al modelo pero no se me ocurrían formas mejores de llevarlo a cabo. Ya que todo este modelo tiene un problema grave; el objeto está en un coordenada concreto pero las intesidades de la señal que emiten son valores continuos. Además que rstan solo nos permite poner distribuciones con forma conocida en su apartado parameters.

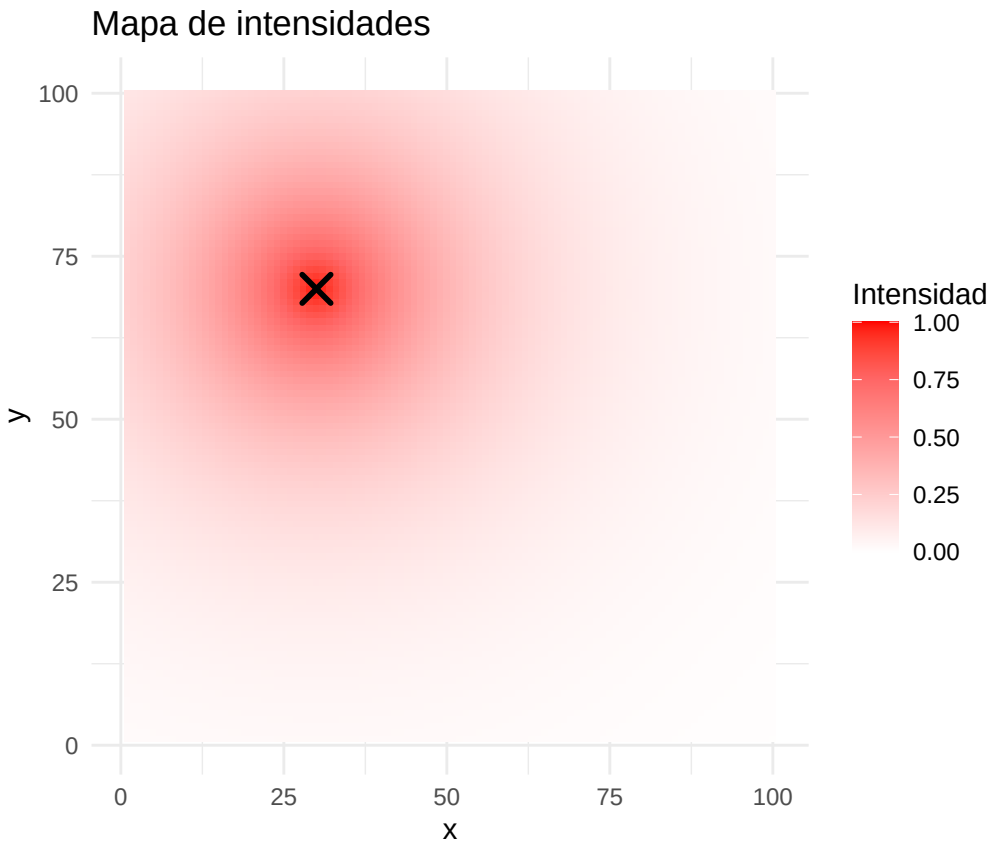
5) Luego crearíamos una matriz que realiza un conteo de cuántas veces aparece cada punto de la matriz, y la normalizaríamos para que dicha matriz contuviese probabilidades. Asimismo cogeraríamos los indices, estos nos indicarían coordenadas, de las celdas con mayor probabilidad y las reutilizaríamos para que sean los nuevos puntos de observación.

6) Posteriormente se calculan las nuevas intesidades a partir de los puntos extraídos.

7) Combinaríamos estos puntos con las muestras de rstan y redondeándolas nos servirán para establecer límites a las nuevas priors uniformes que use el modelo.

4.Consideraciones prévias.

Este sería el mapa de las intensidades que vamos a utilizar.



Una parte que me parece importante remarcar es que este modelo consigue el objetivo de encontrar el objeto o al menos acercarse por dos motivos:

1) A través del historial de los sucesivos puntos de observación y gracias al remuestreo que lleva a cabo rstan de estas evidencias, permite al modelo "encontrar coherencia entre el ruido" y aproximarse o encontrar el objeto. Tenemos que tener en cuenta que las señales no son para nada estables y pueden dar valores algo diferentes dependiendo del ruido.

2) Con las muestras de Rstan y los sucesivos puntos de información se realizan nuevas prior que acotan las coordenadas de las posibles ubicaciones del objeto.

Estas son las razones por las que el modelo consigue encontrar el objeto. De hecho, cabe la posibilidad de pensar que quizás solo siguiendo la intensidad de la señal, es decir, midiéndola constantemente y avanzando, en pro de mayores niveles de intensidad, se pueda encontrar el objeto. En cambio, esto no sucedería ya que el componente de ruido que tiene la señal hace que realmente no sea una señal estable y no se puede seguir sin más, como por ejemplo un rastro de unas pisadas en la arena.

5.Código base

En esta sección voy a presentar cada apartado de la función que voy a utilizar para realizar las simulaciones.

Parámetros básicos del modelo

```
# Localización verdadera del objeto
true_x <- 30
true_y <- 70

# Parámetros del modelo
A <- 1
lambda <- 20
sigma <- 0.2

# Número de observaciones
n_obs <- 25
```

Inicializamos vectores vacíos y una lista para ir metiendo los puntos de observación y las intensidades

```
x_points <- numeric(n_obs)
y_points <- numeric(n_obs)
intensities <- numeric(n_obs)
posterior_matrix <- matrix(0, 100, 100)
resultados <- list())
```

Generamos las cinco primeras observaciones y medimos sus intensidades correspondientes.

```
# Observaciones iniciales dirigidas
obs_iniciales <- data.frame(
    x = c(25, 75, 25, 75, 50),
    y = c(25, 25, 75, 75, 50)
)

n_init <- nrow(obs_iniciales)

x_points[1:n_init] <- obs_iniciales$x
y_points[1:n_init] <- obs_iniciales$y
```

```
for (j in 1:n_init) {
  dx <- x_points[j] - true_x
  dy <- y_points[j] - true_y
  dist_j <- sqrt(dx^2 + dy^2)
  mu_j <- A * exp(-dist_j / lambda)
  intensities[j] <- rnorm(1, mean = mu_j, sd = sigma)
  }
```

La 5 primeras iteraciones son con las 5 observaciones dirigidas, en realidad se hacen meten al modelo a la vez. Igualmente, la prior que se utiliza aqui no sufre niguna actualización, se usa la distribución original.

```
for (i in n_init:n_obs) {

  if (i == n_init) {
    x_min <- 0; x_max <- 100
    y_min <- 0; y_max <- 100
```

Pasadas estas cinco observaciones, se va restringiendo las nuevas priors uniformes a través de estos comandos. Creamos un conjunto con las muestras de Stan y con los puntos que ya hemos observado. A partir de dicho conjunto se propondrá nuevos límites para la prior. Asimismo, los valores continuos se discretizan redondeándolos para que concuerden con las coordenadas del plano.

```
else {
      x_comb <- c(posterior_samples$x_obj, x_points[1:i])
      y_comb <- c(posterior_samples$y_obj, y_points[1:i])
      x_min <- max(floor(min(x_comb)), 0)
      x_max <- min(ceiling(max(x_comb)), 100)
      y_min <- max(floor(min(y_comb)), 0)
      y_max <- min(ceiling(max(y_comb)), 100)

      if (x_min == x_max) { x_min <- max(x_min - 1, 0); x_max <- min(x_max + 1, 100) }
      if (y_min == y_max) { y_min <- max(y_min - 1, 0); y_max <- min(y_max + 1, 100) }
    }
```

El modelo en RStanm, actualizaremos las distribuciones a priori a través de sprintf(). En este punto me encontré con bastantes dificultades, ya que RStan solo admite funciones con formas predeterminadas. Al principio quería especificar directamente en RStan la distribución a posteriori que calcula el modelo, pero no fue posible hacerlo así dentro del bloque parameters.

```
stan_code <- sprintf("
      data {
        int<lower=1> n;
        vector[n] x_obs;
        vector[n] y_obs;
        real<lower=0> A;
        vector[n] intensity;
        real<lower=0> sigma;
        real<lower=0> lambda;
      }
      parameters {
        real<lower=%f, upper=%f> x_obj;
        real<lower=%f, upper=%f> y_obj;
```

```
  }
  model {
    for (i in 1:n) {
      real d = sqrt((x_obs[i] - x_obj)^2 + (y_obs[i] - y_obj)^2);
      real mu = A * exp(-d / lambda);
      intensity[i] ~ normal(mu, sigma);
    }
  }
", x_min, x_max, y_min, y_max)
```

Comandos de control, los datos con los que entrenamos el modelo y el entrenamiento en sí.

```
# Comandos de control
archivo_stan <- tempfile(fileext = ".stan")
writeLines(stan_code, archivo_stan)
modelo_dinamico <- cmdstan_model(archivo_stan)

# Datos
data_i <- list(
  n = i,
  A = A,
  x_obs = as.numeric(x_points[1:i]),
  y_obs = as.numeric(y_points[1:i]),
  intensity = as.numeric(intensities[1:i]),
  sigma = sigma,
  lambda = lambda
)

# Entrenamos el modelo
fit <- modelo_dinamico$sample(
  data = data_i,
  seed = 123,
  chains = 4,
  parallel_chains = 14,
  iter_warmup = 500,
  iter_sampling = 500,
  refresh = 0
)
```

Extraemos los datos del modelo entrenado, rendondeamos y nos aseguramos que el mínimo sea al menos un 1 y el máximo un 100.

```
posterior_samples <- fit$draws(variables = c("x_obj", "y_obj"), format = "df")
x_post <- pmin(pmax(round(posterior_samples$x_obj), 1), 100)
y_post <- pmin(pmax(round(posterior_samples$y_obj), 1), 100)
```

Cuenta cuántas veces cada celda (x, y) ha sido predicha como posible ubicación del objeto. Posterior matriz represetaría una especie de mapa de calor de las muestras generadas.

```
for (j in seq_along(x_post)) {
  posterior_matrix[y_post[j], x_post[j]] <- posterior_matrix[y_post[j], x_post[j]] + 1
}
```

Normaliza la matriz posterior para que contenga probabilidades. Asimismo, detecta el número máximo y extrae sus índices , estos representarían sus coordenadas en el plano. Estós maximos son las celdas con mayor probabilidad.

```
posterior_matrix_norm <- posterior_matrix / sum(posterior_matrix)
   max_idx <- which(posterior_matrix_norm == max(posterior_matrix_norm), arr.ind = TRUE)
```

Gráficos e impresiones.

```
plot_title <- sprintf("Iteración %d: Prior [%d,%d] x [%d,%d]", i, x_min, x_max, y_min, y_max)

    if (mostrar_plot) {
      print(
        ggplot(image_matrix, aes(x = x, y = y)) +
          geom_tile(aes(fill = prob)) +
          geom_point(data = data.frame(x = true_x, y = true_y),
                     mapping = aes(x = x, y = y),
                     color = "black", size = 3, shape = 4) +
          geom_point(data = data.frame(x = x_points[i], y = y_points[i]),
                     mapping = aes(x = x, y = y),
                     color = "blue", size = 3, shape = 4) +
          scale_fill_gradient(low = "white", high = "red") +
          coord_fixed() +
          labs(title = plot_title) +
          theme_minimal()
      )
    }

    cat("\nIteración:", i,
        "\nCoordenadas de observación:", x_points[i], y_points[i],
        "\nIntensidad observada =", round(intensities[i], 4),
        "\nCelda más probable =", max_idx[1, 2], max_idx[1, 1],
        "\nProbabilidad posterior  ", round(max(posterior_matrix_norm), 4),
            "\nPrior actual: x   [", x_min, ",", x_max, "], y   [", y_min, ",", y_max, "]\n"
```

En esta parte guardamos los datos.

```
if (guardar_datos) {
      resultados[[i]] <- list(
        image_matrix = image_matrix,
        title = plot_title,
        max_idx = max_idx,
        posterior_matrix = posterior_matrix,
        posterior_matrix_norm = posterior_matrix_norm
      )
      }
```

En esta parte de la función voy llevar a cabo un registro de la celdas que ya he visitado para no volverlas a visitar. He hecho esto porque en las continuas ejecuciones de las funciones veo que en ocasiones, sobre todo cuando el ruido es alto; valores σ altos, el modelo sé queda atrapado en realizar continuamente las mismas observaciones.

```
if (i < n_obs) {
      celdas_usadas <- paste(x_points[1:i], y_points[1:i], sep = "_")
      orden <- order(posterior_matrix_norm, decreasing = TRUE)
      posiciones <- arrayInd(orden, .dim = dim(posterior_matrix_norm))
      encontrado <- FALSE
      for (k in 1:nrow(posiciones)) {
        propuesta <- posiciones[k, ]
        celda_id <- paste(propuesta[2], propuesta[1], sep = "_")
        if (!(celda_id %in% celdas_usadas)) {
          x_points[i + 1] <- propuesta[2]
          y_points[i + 1] <- propuesta[1]
          encontrado <- TRUE
          break
        }
      }
      # Si no se encuentra celda nueva, se repite la más probable
      if (!encontrado) {
        x_points[i + 1] <- max_idx[1, 2]
        y_points[i + 1] <- max_idx[1, 1]
      }
```

Volvemos a calcular las intesidades correspondientes al nuevo punto de observación.

```
# Calcular nueva intensidad
      dx <- x_points[i + 1] - true_x
      dy <- y_points[i + 1] - true_y
      dist_i <- sqrt(dx^2 + dy^2)
      mu_true <- A * exp(-dist_i / lambda)
      intensities[i + 1] <- rnorm(1, mean = mu_true, sd = sigma)
}
```

Finalmente si el modelo se acerca en un rango de tres casillas tanto para x como para y, el bucle se terminaría ya que daríamos por sentado que el modelo ha encontrado el objeto escondido en nuestro caso el barco.

```
if (abs(max_idx[1, 2] - true_x) <= 3 && abs(max_idx[1, 1] - true_y) <= 3) {
      cat("El modelo ha localizado el objeto en la iteración", i, "\n")
      break
}
```

Con todas estas explicaciones vamos empezar con las simulaciones. El objetivo es ver como cambiando los parámetros encontramos resultados diferentes.

6. Simulaciones

La función que vamos a utlizar para nuestras simulaciones va ser la siguiente:

```
library(cmdstanr)
```

```
## This is cmdstanr version 0.9.0
```

```
## - CmdStanR documentation and vignettes: mc-stan.org/cmdstanr
```

```
## - CmdStan path: C:/Users/madbo/.cmdstan/cmdstan-2.36.0

## - CmdStan version: 2.36.0
```

```
library(ggplot2)

localiza_objeto <- function(true_x = 30, true_y = 70,
                            A = 1, lambda = 20, sigma = 0.5,
                            n_obs = 25, mostrar_plot = TRUE,
                            guardar_datos = TRUE) {

  x_points <- numeric(n_obs)
  y_points <- numeric(n_obs)
  intensities <- numeric(n_obs)
  posterior_matrix <- matrix(0, 100, 100)
  resultados <- list()

  # Observaciones iniciales dirigidas
  obs_iniciales <- data.frame(
    x = c(25, 75, 25, 75, 50),
    y = c(25, 25, 75, 75, 50)
  )
  n_init <- nrow(obs_iniciales)

  x_points[1:n_init] <- obs_iniciales$x
  y_points[1:n_init] <- obs_iniciales$y

  for (j in 1:n_init) {
    dx <- x_points[j] - true_x
    dy <- y_points[j] - true_y
    dist_j <- sqrt(dx^2 + dy^2)
    mu_j <- A * exp(-dist_j / lambda)
    intensities[j] <- rnorm(1, mean = mu_j, sd = sigma)
  }

  for (i in n_init:n_obs) {

    if (i == n_init) {
      x_min <- 0; x_max <- 100
      y_min <- 0; y_max <- 100
    } else {
      x_comb <- c(posterior_samples$x_obj, x_points[1:i])
      y_comb <- c(posterior_samples$y_obj, y_points[1:i])
      x_min <- max(floor(min(x_comb)), 0)
      x_max <- min(ceiling(max(x_comb)), 100)
      y_min <- max(floor(min(y_comb)), 0)
      y_max <- min(ceiling(max(y_comb)), 100)

      if (x_min == x_max) { x_min <- max(x_min - 1, 0); x_max <- min(x_max + 1, 100) }
      if (y_min == y_max) { y_min <- max(y_min - 1, 0); y_max <- min(y_max + 1, 100) }
    }

    stan_code <- sprintf("
      data {
```

```
    int<lower=1> n;
    vector[n] x_obs;
    vector[n] y_obs;
    real<lower=0> A;
    vector[n] intensity;
    real<lower=0> sigma;
    real<lower=0> lambda;
  }
  parameters {
    real<lower=%f, upper=%f> x_obj;
    real<lower=%f, upper=%f> y_obj;
  }
  model {
    for (i in 1:n) {
      real d = sqrt((x_obs[i] - x_obj)^2 + (y_obs[i] - y_obj)^2);
      real mu = A * exp(-d / lambda);
      intensity[i] ~ normal(mu, sigma);
    }
  }
", x_min, x_max, y_min, y_max)

archivo_stan <- tempfile(fileext = ".stan")
writeLines(stan_code, archivo_stan)
modelo_dinamico <- cmdstan_model(archivo_stan)

data_i <- list(
  n = i,
  A = A,
  x_obs = as.numeric(x_points[1:i]),
  y_obs = as.numeric(y_points[1:i]),
  intensity = as.numeric(intensities[1:i]),
  sigma = sigma,
  lambda = lambda
)

fit <- modelo_dinamico$sample(
  data = data_i,
  seed = 123,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 500,
  iter_sampling = 500,
  refresh = 0
)

posterior_samples <- fit$draws(variables = c("x_obj", "y_obj"), format = "df")
x_post <- pmin(pmax(round(posterior_samples$x_obj), 1), 100)
y_post <- pmin(pmax(round(posterior_samples$y_obj), 1), 100)

for (j in seq_along(x_post)) {
  posterior_matrix[y_post[j], x_post[j]] <- posterior_matrix[y_post[j], x_post[j]] + 1
}
```

```
  posterior_matrix_norm <- posterior_matrix / sum(posterior_matrix)
  max_idx <- which(posterior_matrix_norm == max(posterior_matrix_norm), arr.ind = TRUE)

  image_matrix <- as.data.frame(as.table(posterior_matrix_norm))
  colnames(image_matrix) <- c("y", "x", "prob")
  image_matrix$x <- as.numeric(image_matrix$x)
  image_matrix$y <- as.numeric(image_matrix$y)

  plot_title <- sprintf("Iteración %d: Prior [%d,%d] x [%d,%d]", i, x_min, x_max, y_min, y_max)

  if (mostrar_plot) {
    print(
      ggplot(image_matrix, aes(x = x, y = y)) +
        geom_tile(aes(fill = prob)) +
        geom_point(data = data.frame(x = true_x, y = true_y),
                   mapping = aes(x = x, y = y),
                   color = "black", size = 3, shape = 4) +
        geom_point(data = data.frame(x = x_points[i], y = y_points[i]),
                   mapping = aes(x = x, y = y),
                   color = "blue", size = 3, shape = 4) +
        scale_fill_gradient(low = "white", high = "red") +
        coord_fixed() +
        labs(title = plot_title) +
        theme_minimal()
    )
  }

  cat("\nIteración:", i,
      "\nCoordenadas de observación:", x_points[i], y_points[i],
      "\nIntensidad observada =", round(intensities[i], 4),
      "\nCelda más probable =", max_idx[1, 2], max_idx[1, 1],
      "\nProbabilidad posterior  ", round(max(posterior_matrix_norm), 4),
      "\nPrior actual: x   [", x_min, ",", x_max, "], y   [", y_min, ",", y_max, "]\n")

  if (guardar_datos) {
    resultados[[i]] <- list(
      image_matrix = image_matrix,
      title = plot_title,
      max_idx = max_idx,
      posterior_matrix = posterior_matrix,
      posterior_matrix_norm = posterior_matrix_norm
    )
  }

  if (i < n_obs) {
    celdas_usadas <- paste(x_points[1:i], y_points[1:i], sep = "_")
    orden <- order(posterior_matrix_norm, decreasing = TRUE)
    posiciones <- arrayInd(orden, .dim = dim(posterior_matrix_norm))
    encontrado <- FALSE
    for (k in 1:nrow(posiciones)) {
      propuesta <- posiciones[k, ]
      celda_id <- paste(propuesta[2], propuesta[1], sep = "_")
      if (!(celda_id %in% celdas_usadas)) {
```

```
          x_points[i + 1] <- propuesta[2]
          y_points[i + 1] <- propuesta[1]
          encontrado <- TRUE
          break
        }
      }
      # Si no se encuentra celda nueva, se repite la más probable
      if (!encontrado) {
        x_points[i + 1] <- max_idx[1, 2]
        y_points[i + 1] <- max_idx[1, 1]
      }

      # Calcular nueva intensidad
      dx <- x_points[i + 1] - true_x
      dy <- y_points[i + 1] - true_y
      dist_i <- sqrt(dx^2 + dy^2)
      mu_true <- A * exp(-dist_i / lambda)
      intensities[i + 1] <- rnorm(1, mean = mu_true, sd = sigma)
    }

    if (abs(max_idx[1, 2] - true_x) <= 3 && abs(max_idx[1, 1] - true_y) <= 3) {
      cat("El modelo ha localizado el objeto en la iteración", i, "\n")
      break
    }
  }

  return(invisible(resultados))
}
```

Para todas las simulaciones la X azul es donde se ha realizado la última observación y la X negra es donde está el objeto.

Las simulaciones no son siempre iguales así que pueden suceder resultados en ocasiones un tanto diferentes. Supongo que esto se debe a parte del ruido que se parametriza en el modelo, al ruido que yo mismo género con los redondeos de ciertas probabilidades continuas. Por eso algunas conclusiones, al hacerse estas antes del renderizado, donde se vuelven ha ejecutar todas las simulaciones pueden llegar a ser algo diferentes.

1. Grupo de simulaciones

En este primer grupo de simulaciones vamos a varíar σ para ver como cambían los resultados.

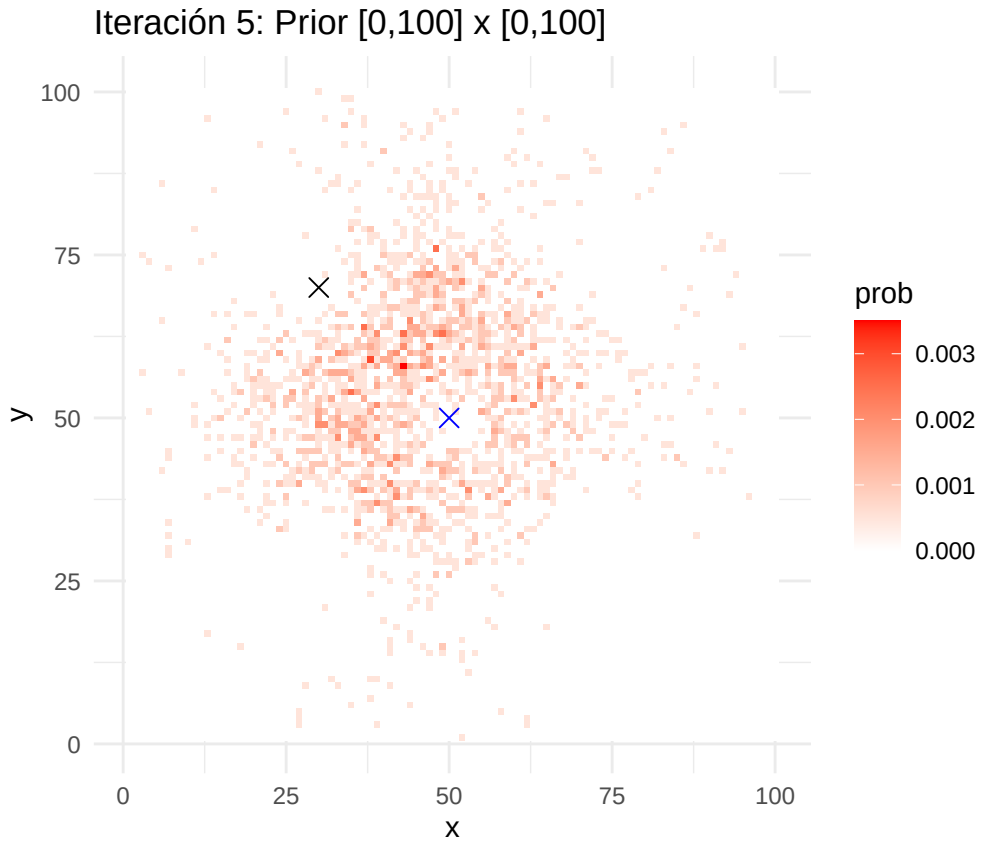
1.1

Para la primera simulación vamos a poner un valor de 0,2.

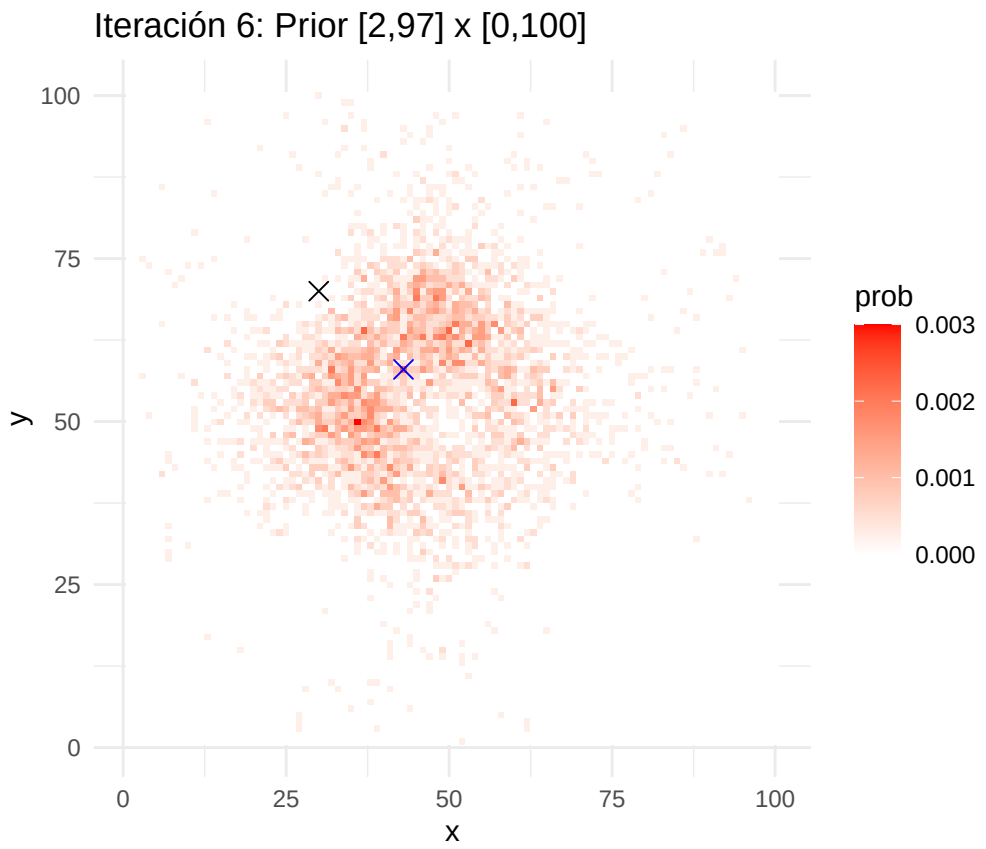
```
sim1 <- localiza_objeto(true_x = 30, true_y = 70,
                        A = 1, lambda = 20, sigma = 0.2,
                        n_obs = 25, mostrar_plot = TRUE,
                        guardar_datos = TRUE)
```

```
## Running MCMC with 4 parallel chains...
##
```

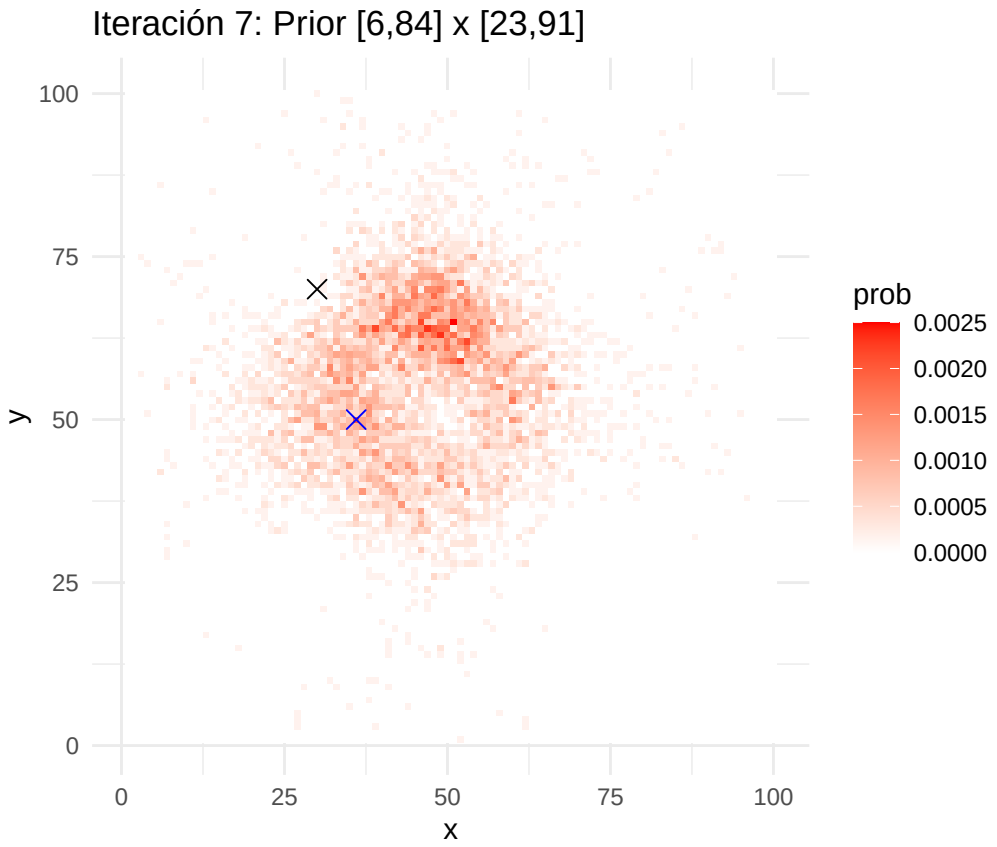
```
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.4 seconds.
```



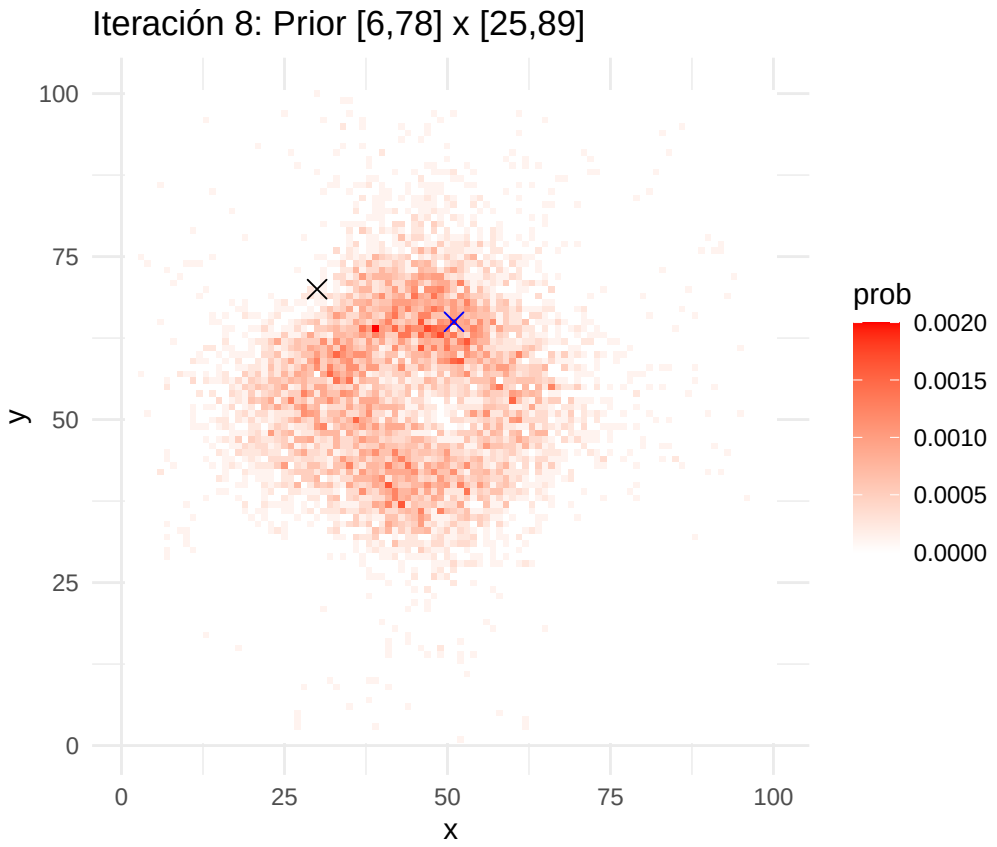

```
##
## Iteración: 5
## Coordenadas de observación: 50 50
## Intensidad observada = 0.4991
## Celda más probable = 43 58
## Probabilidad posterior  0.0035
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



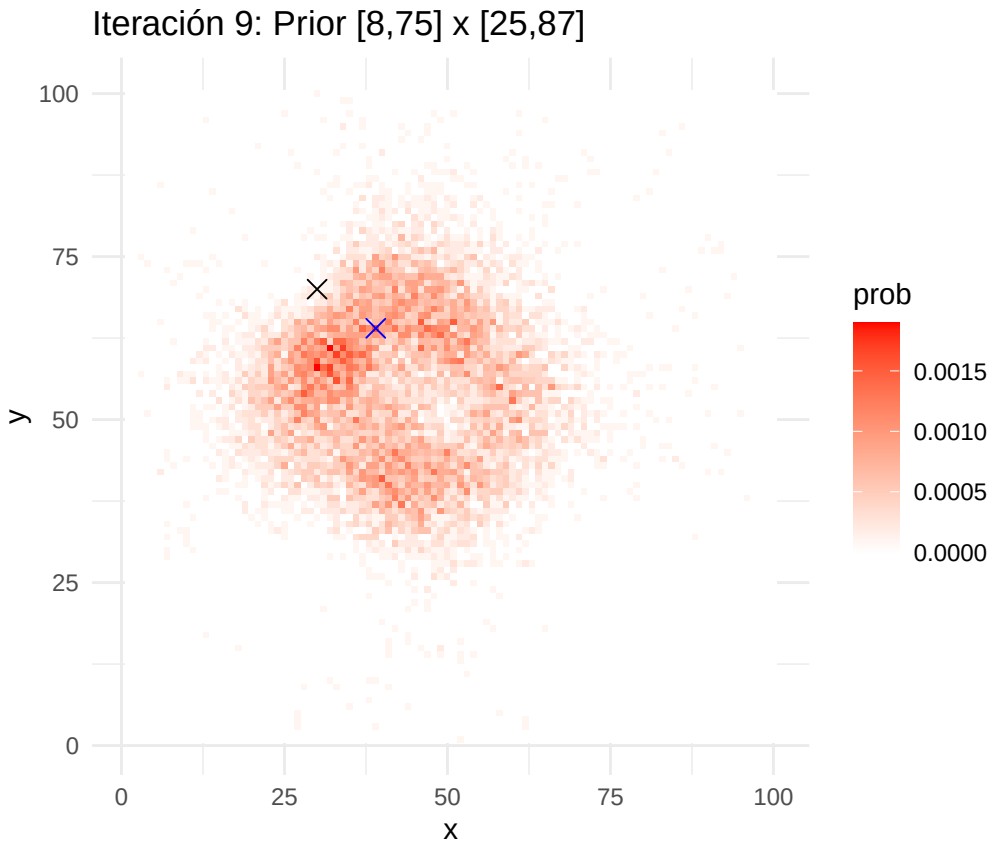

```
##
## Iteración: 6
## Coordenadas de observación: 43 58
## Intensidad observada = 0.5962
## Celda más probable = 36 50
## Probabilidad posterior  0.003
## Prior actual: x  [ 2 , 97 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



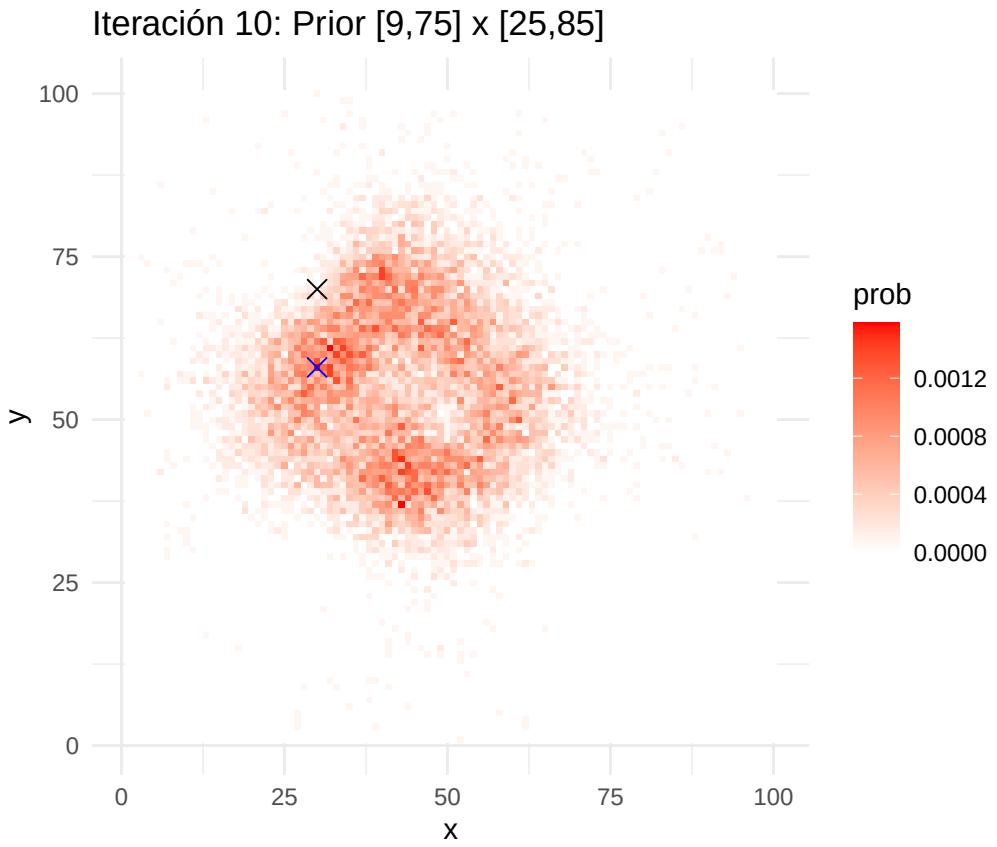

```
##
## Iteración: 7
## Coordenadas de observación: 36 50
## Intensidad observada = 0.3909
## Celda más probable = 51 65
## Probabilidad posterior  0.0025
## Prior actual: x  [ 6 , 84 ], y  [ 23 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.2 seconds.
```



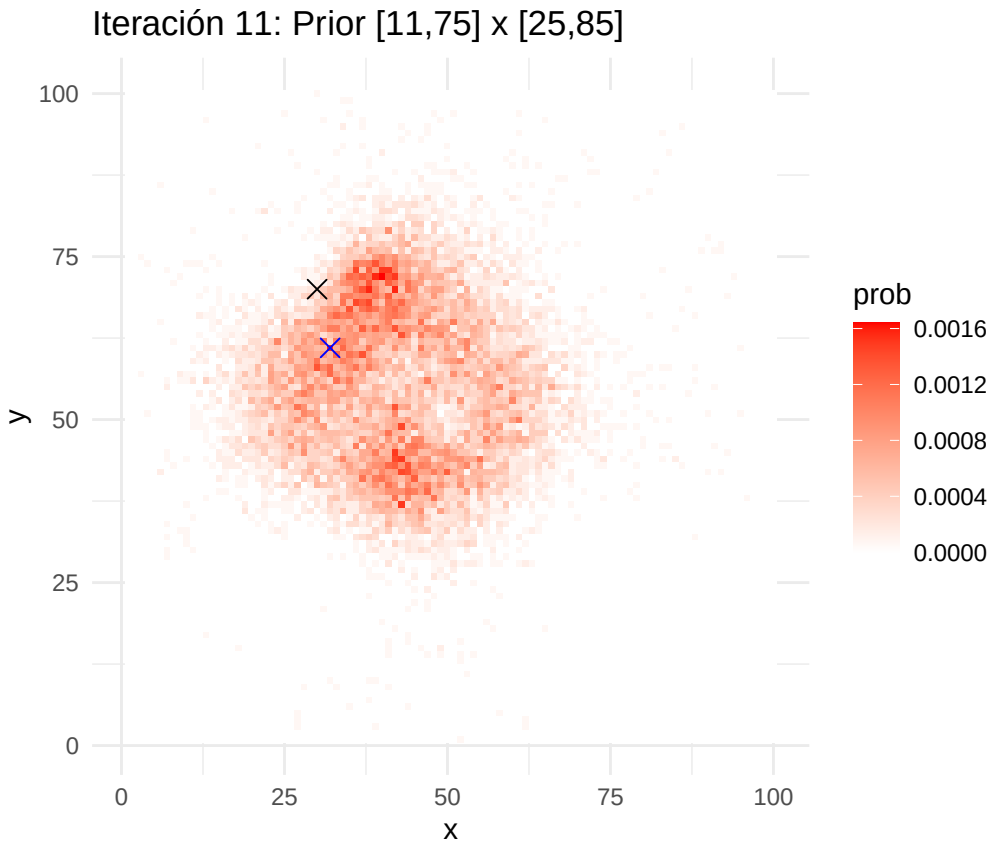

```
##
## Iteración: 8
## Coordenadas de observación: 51 65
## Intensidad observada = 0.1737
## Celda más probable = 39 64
## Probabilidad posterior  0.002
## Prior actual: x  [ 6 , 78 ], y  [ 25 , 89 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



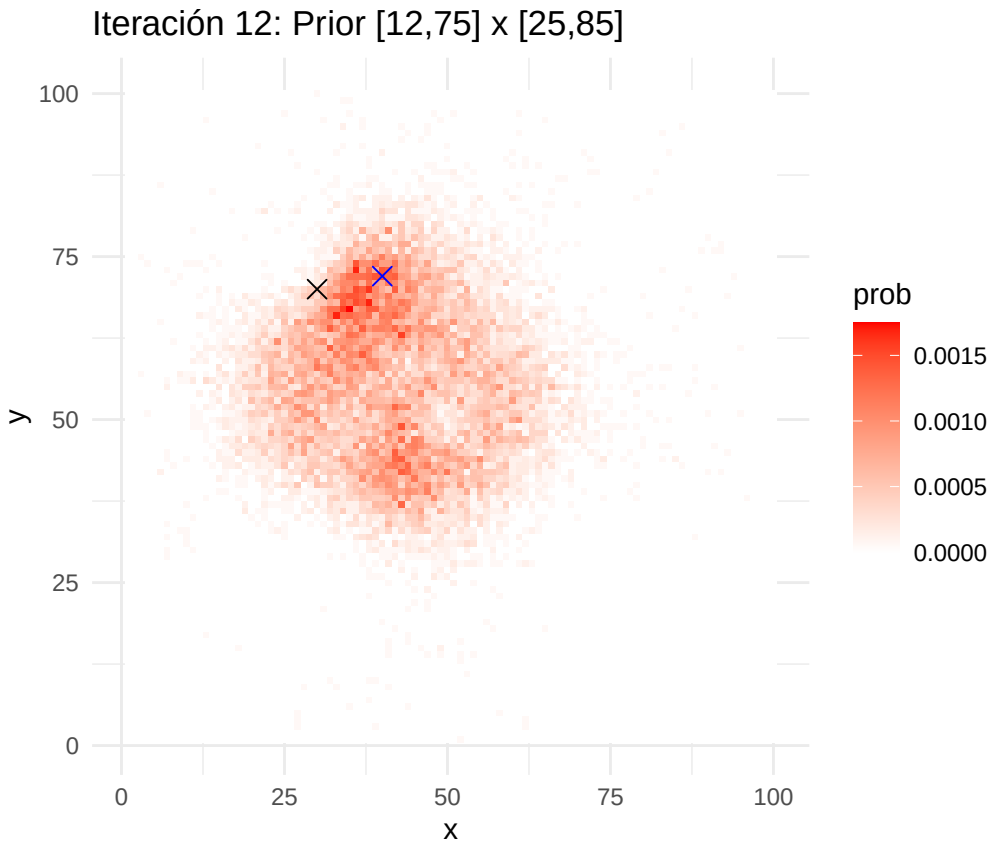

```
##
## Iteración: 9
## Coordenadas de observación: 39 64
## Intensidad observada = 0.6576
## Celda más probable = 30 58
## Probabilidad posterior  0.0019
## Prior actual: x  [ 8 , 75 ], y  [ 25 , 87 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



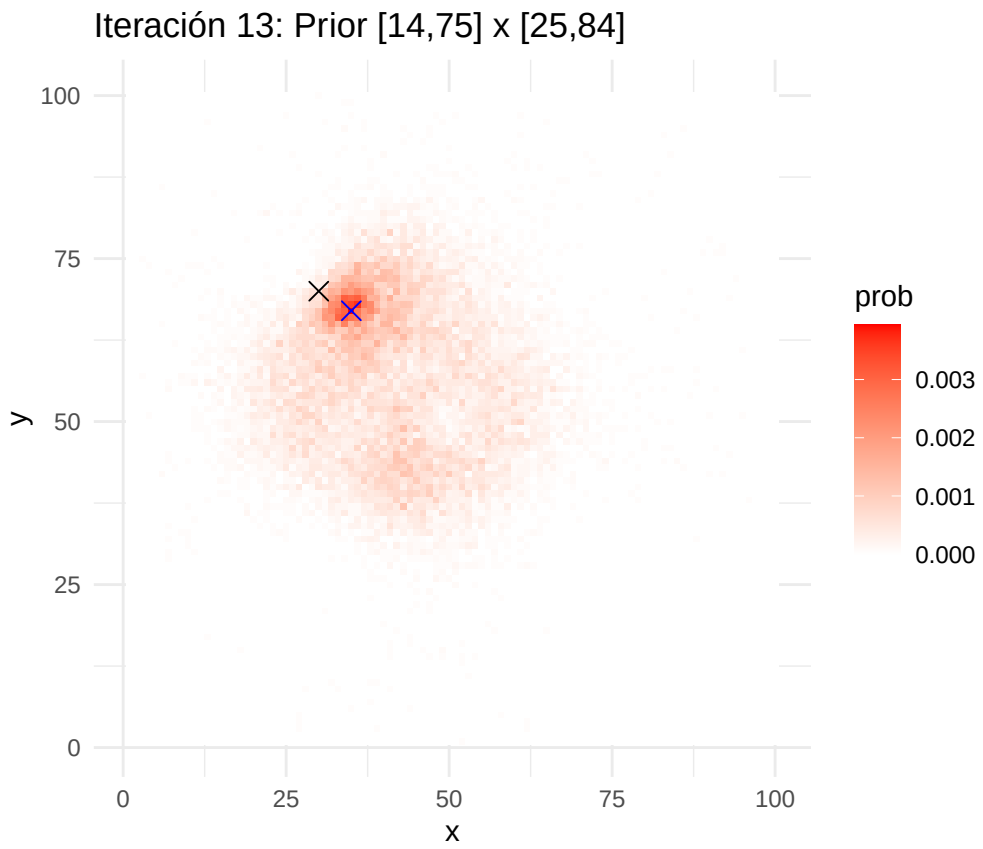

```
##
## Iteración: 10
## Coordenadas de observación: 30 58
## Intensidad observada = 0.2231
## Celda más probable = 30 58
## Probabilidad posterior  0.0016
## Prior actual: x  [ 9 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



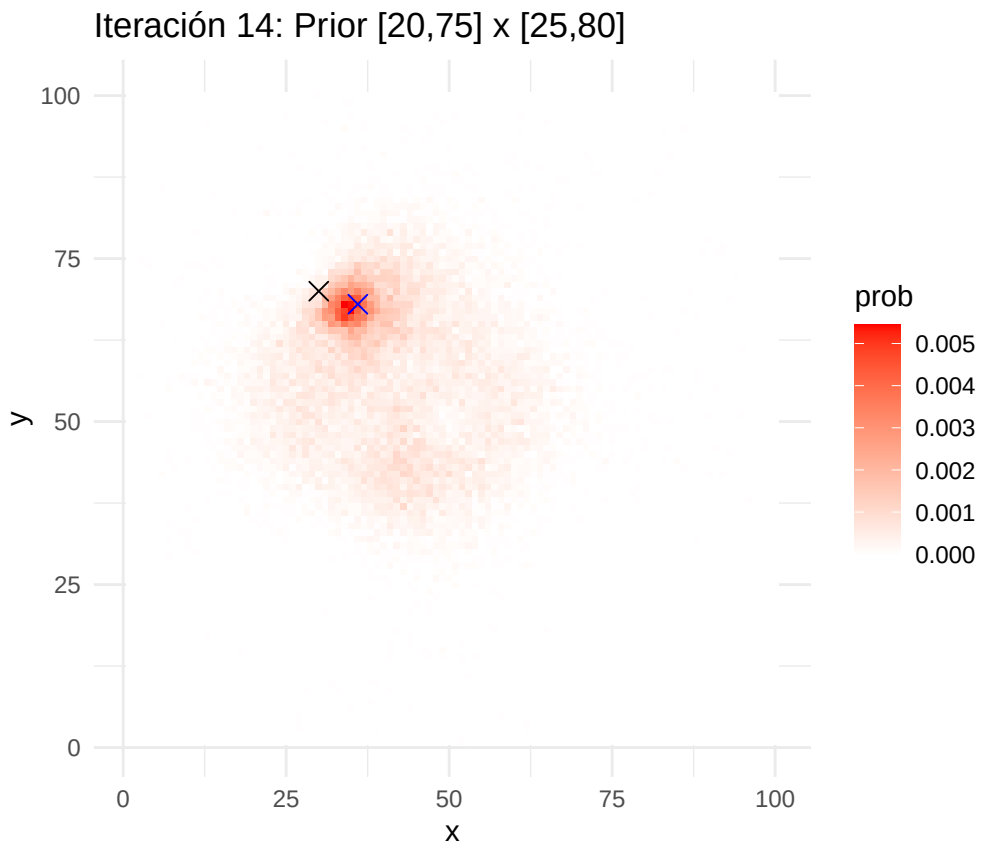

```
##
## Iteración: 11
## Coordenadas de observación: 32 61
## Intensidad observada = 0.7401
## Celda más probable = 40 72
## Probabilidad posterior  0.0016
## Prior actual: x  [ 11 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



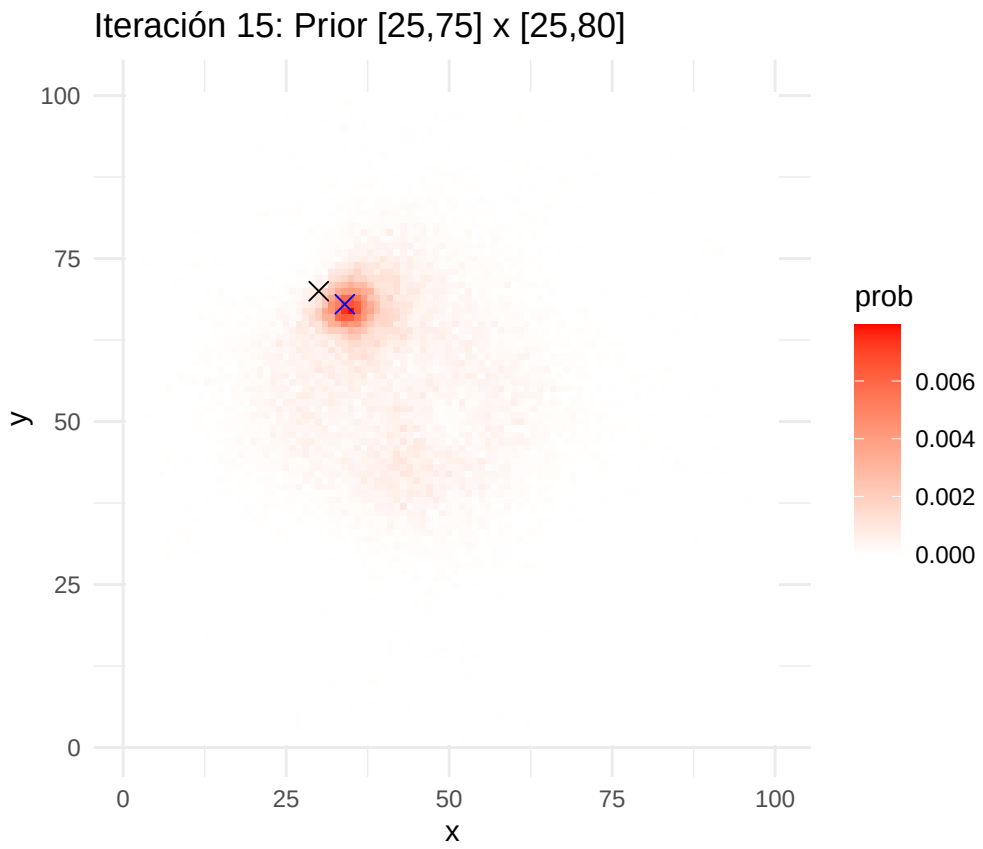

```
##
## Iteración: 12
## Coordenadas de observación: 40 72
## Intensidad observada = 0.5336
## Celda más probable = 35 67
## Probabilidad posterior  0.0018
## Prior actual: x  [ 12 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



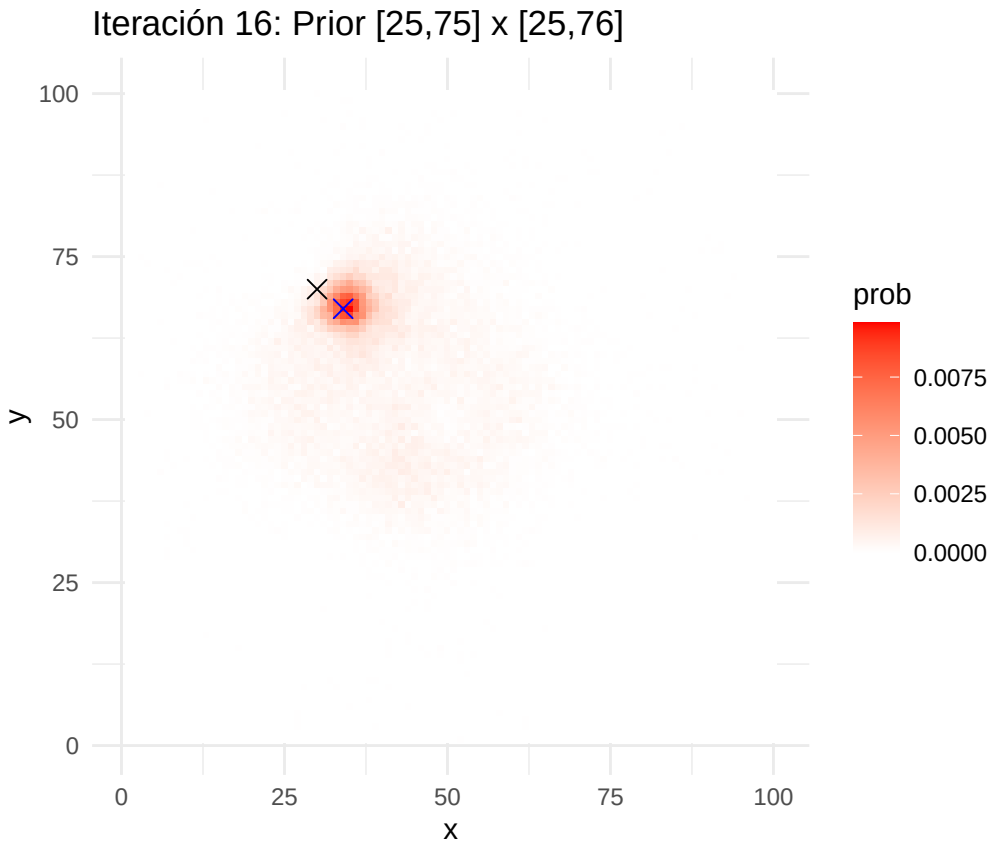

```
##
## Iteración: 13
## Coordenadas de observación: 35 67
## Intensidad observada = 1.0799
## Celda más probable = 35 67
## Probabilidad posterior  0.0039
## Prior actual: x  [ 14 , 75 ], y  [ 25 , 84 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



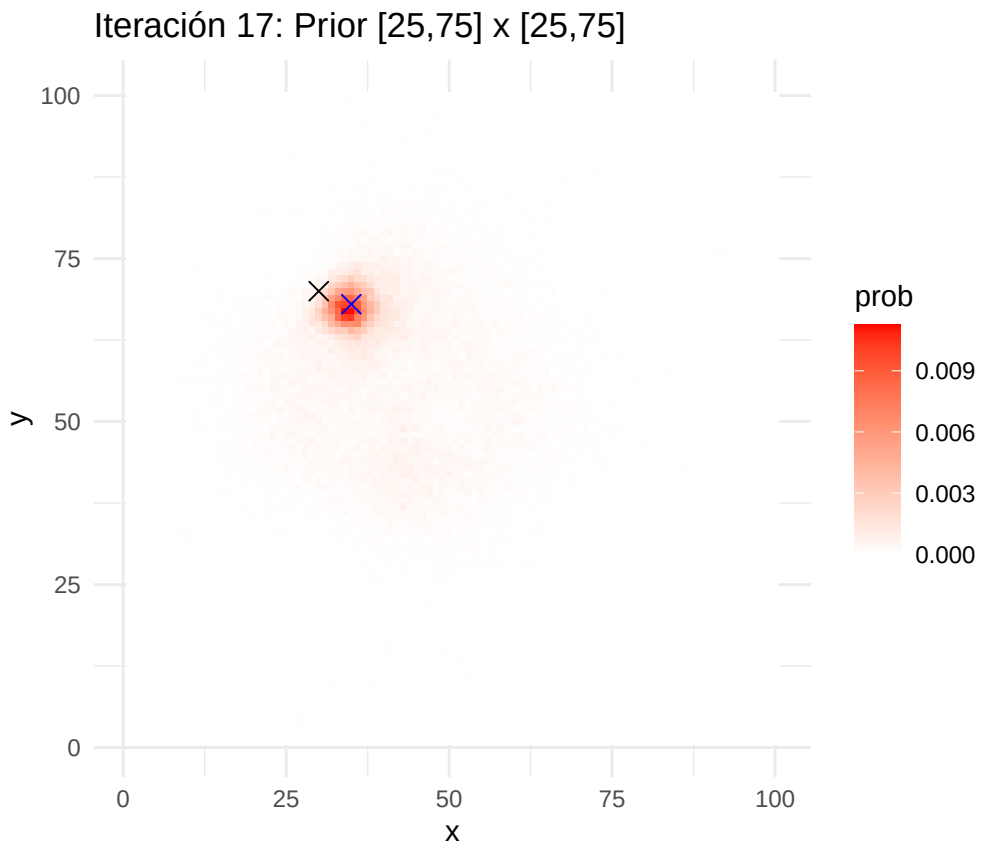

```
##
## Iteración: 14
## Coordenadas de observación: 36 68
## Intensidad observada = 0.8679
## Celda más probable = 34 68
## Probabilidad posterior  0.0054
## Prior actual: x  [ 20 , 75 ], y  [ 25 , 80 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



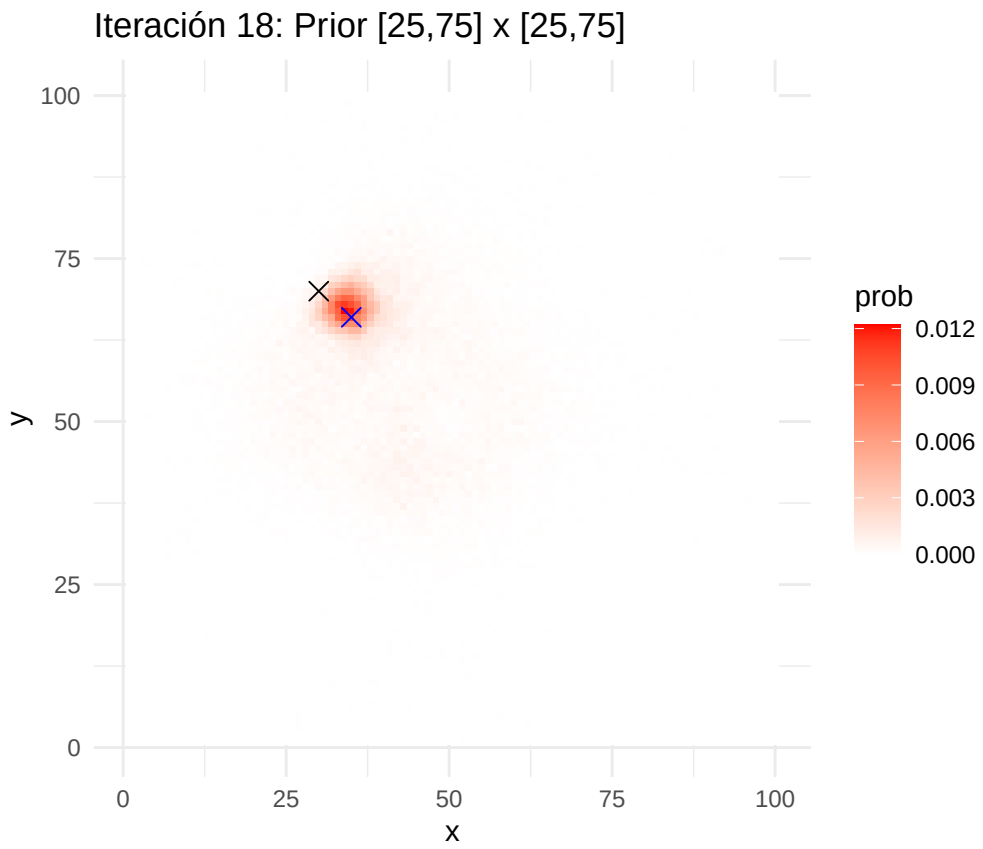

```
##
## Iteración: 15
## Coordenadas de observación: 34 68
## Intensidad observada = 0.9526
## Celda más probable = 35 67
## Probabilidad posterior  0.008
## Prior actual: x  [ 25 , 75 ], y  [ 25 , 80 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



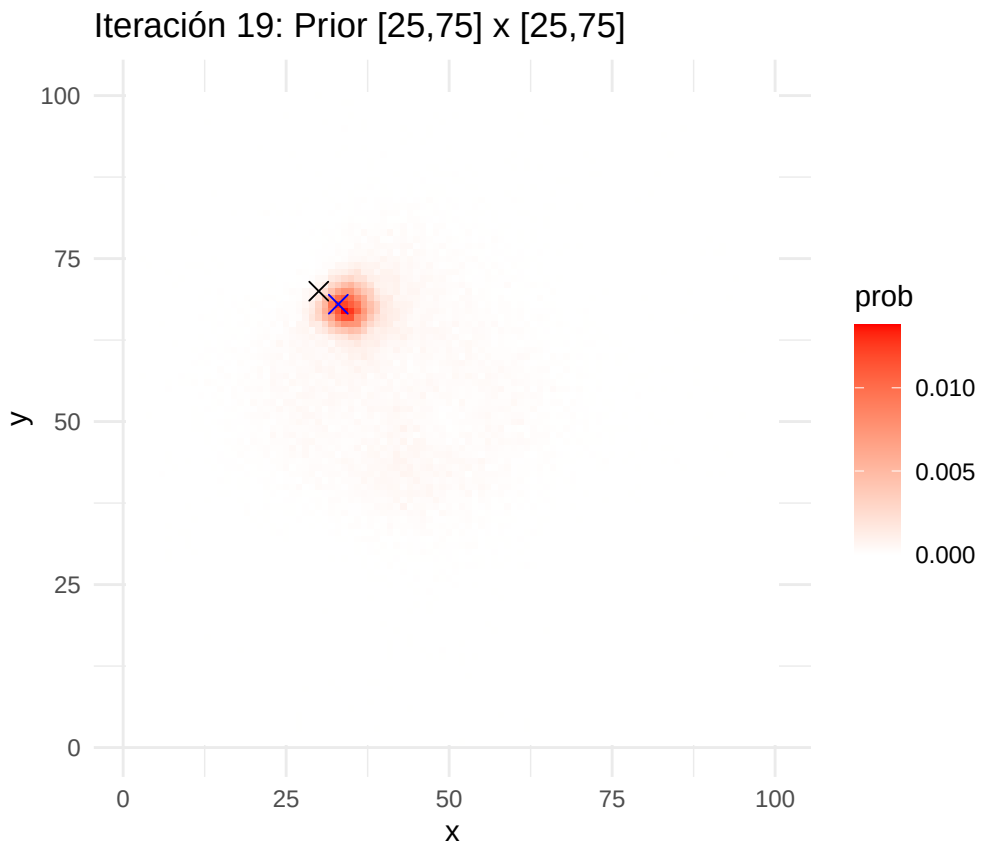

```
##
## Iteración: 16
## Coordenadas de observación: 34 67
## Intensidad observada = 1.0212
## Celda más probable = 35 67
## Probabilidad posterior  0.0098
## Prior actual: x   [ 25 , 75 ], y   [ 25 , 76 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



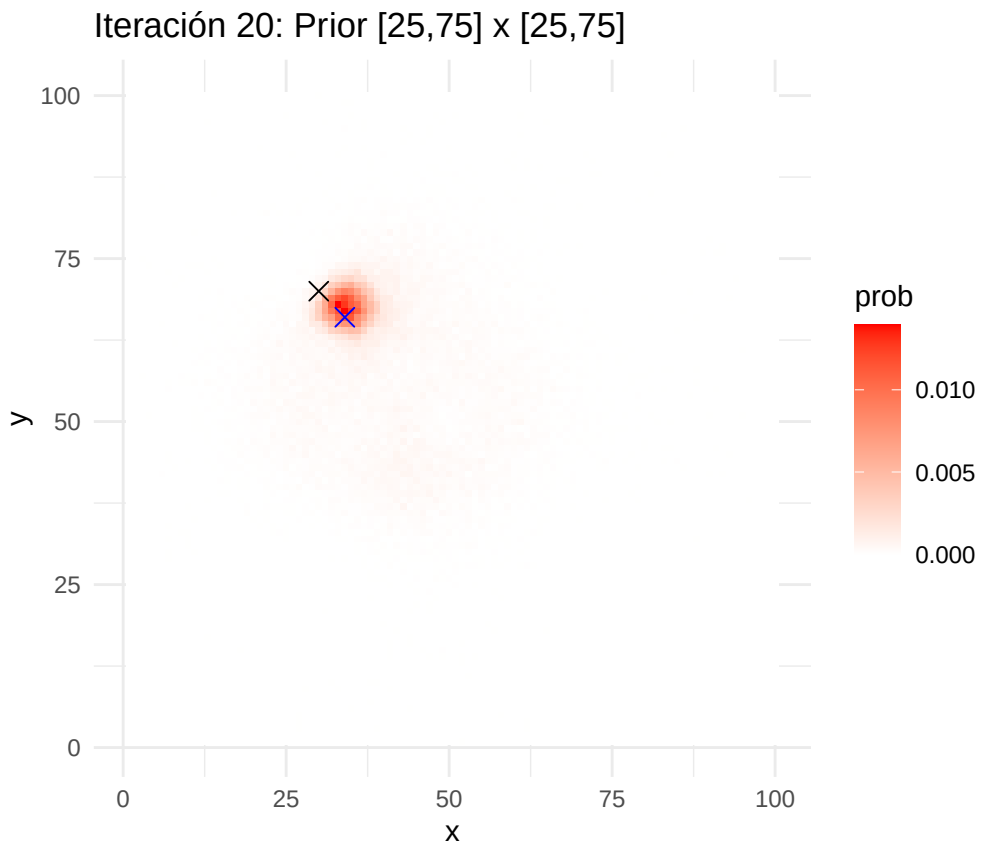

```
##
## Iteración: 17
## Coordenadas de observación: 35 68
## Intensidad observada = 0.732
## Celda más probable = 34 67
## Probabilidad posterior  0.0113
## Prior actual: x  [ 25 , 75 ], y  [ 25 , 75 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 18
## Coordenadas de observación: 35 66
## Intensidad observada = 0.72
## Celda más probable = 34 67
## Probabilidad posterior  0.0122
## Prior actual: x  [ 25 , 75 ], y  [ 25 , 75 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 19
## Coordenadas de observación: 33 68
## Intensidad observada = 1.0495
## Celda más probable = 34 67
## Probabilidad posterior  0.0138
## Prior actual: x  [ 25 , 75 ], y  [ 25 , 75 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 20
## Coordenadas de observación: 34 66
## Intensidad observada = 0.589
## Celda más probable = 33 68
## Probabilidad posterior  0.014
## Prior actual: x   [ 25 , 75 ], y   [ 25 , 75 ]
## El modelo ha localizado el objeto en la iteración 20
```

En está simulación el modelo suele encontrar el objeto o al menos aproximarse rápidamente a su ubicación. El sigma en este nivel es bastante bajo para que no moleste demasiado.

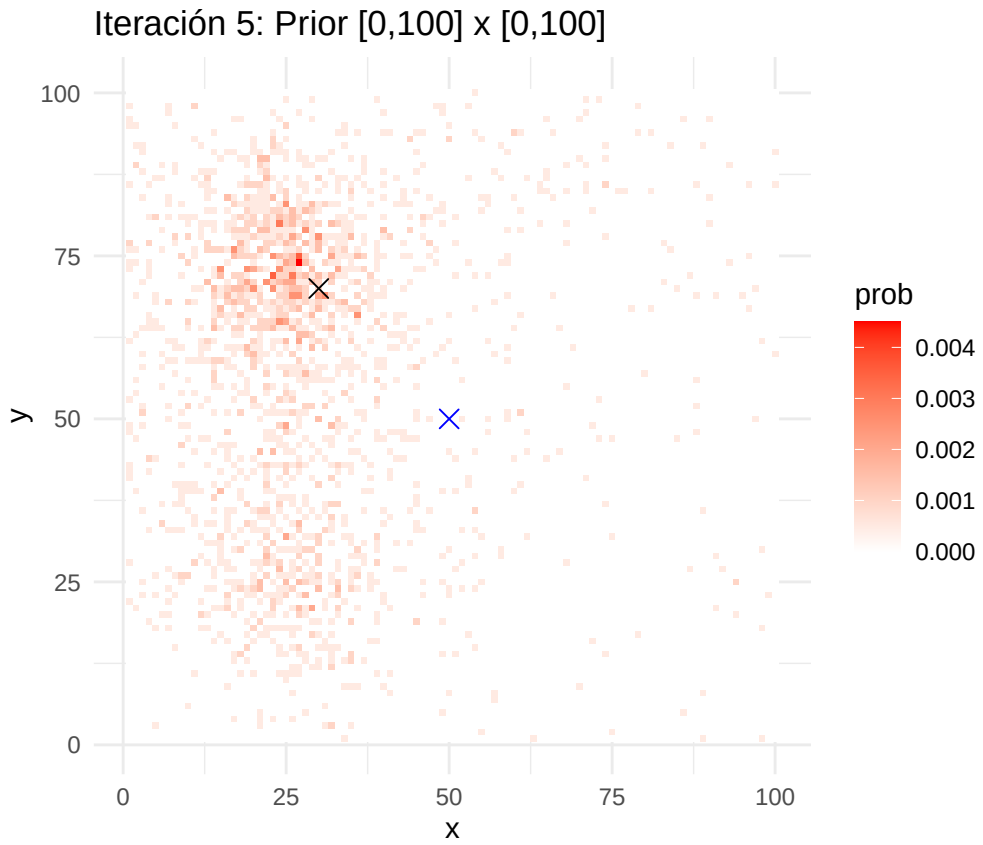
1.2

Para esta simulación el valor de sigma es 0,5

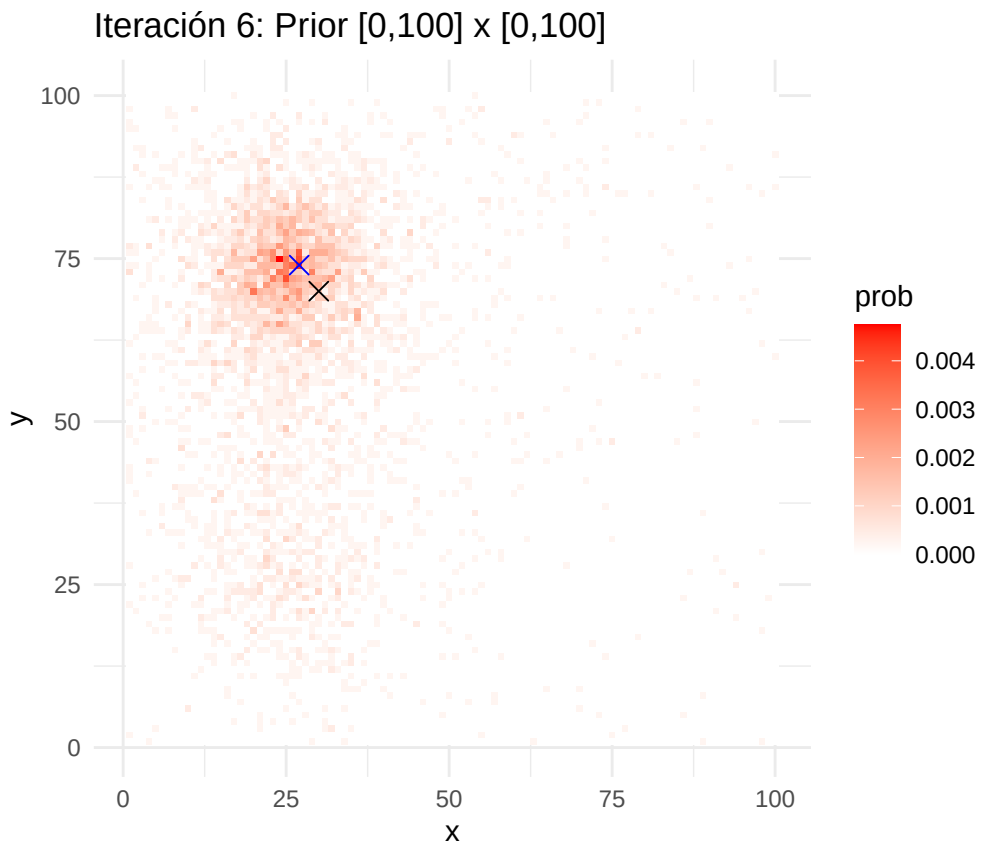
```
sim2 <- localiza_objeto(true_x = 30, true_y = 70,
                            A = 1, lambda = 20, sigma = 0.5,
                            n_obs = 25, mostrar_plot = TRUE,
                            guardar_datos = TRUE)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
```

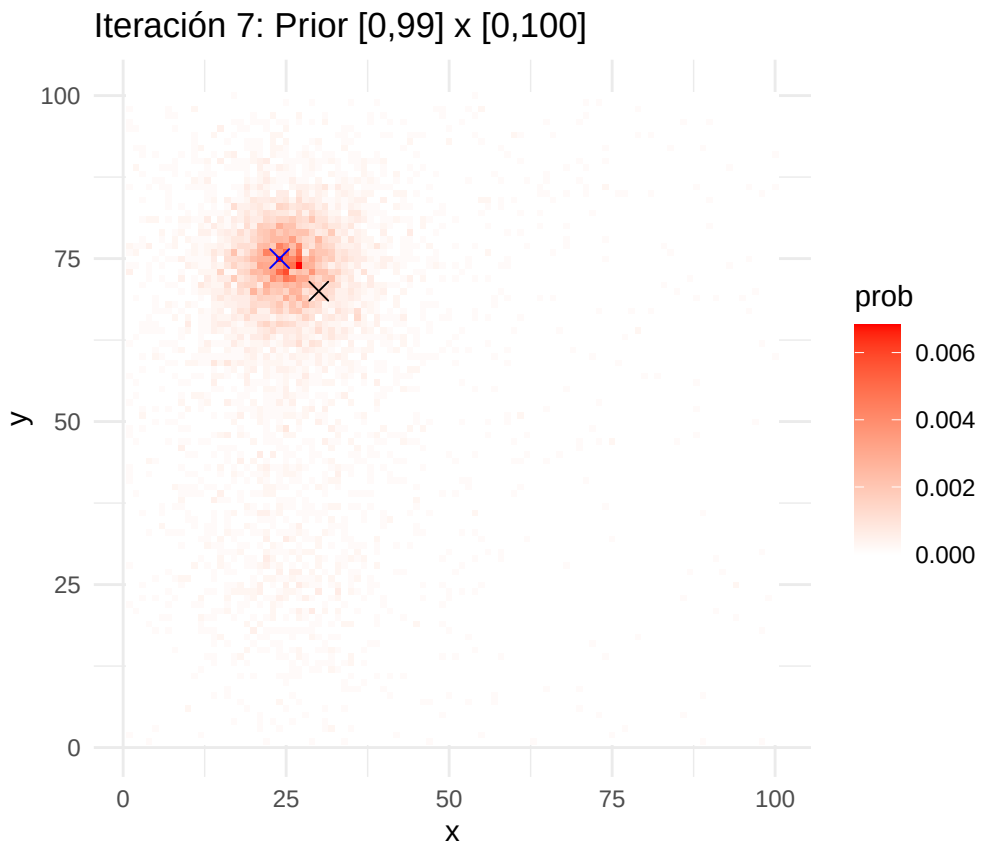
```
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



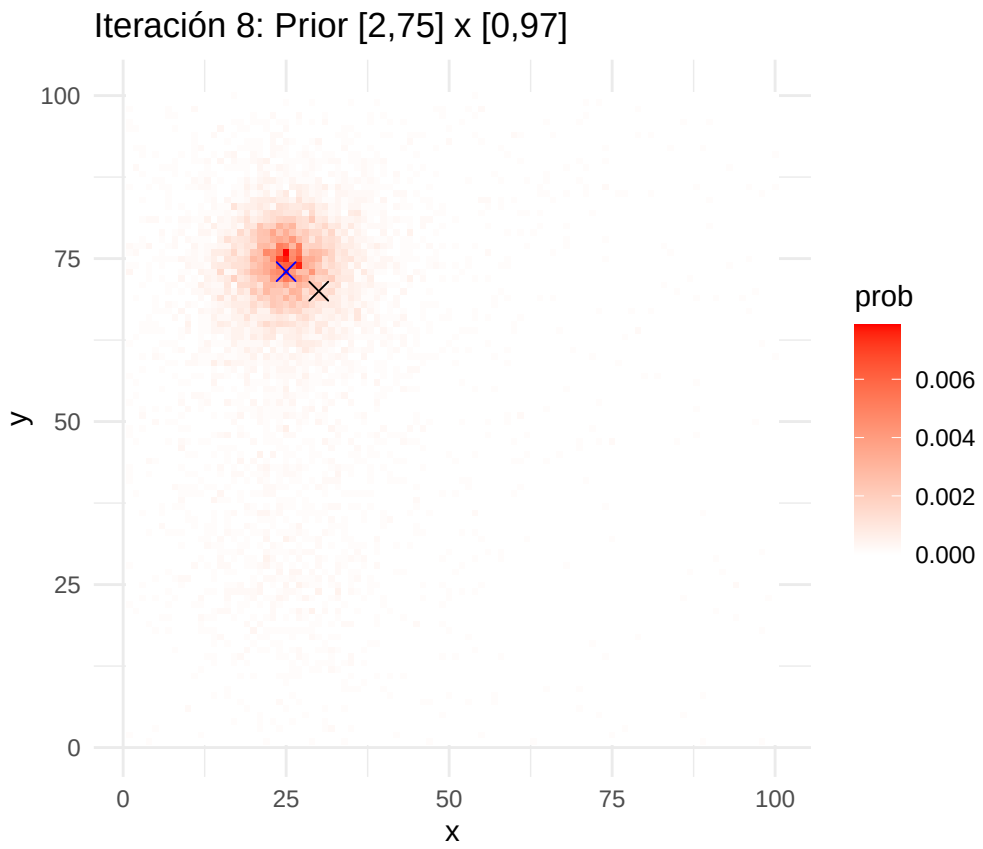

```
##
## Iteración: 5
## Coordenadas de observación: 50 50
## Intensidad observada = 0.2574
## Celda más probable = 27 74
## Probabilidad posterior  0.0045
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



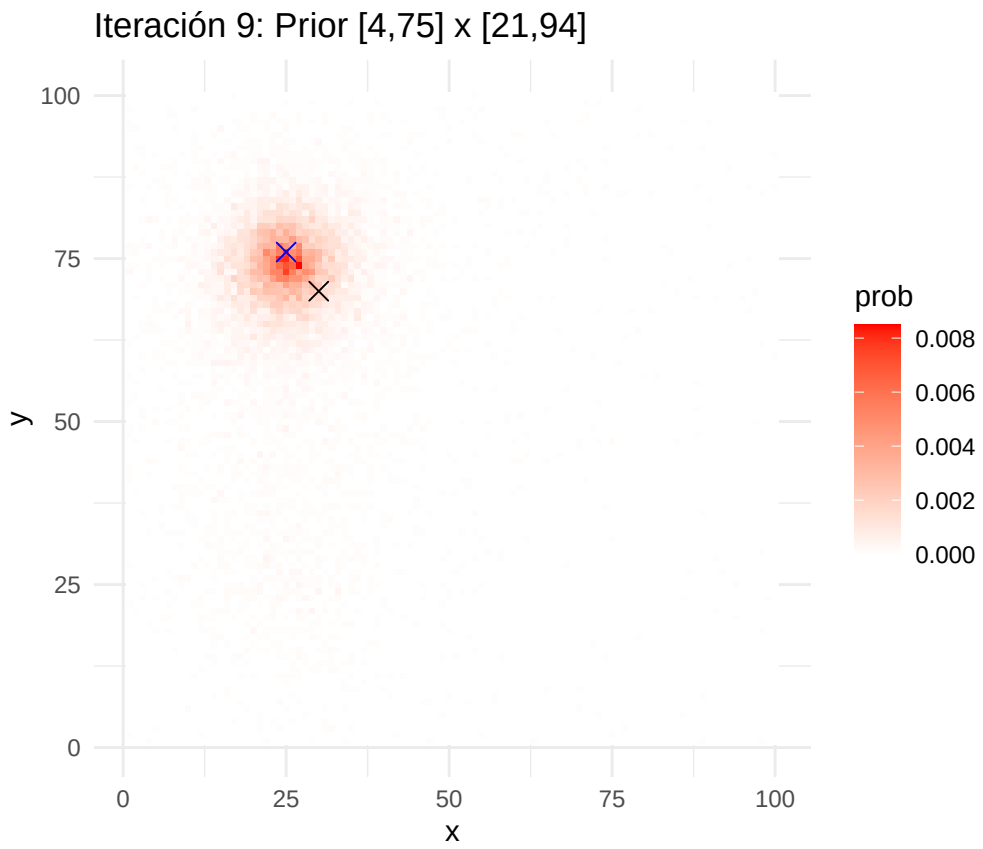

```
##
## Iteración: 6
## Coordenadas de observación: 27 74
## Intensidad observada = 1.1042
## Celda más probable = 24 75
## Probabilidad posterior  0.0048
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



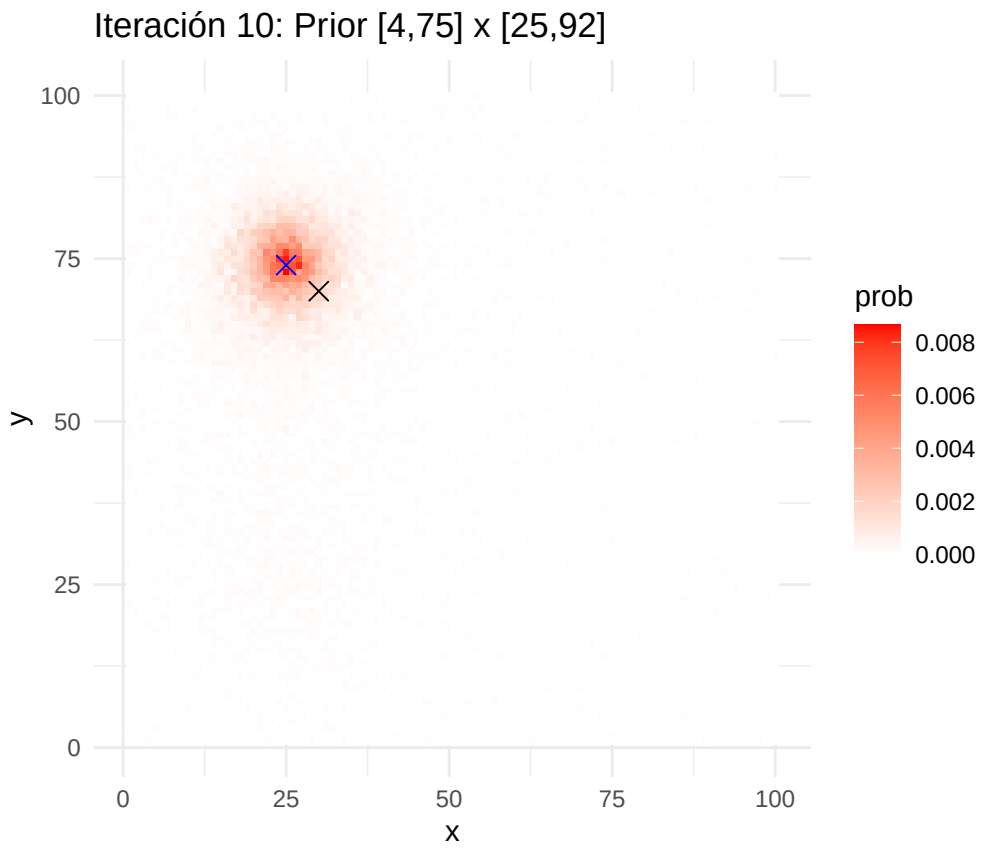

```
##
## Iteración: 7
## Coordenadas de observación: 24 75
## Intensidad observada = 1.501
## Celda más probable = 24 75
## Probabilidad posterior  0.0068
## Prior actual: x  [ 0 , 99 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



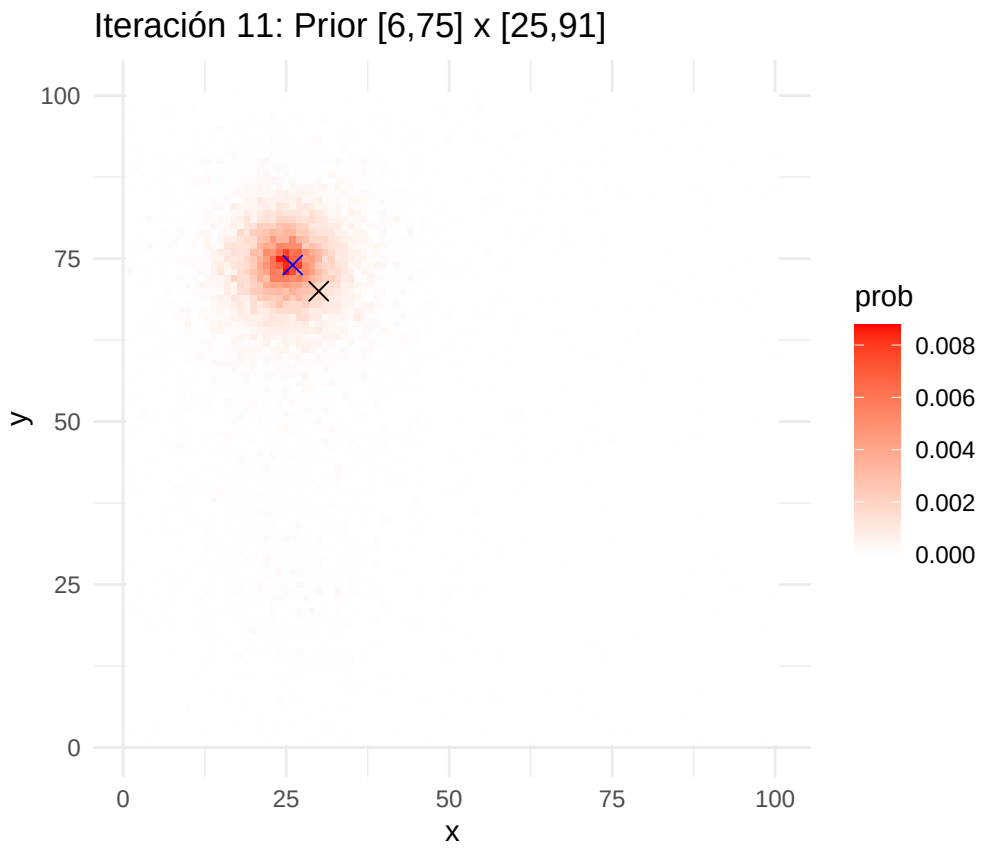

```
##
## Iteración: 8
## Coordenadas de observación: 25 73
## Intensidad observada = 0.8942
## Celda más probable = 25 76
## Probabilidad posterior  0.0079
## Prior actual: x  [ 2 , 75 ], y  [ 0 , 97 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



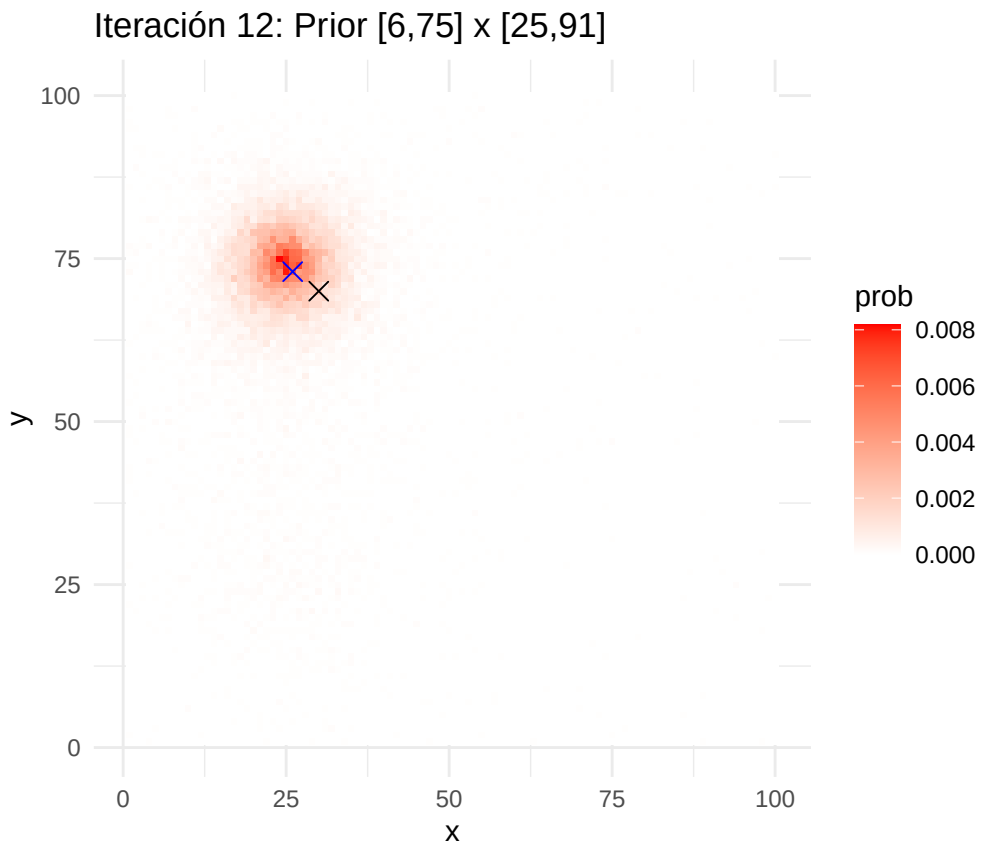

```
##
## Iteración: 9
## Coordenadas de observación: 25 76
## Intensidad observada = 0.4471
## Celda más probable = 27 74
## Probabilidad posterior  0.0085
## Prior actual: x  [ 4 , 75 ], y  [ 21 , 94 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



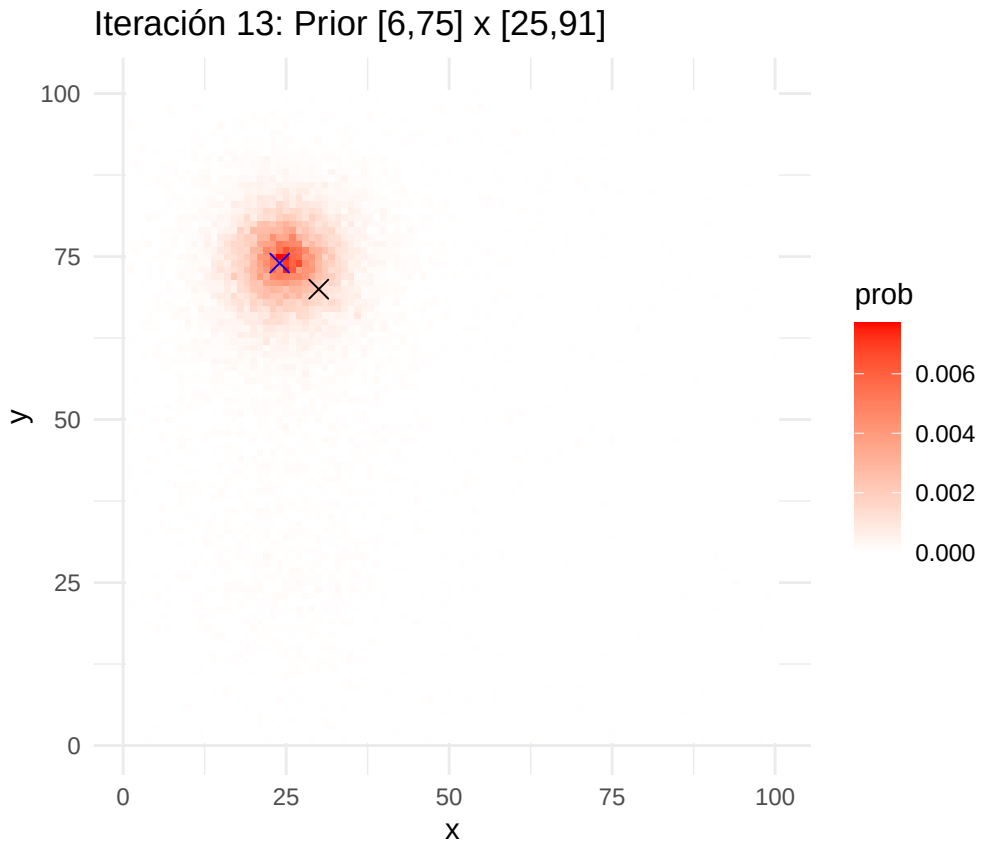

```
##
## Iteración: 10
## Coordenadas de observación: 25 74
## Intensidad observada = 1.0826
## Celda más probable = 25 73
## Probabilidad posterior  0.0087
## Prior actual: x  [ 4 , 75 ], y  [ 25 , 92 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



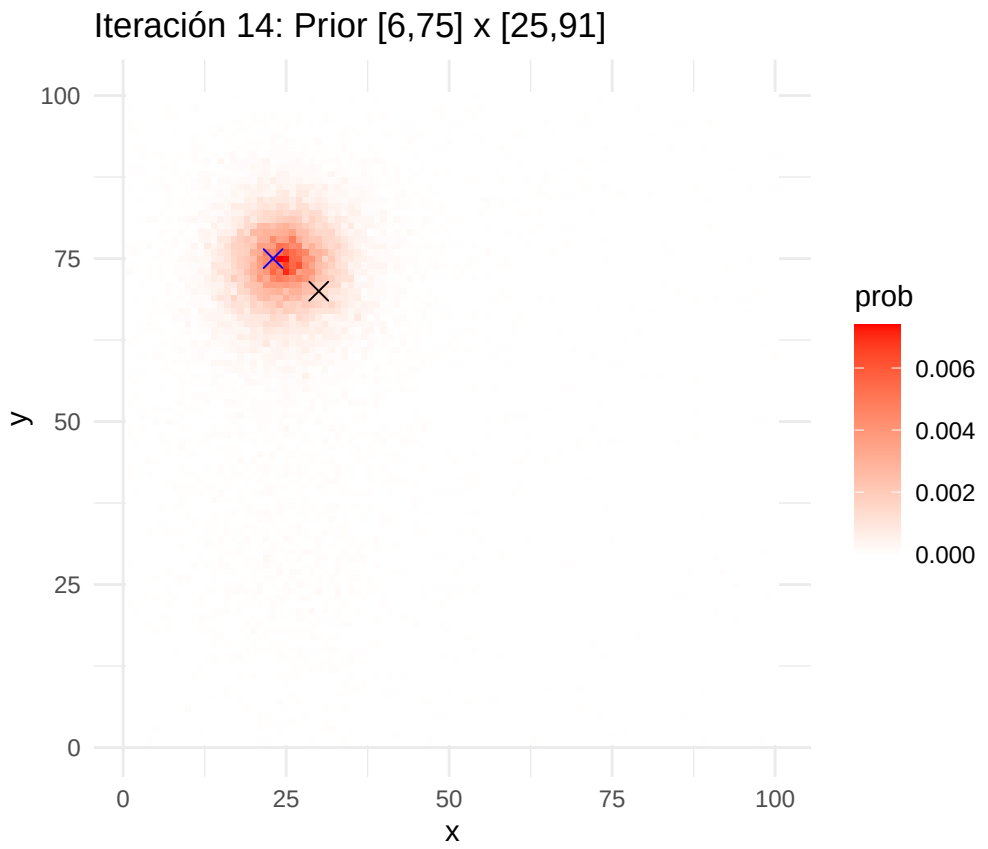

```
##
## Iteración: 11
## Coordenadas de observación: 26 74
## Intensidad observada = 0.458
## Celda más probable = 25 73
## Probabilidad posterior  0.0088
## Prior actual: x  [ 6 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



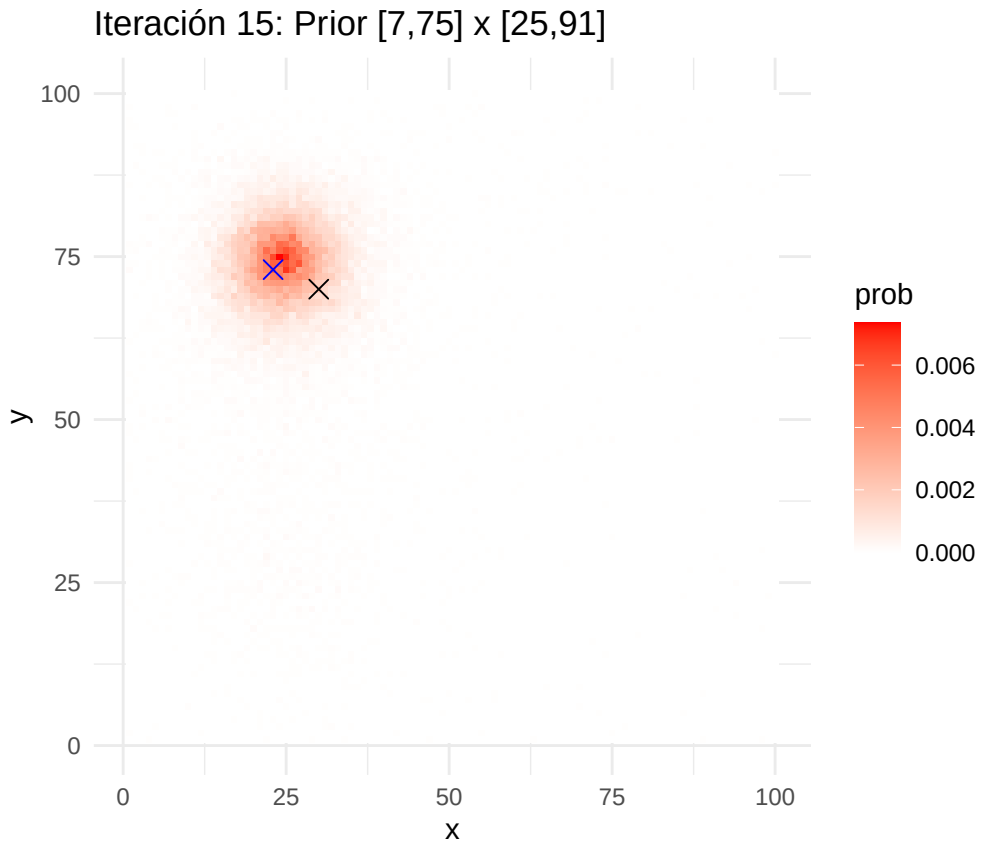

```
##
## Iteración: 12
## Coordenadas de observación: 26 73
## Intensidad observada = -0.5697
## Celda más probable = 24 75
## Probabilidad posterior  0.0082
## Prior actual: x  [ 6 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



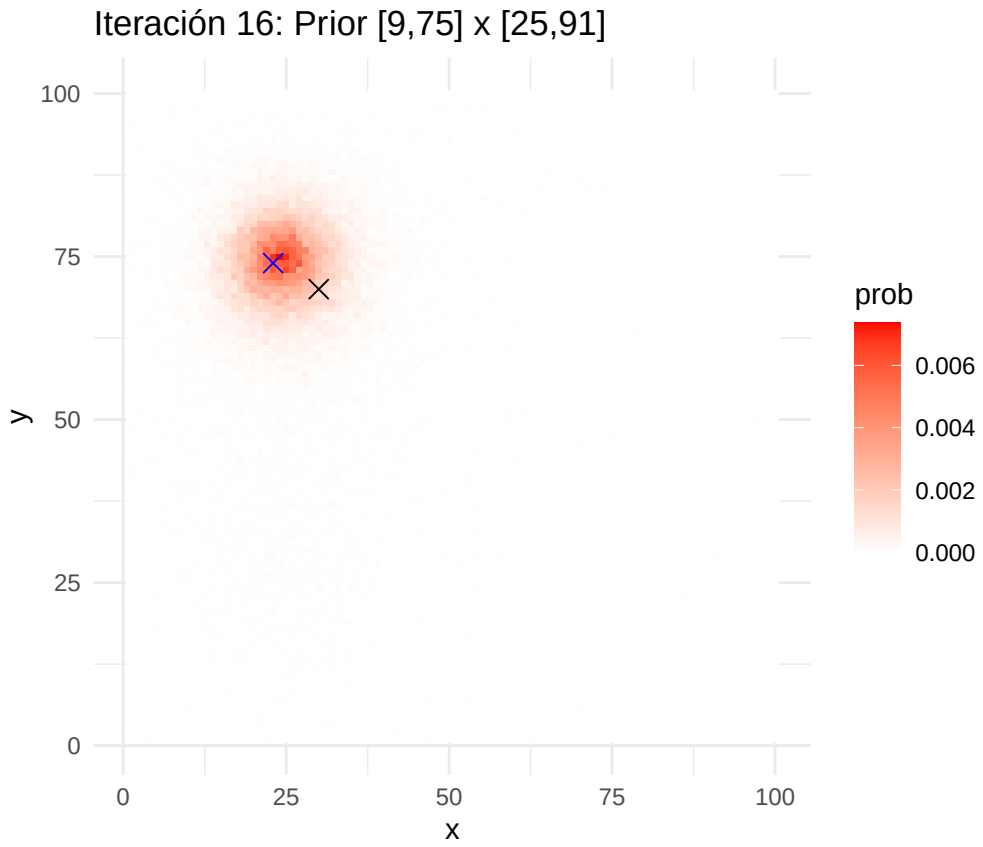

```
##
## Iteración: 13
## Coordenadas de observación: 24 74
## Intensidad observada = 0.9193
## Celda más probable = 24 75
## Probabilidad posterior  0.0077
## Prior actual: x  [ 6 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



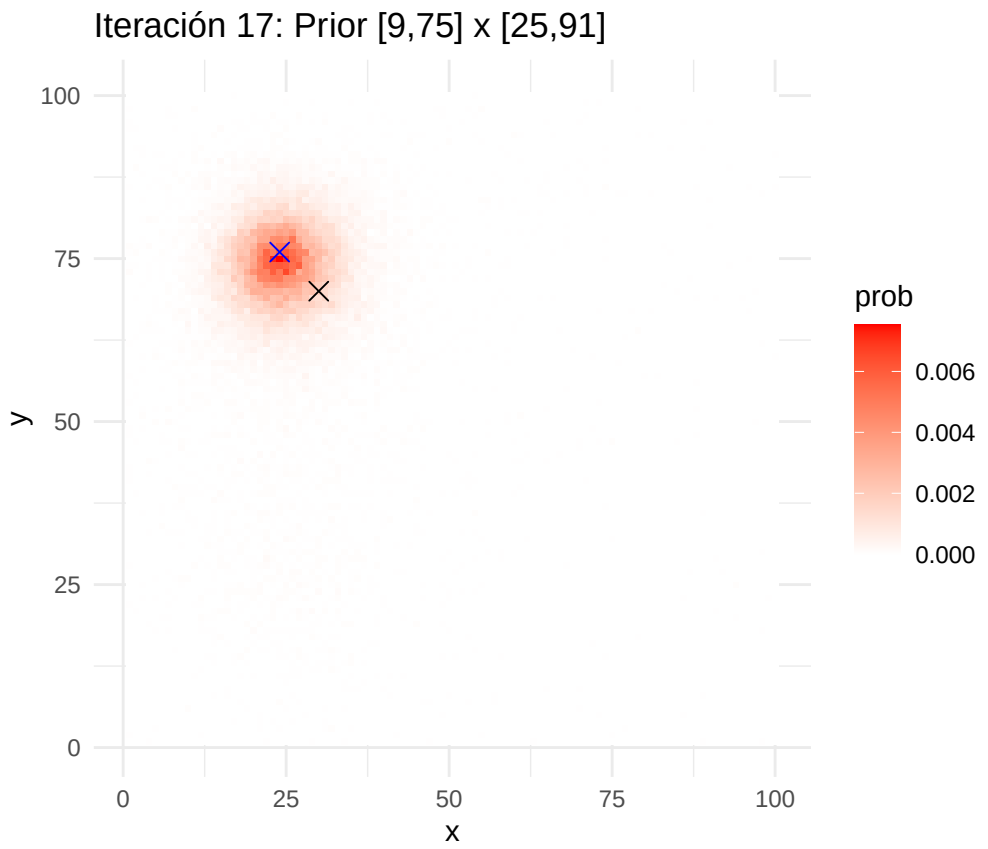

```
##
## Iteración: 14
## Coordenadas de observación: 23 75
## Intensidad observada = 0.9114
## Celda más probable = 24 75
## Probabilidad posterior  0.0074
## Prior actual: x  [ 6 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



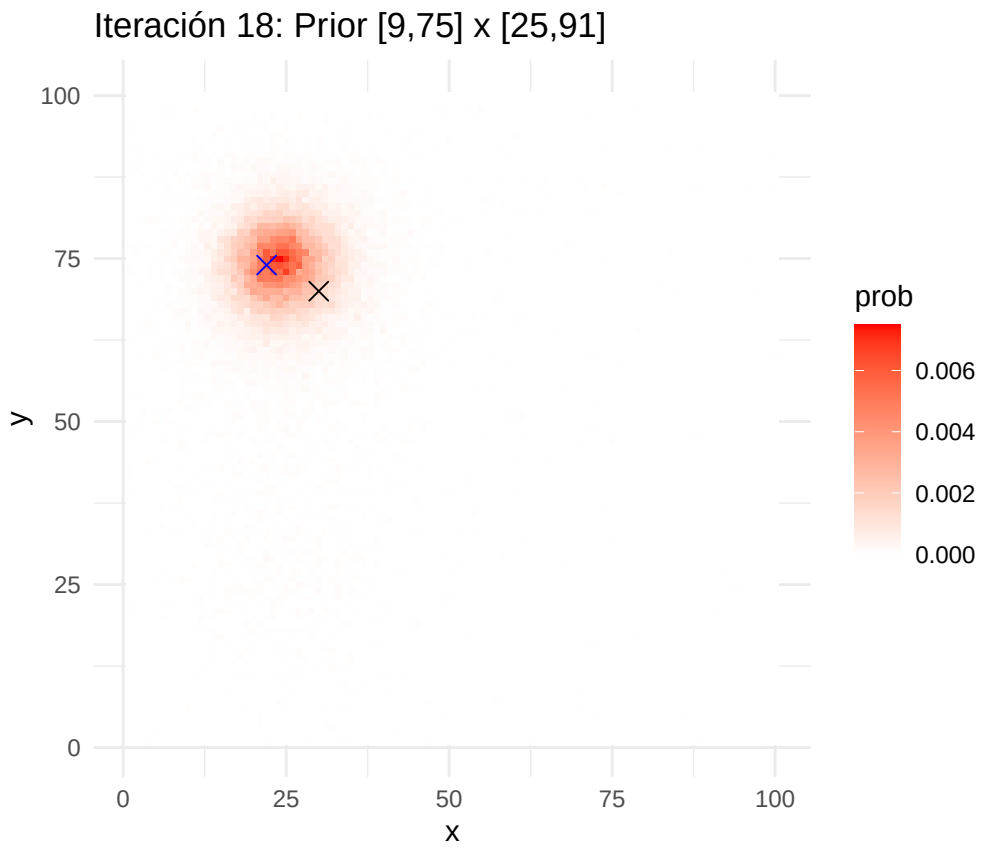

```
##
## Iteración: 15
## Coordenadas de observación: 23 73
## Intensidad observada = 1.4806
## Celda más probable = 24 75
## Probabilidad posterior  0.0074
## Prior actual: x  [ 7 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



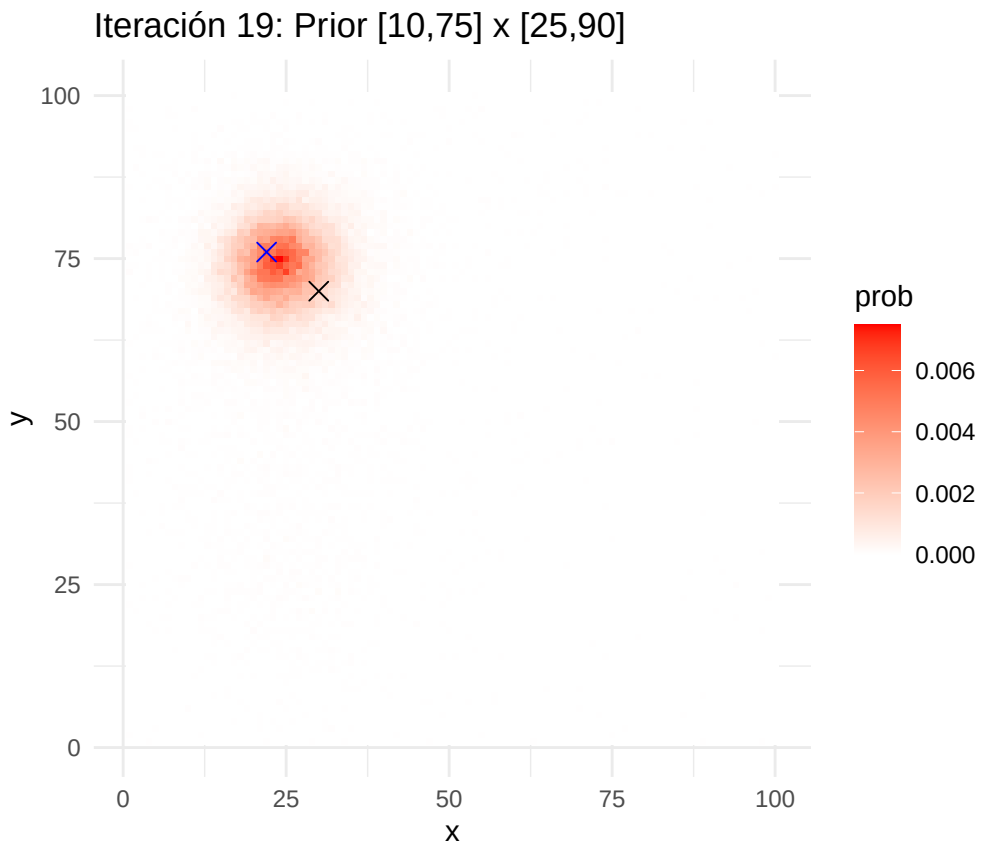

```
##
## Iteración: 16
## Coordenadas de observación: 23 74
## Intensidad observada = 0.877
## Celda más probable = 24 75
## Probabilidad posterior  0.0074
## Prior actual: x  [ 9 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



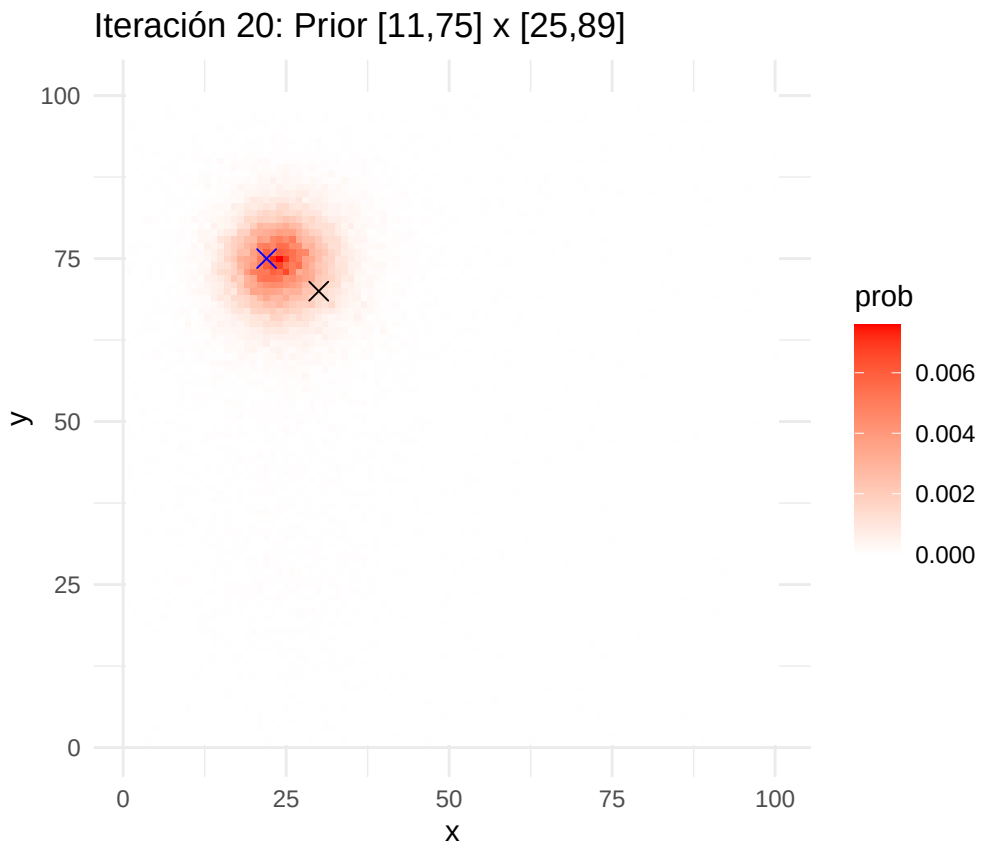

```
##
## Iteración: 17
## Coordenadas de observación: 24 76
## Intensidad observada = 0.5934
## Celda más probable = 24 75
## Probabilidad posterior  0.0075
## Prior actual: x  [ 9 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



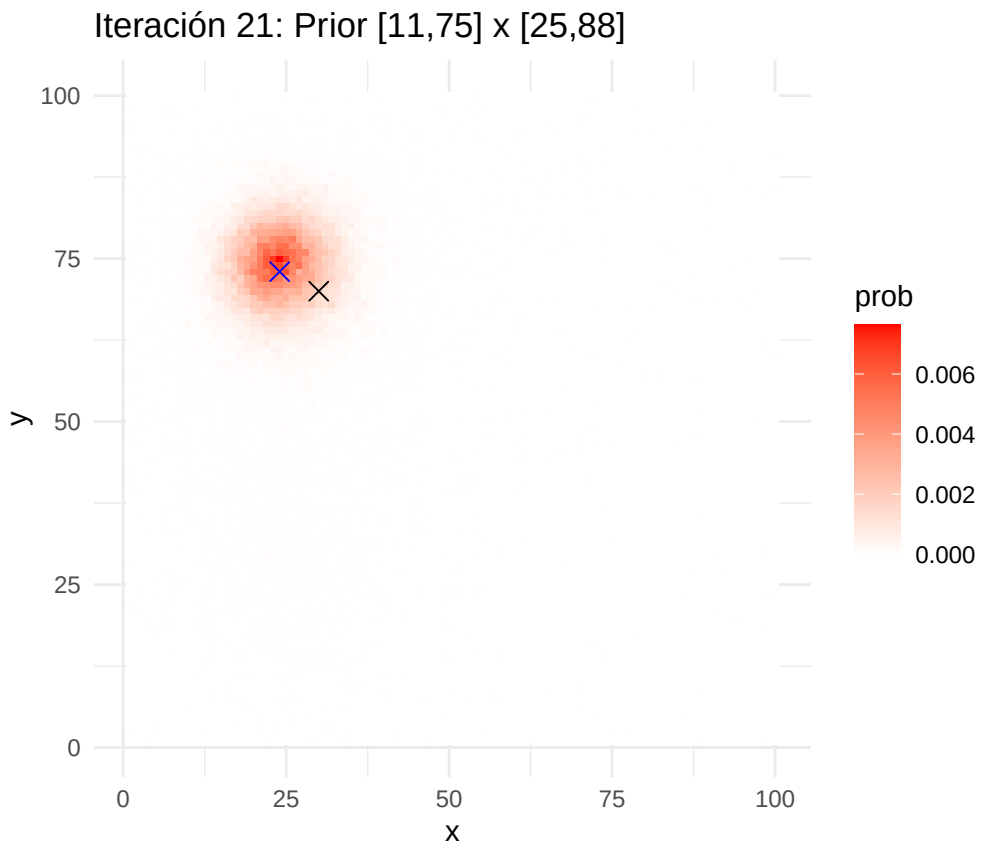

```
##
## Iteración: 18
## Coordenadas de observación: 22 74
## Intensidad observada = 0.877
## Celda más probable = 24 75
## Probabilidad posterior  0.0075
## Prior actual: x  [ 9 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.1 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



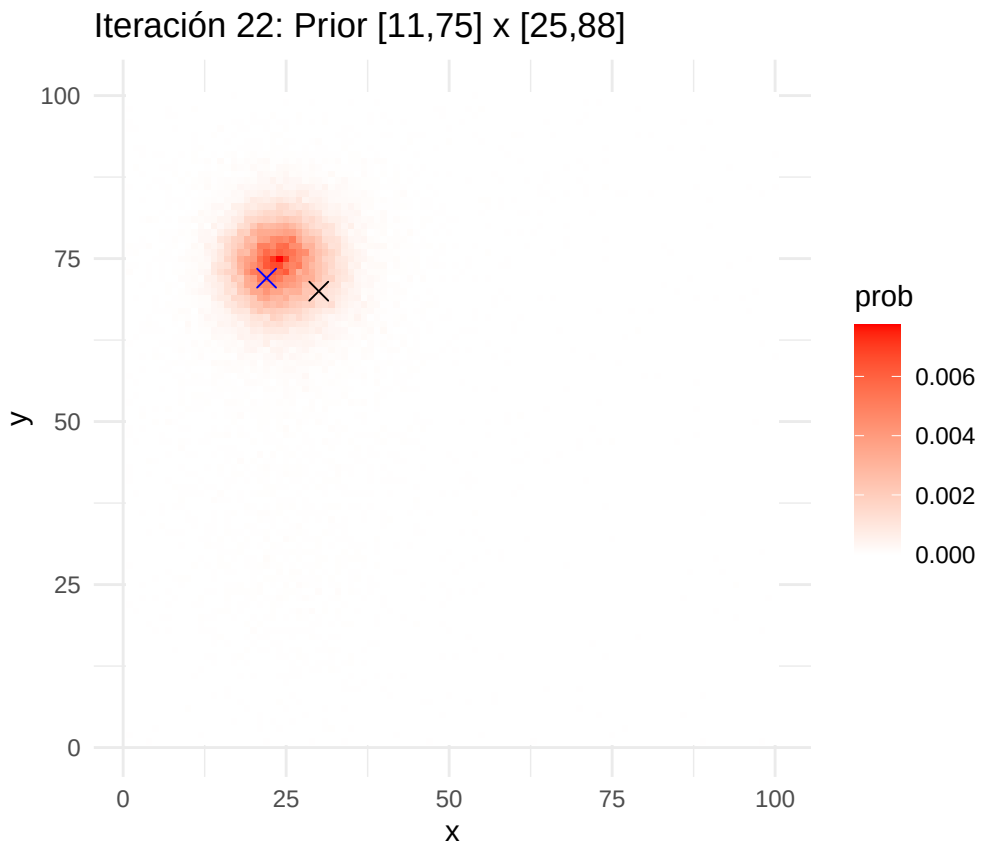

```
##
## Iteración: 19
## Coordenadas de observación: 22 76
## Intensidad observada = 0.3961
## Celda más probable = 24 75
## Probabilidad posterior  0.0075
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 90 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



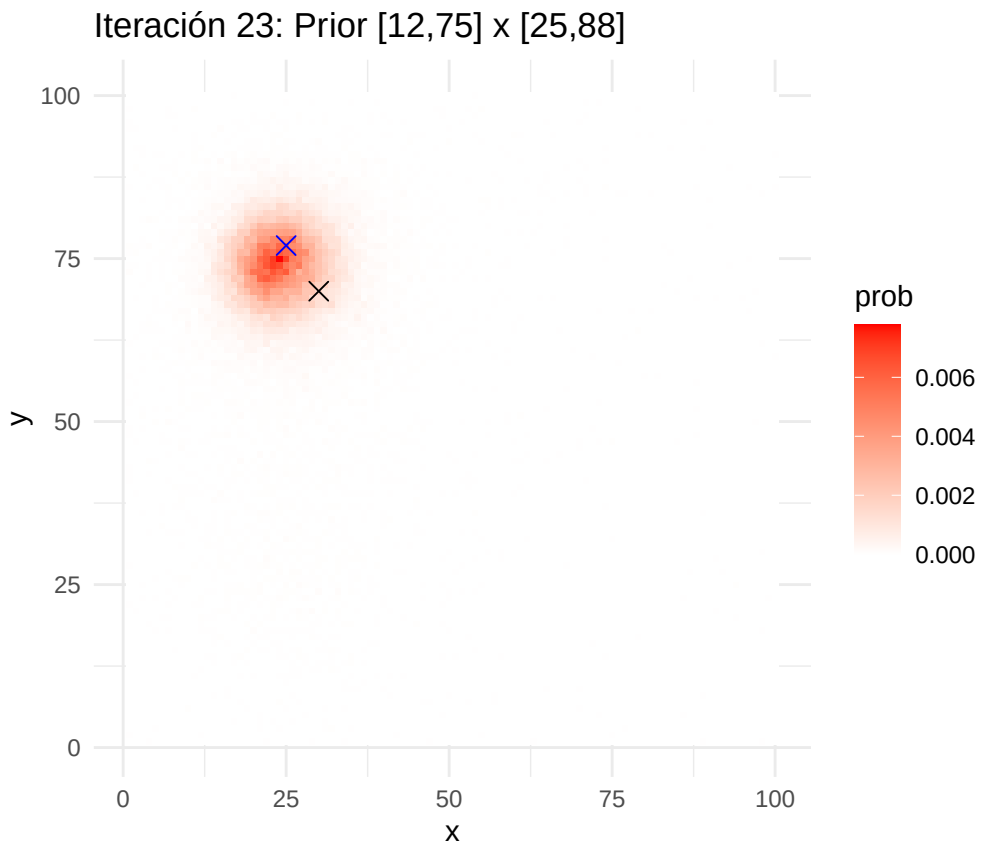

```
##
## Iteración: 20
## Coordenadas de observación: 22 75
## Intensidad observada = 1.0698
## Celda más probable = 24 75
## Probabilidad posterior  0.0076
## Prior actual: x  [ 11 , 75 ], y  [ 25 , 89 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



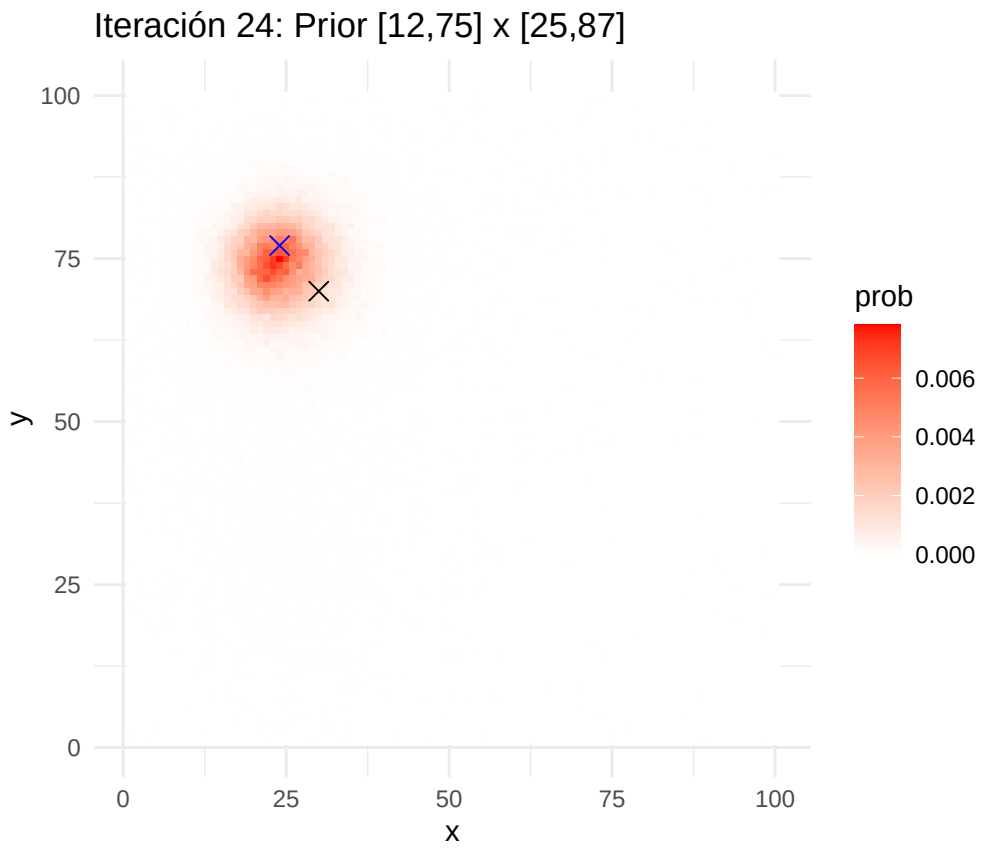

```
##
## Iteración: 21
## Coordenadas de observación: 24 73
## Intensidad observada = 0.3023
## Celda más probable = 24 75
## Probabilidad posterior  0.0076
## Prior actual: x  [ 11 , 75 ], y  [ 25 , 88 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



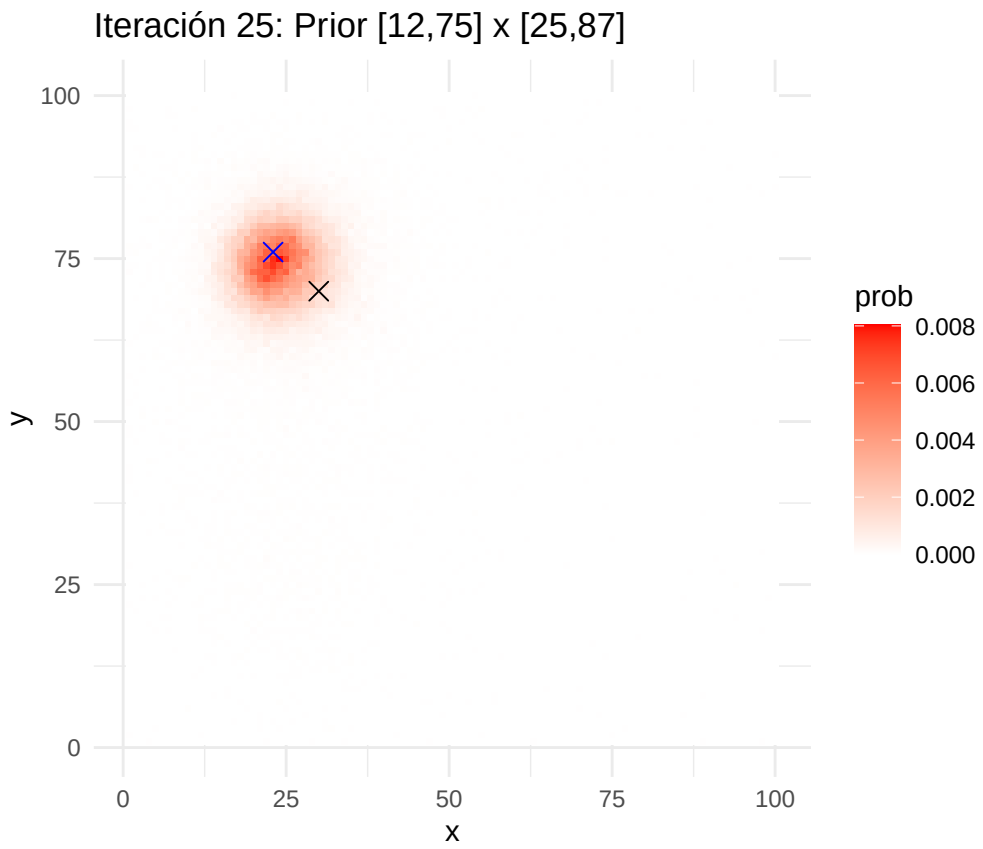

```
##
## Iteración: 22
## Coordenadas de observación: 22 72
## Intensidad observada = 1.3533
## Celda más probable = 24 75
## Probabilidad posterior  0.0078
## Prior actual: x  [ 11 , 75 ], y  [ 25 , 88 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 23
## Coordenadas de observación: 25 77
## Intensidad observada = 1.2219
## Celda más probable = 24 75
## Probabilidad posterior  0.0078
## Prior actual: x  [ 12 , 75 ], y  [ 25 , 88 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 24
## Coordenadas de observación: 24 77
## Intensidad observada = 0.9116
## Celda más probable = 24 75
## Probabilidad posterior  0.0078
## Prior actual: x  [ 12 , 75 ], y  [ 25 , 87 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 25
## Coordenadas de observación: 23 76
## Intensidad observada = 0.8954
## Celda más probable = 24 75
## Probabilidad posterior  0.008
## Prior actual: x  [ 12 , 75 ], y  [ 25 , 87 ]
```

Con este nivel de sigma el ruido es muy intenso. Al modelo le tiene que costar mucho encontrar el objeto y seguramente no pueda ni llegar acotar la prior.

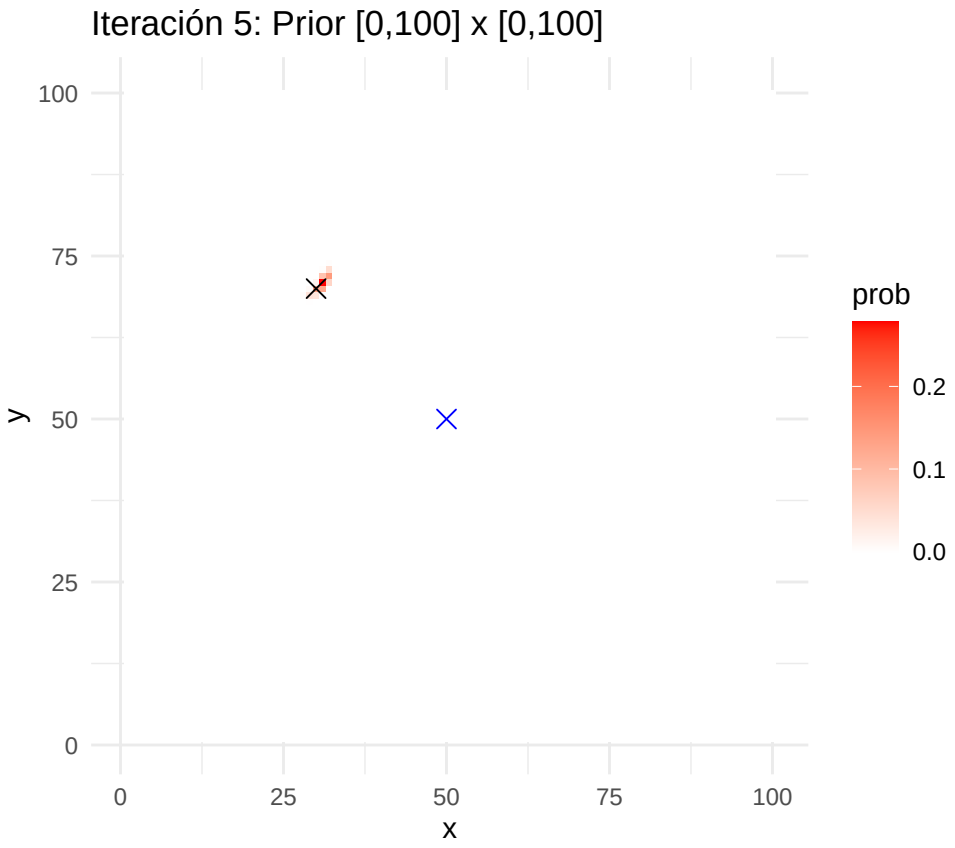
1.3

El nivel de sigma va ser de 0,01.

```
sim3 <- localiza_objeto(true_x = 30, true_y = 70,
                        A = 1, lambda = 20, sigma = 0.01,
                        n_obs = 25, mostrar_plot = TRUE,
                        guardar_datos = TRUE)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
```

```
## Chain 3 finished in 0.1 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 5
## Coordenadas de observación: 50 50
## Intensidad observada = 0.2521
## Celda más probable = 31 71
## Probabilidad posterior  0.278
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## El modelo ha localizado el objeto en la iteración 5
```

Para este nivel el modelo tiene que encontrar rápidamente el objeto.

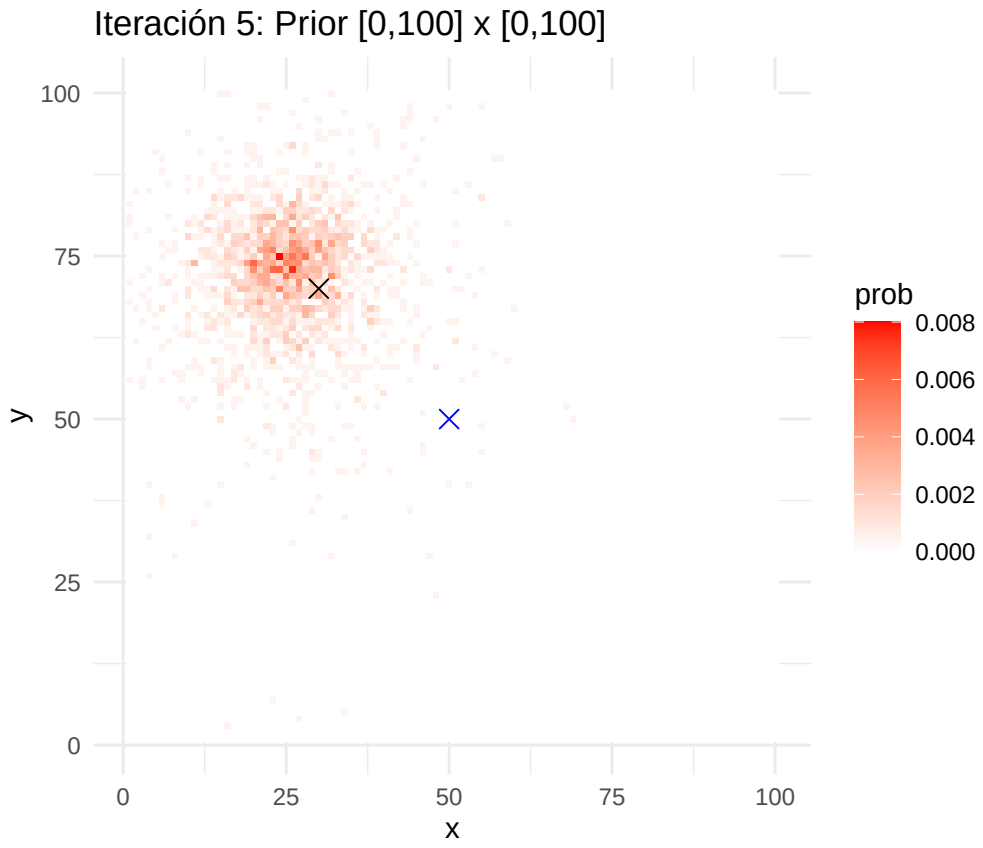
1.4

En esta ocasión el valor de sigma 0,03.

```
sim4 <- localiza_objeto(true_x = 30, true_y = 70,
                            A = 1, lambda = 20, sigma = 0.3,
```

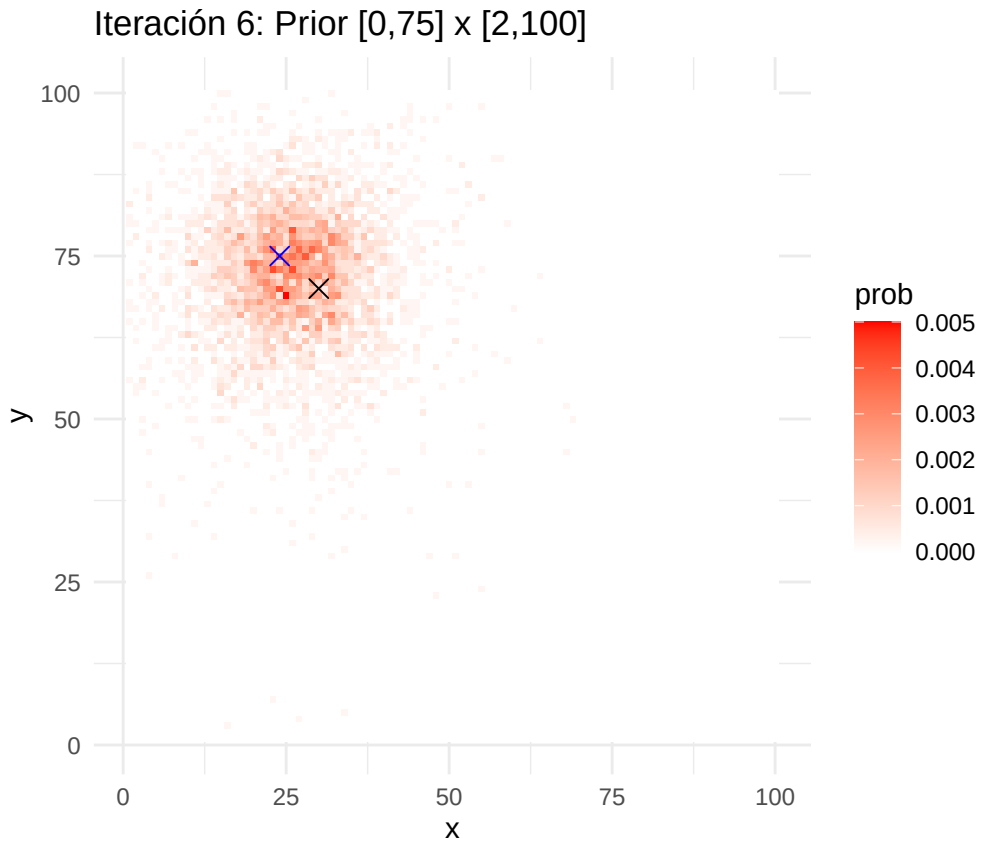
```
                        n_obs = 25, mostrar_plot = TRUE,
                        guardar_datos = TRUE)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```

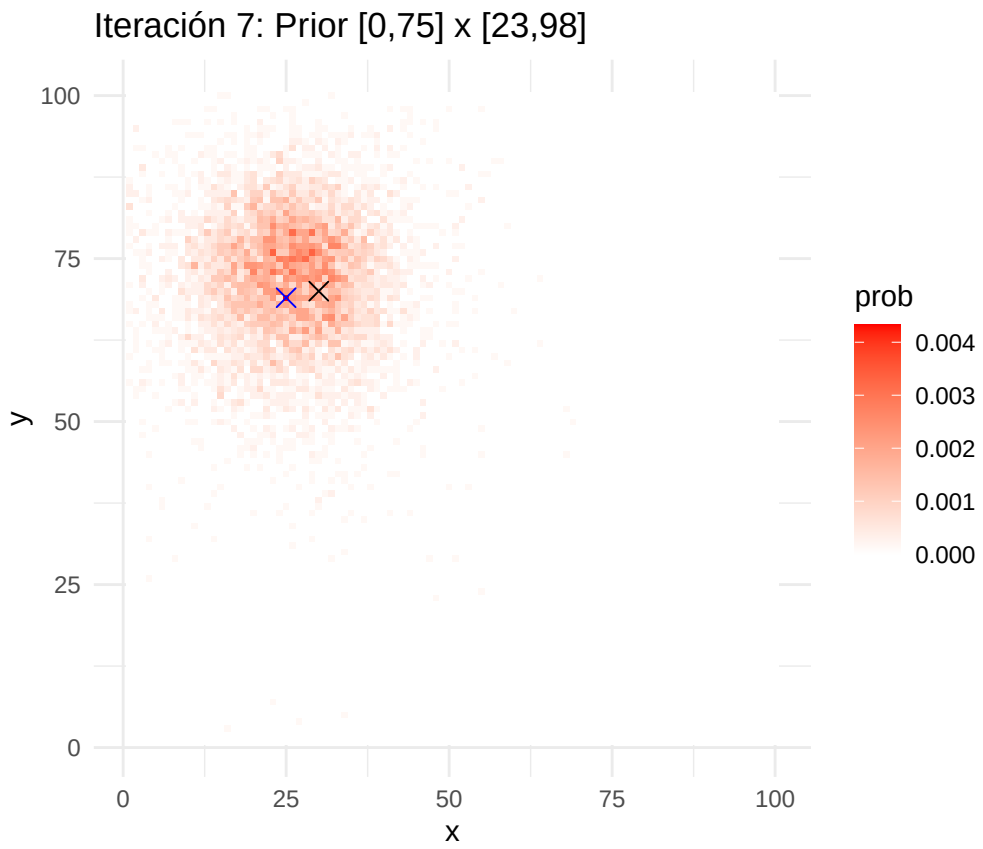



```
##
## Iteración: 5
## Coordenadas de observación: 50 50
## Intensidad observada = 0.566
## Celda más probable = 24 75
## Probabilidad posterior  0.008
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
```

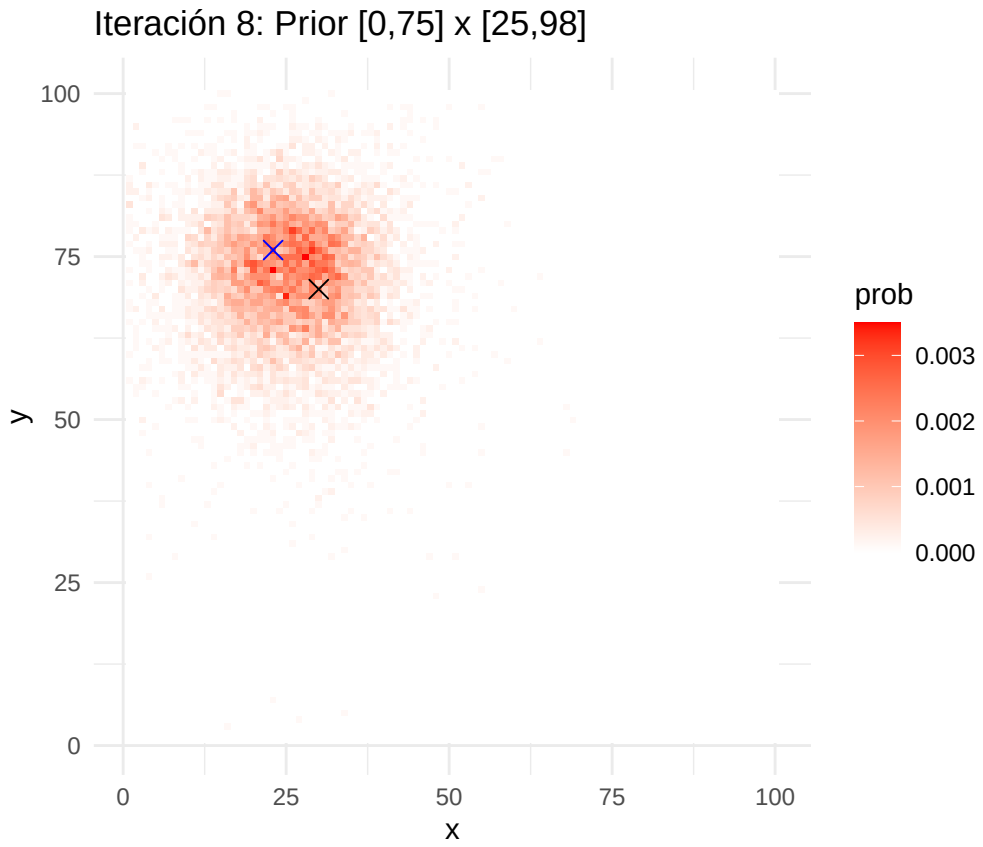
```
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



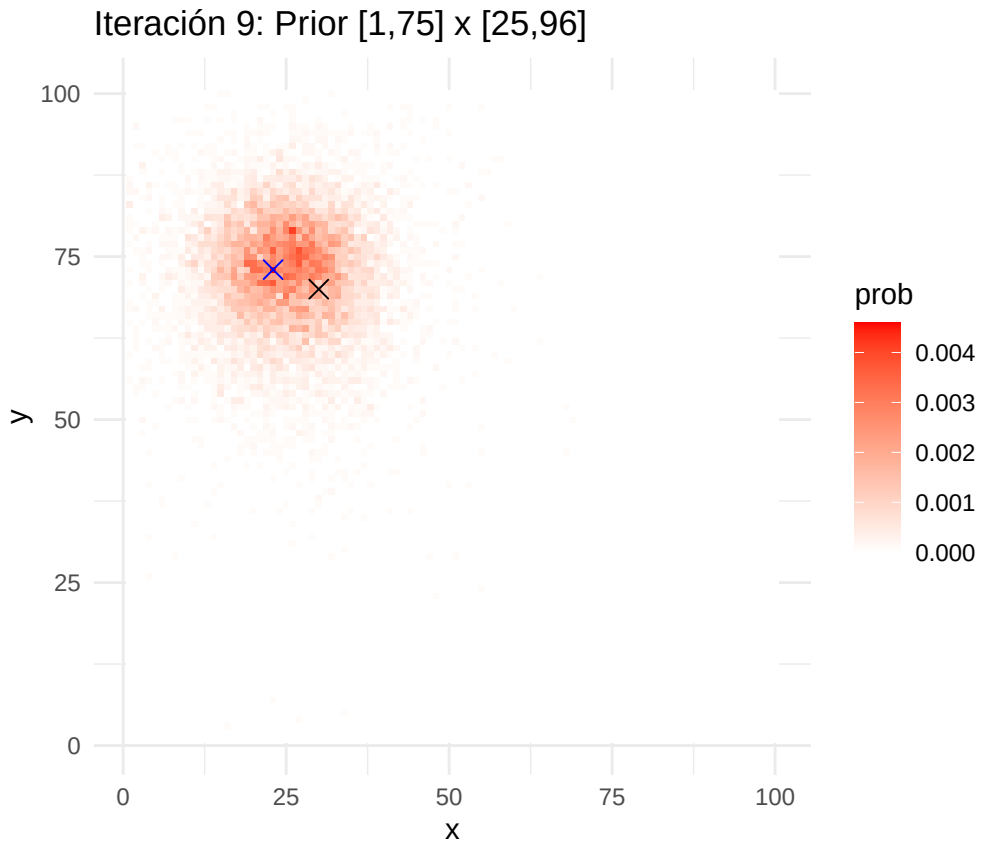

```
##
## Iteración: 6
## Coordenadas de observación: 24 75
## Intensidad observada = 0.4121
## Celda más probable = 25 69
## Probabilidad posterior  0.005
## Prior actual: x  [ 0 , 75 ], y  [ 2 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



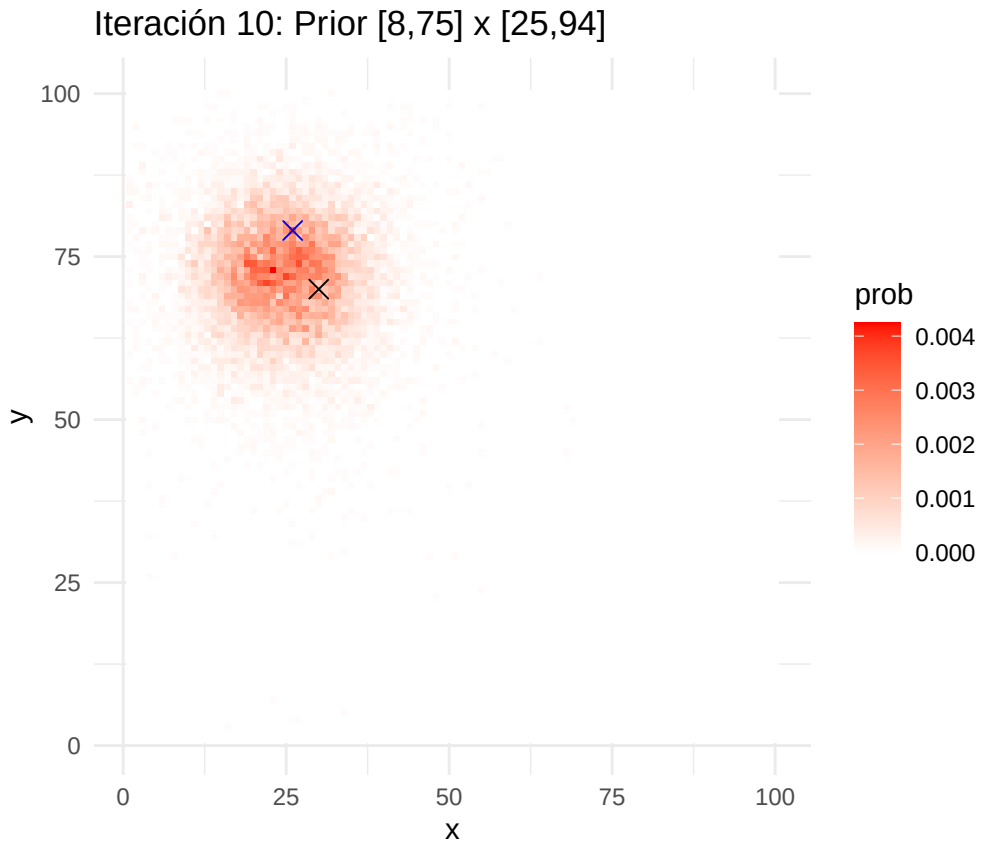

```
##
## Iteración: 7
## Coordenadas de observación: 25 69
## Intensidad observada = 0.6008
## Celda más probable = 25 69
## Probabilidad posterior  0.0043
## Prior actual: x  [ 0 , 75 ], y  [ 23 , 98 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



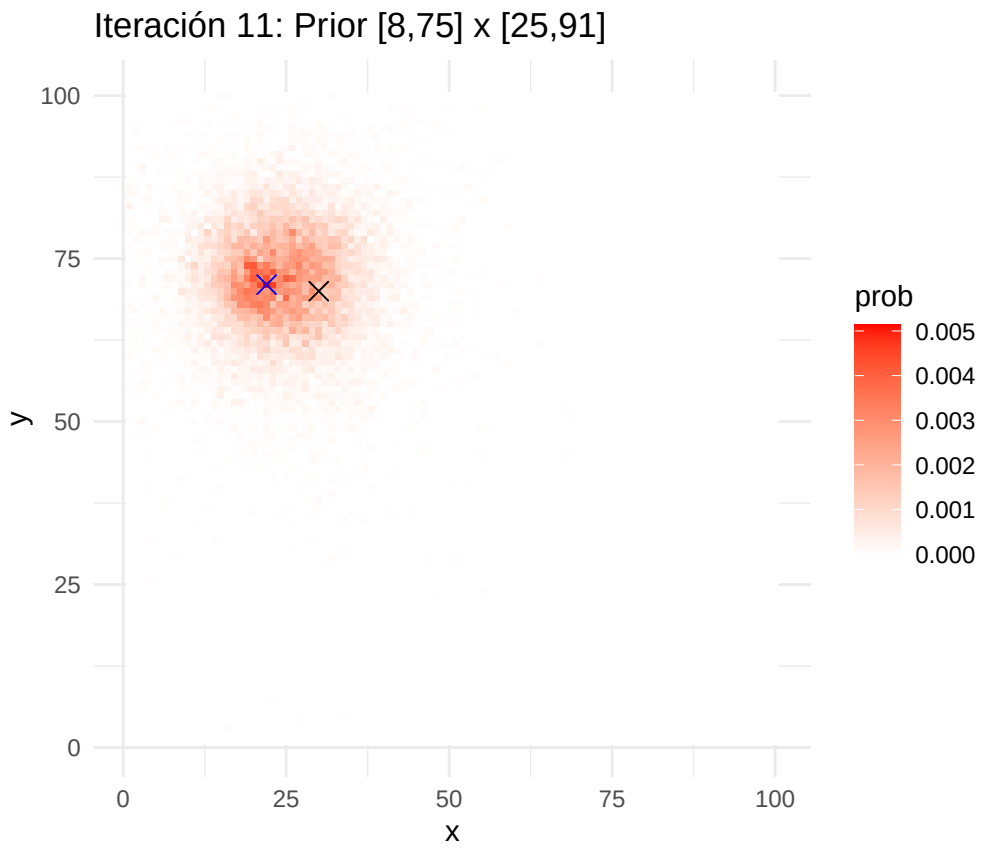

```
##
## Iteración: 8
## Coordenadas de observación: 23 76
## Intensidad observada = 0.8929
## Celda más probable = 23 73
## Probabilidad posterior  0.0035
## Prior actual: x  [ 0 , 75 ], y  [ 25 , 98 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



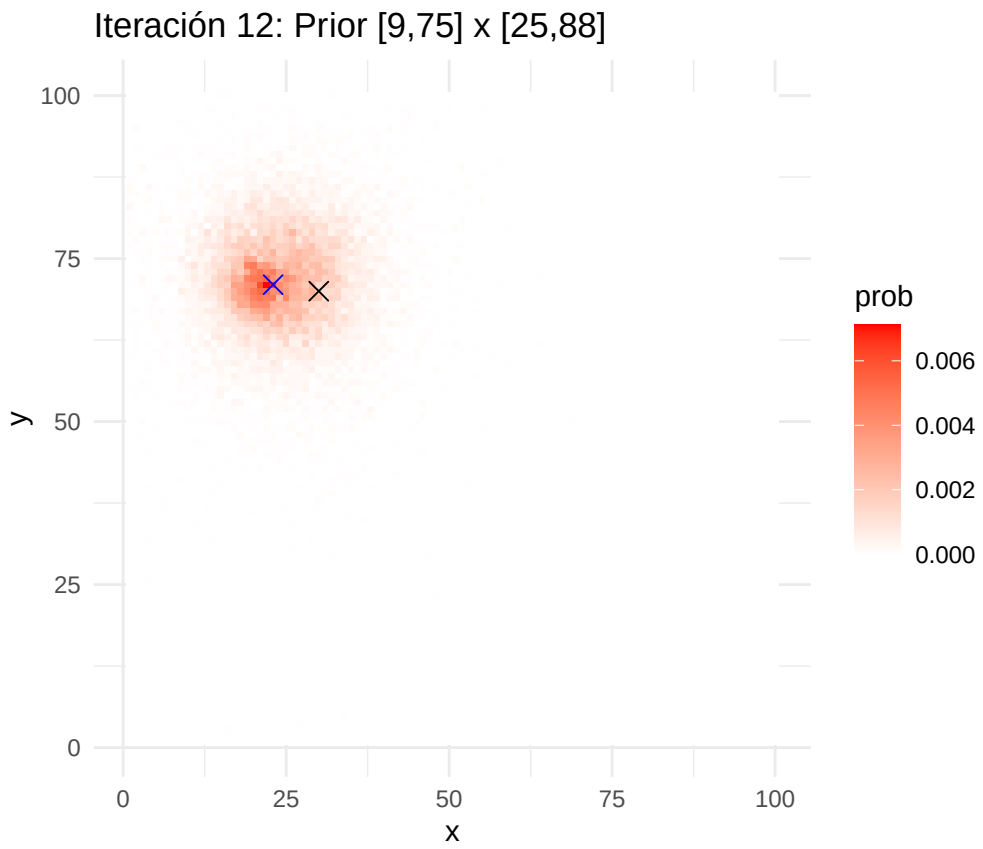

```
##
## Iteración: 9
## Coordenadas de observación: 23 73
## Intensidad observada = 1.1307
## Celda más probable = 23 73
## Probabilidad posterior  0.0046
## Prior actual: x  [ 1 , 75 ], y  [ 25 , 96 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



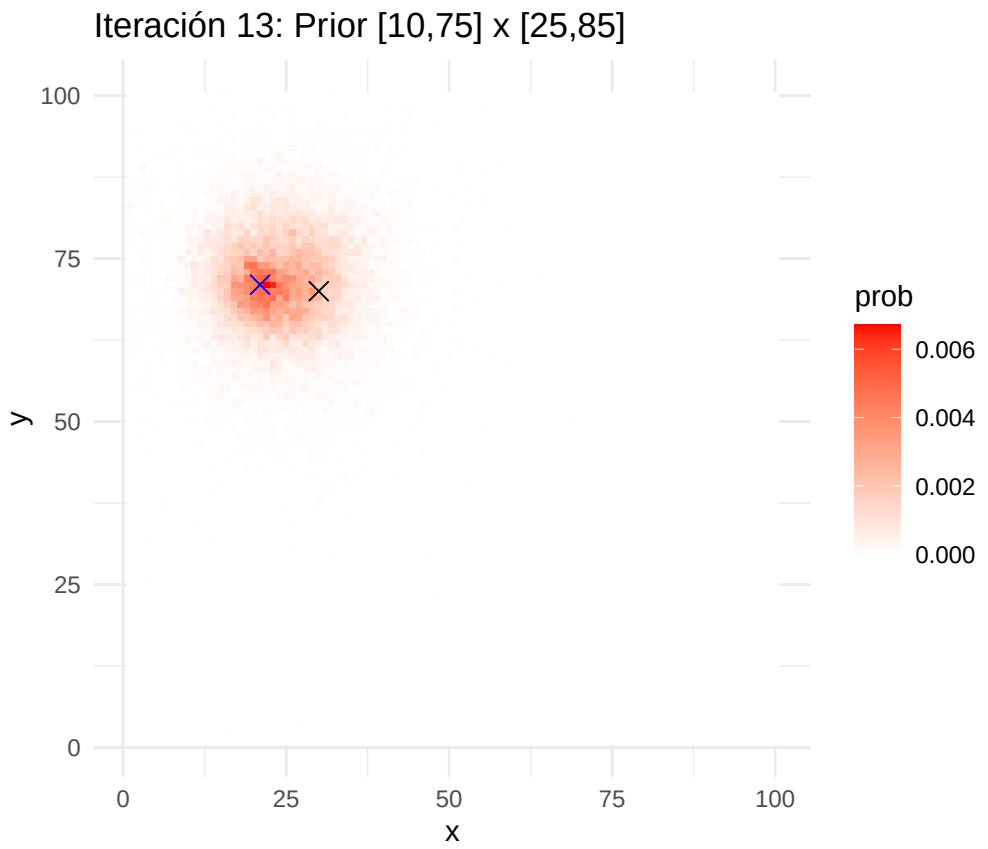

```
##
## Iteración: 10
## Coordenadas de observación: 26 79
## Intensidad observada = -0.1202
## Celda más probable = 23 73
## Probabilidad posterior  0.0043
## Prior actual: x  [ 8 , 75 ], y  [ 25 , 94 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



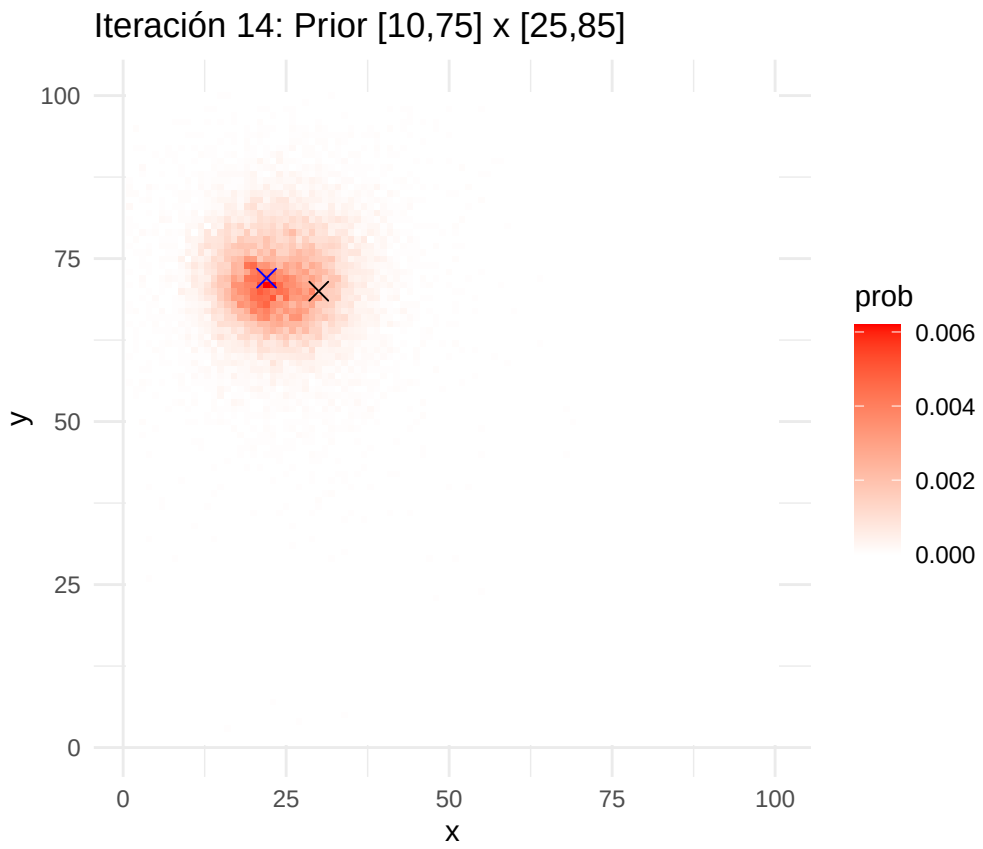

```
##
## Iteración: 11
## Coordenadas de observación: 22 71
## Intensidad observada = 1.1748
## Celda más probable = 22 71
## Probabilidad posterior  0.0051
## Prior actual: x  [ 8 , 75 ], y  [ 25 , 91 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



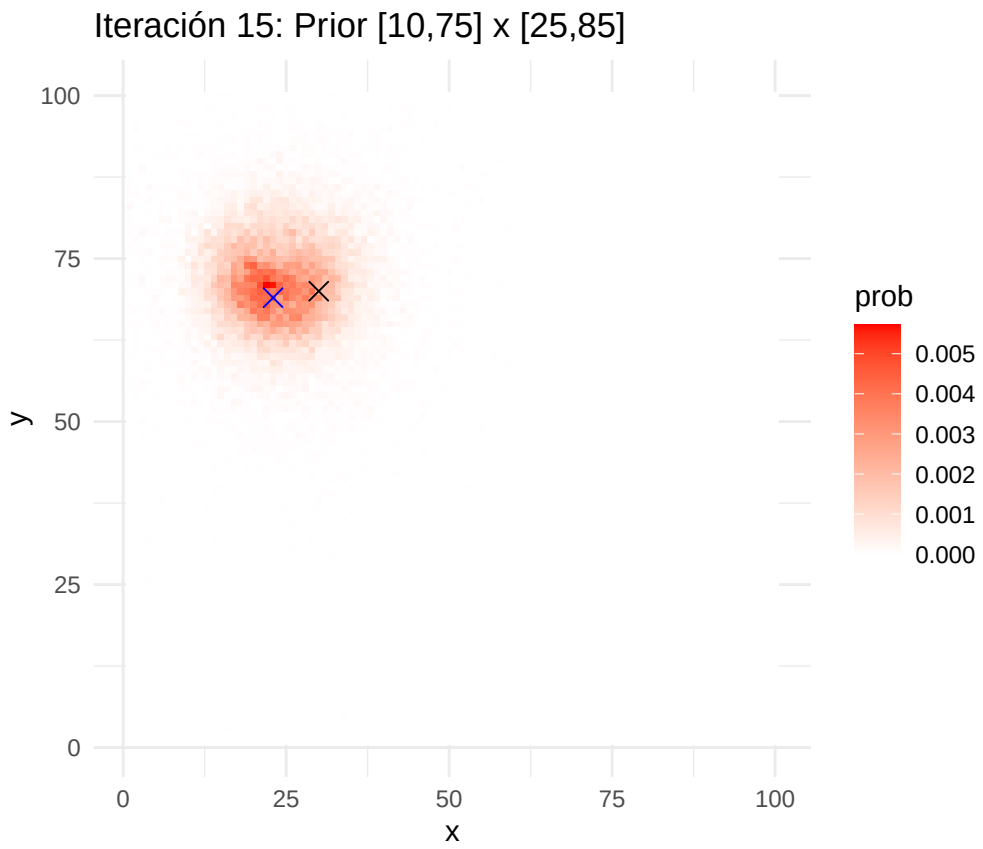

```
##
## Iteración: 12
## Coordenadas de observación: 23 71
## Intensidad observada = 1.1354
## Celda más probable = 22 71
## Probabilidad posterior  0.0071
## Prior actual: x  [ 9 , 75 ], y  [ 25 , 88 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



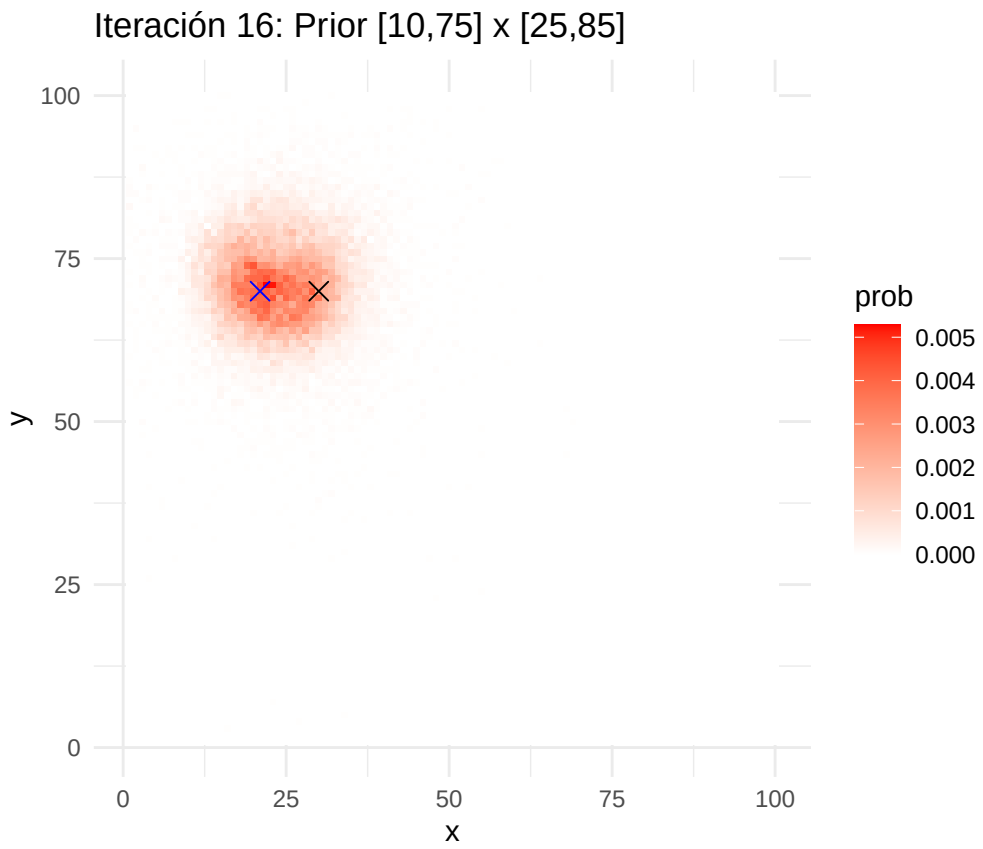

```
##
## Iteración: 13
## Coordenadas de observación: 21 71
## Intensidad observada = 0.0465
## Celda más probable = 22 71
## Probabilidad posterior  0.0067
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



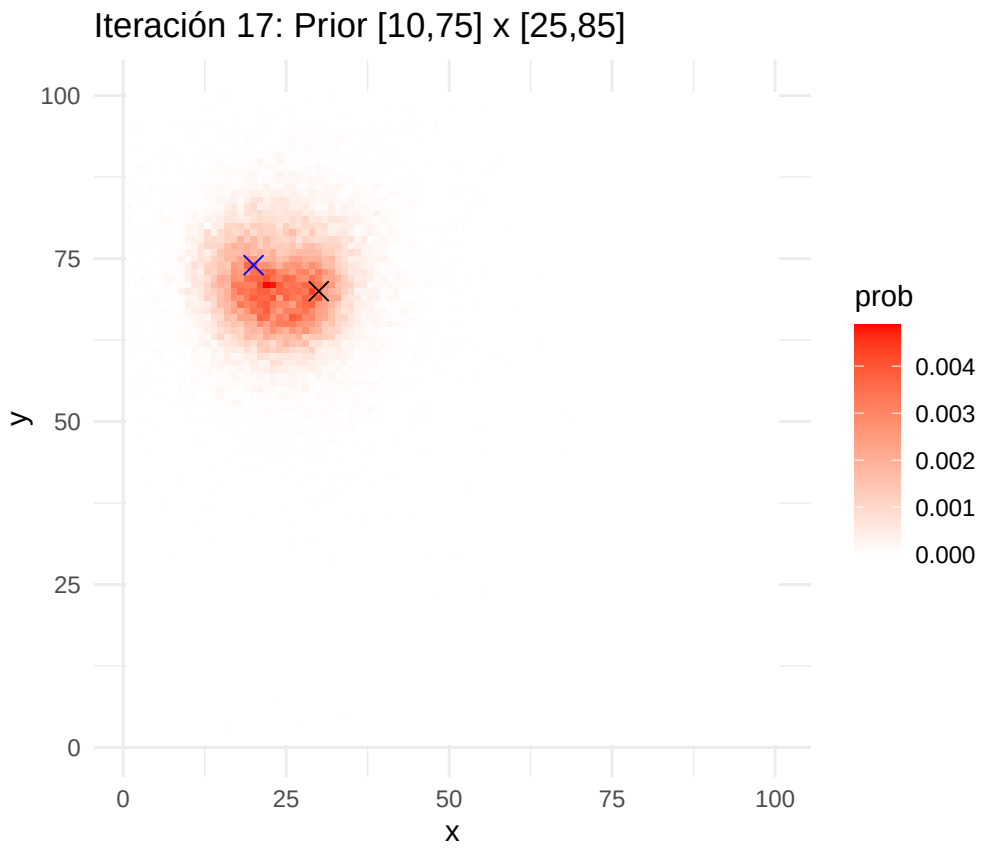

```
##
## Iteración: 14
## Coordenadas de observación: 22 72
## Intensidad observada = 0.5835
## Celda más probable = 22 71
## Probabilidad posterior  0.0062
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



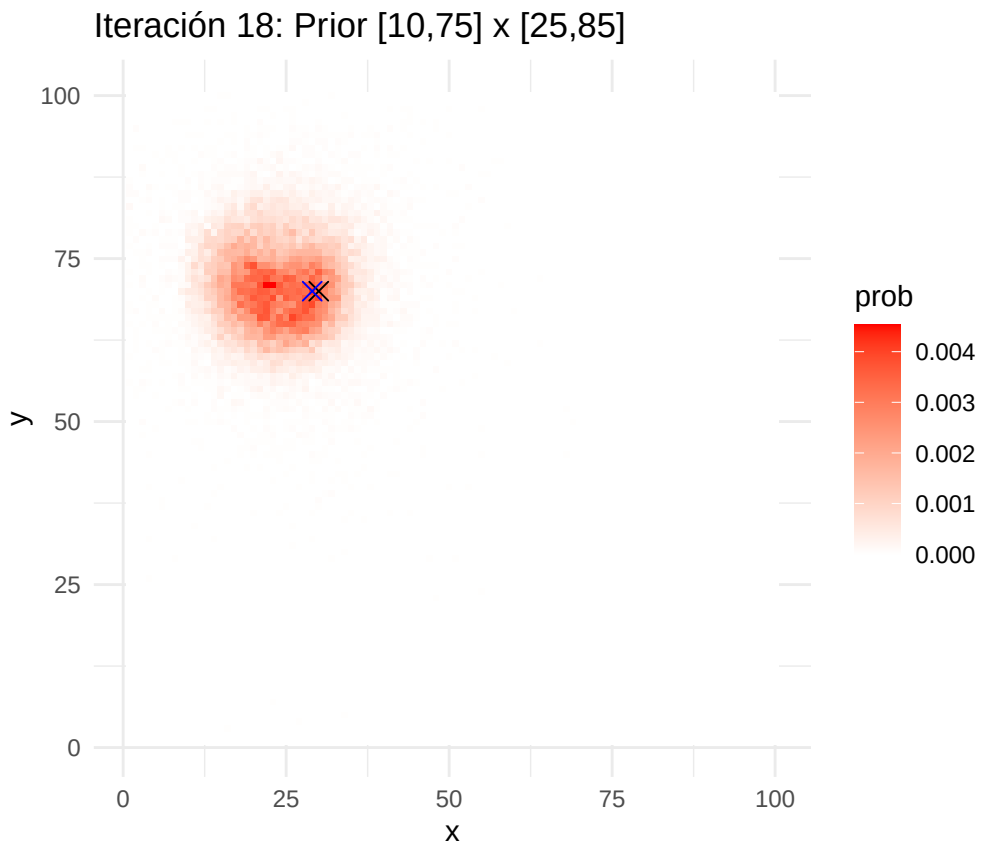

```
##
## Iteración: 15
## Coordenadas de observación: 23 69
## Intensidad observada = 0.4247
## Celda más probable = 22 71
## Probabilidad posterior  0.0057
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



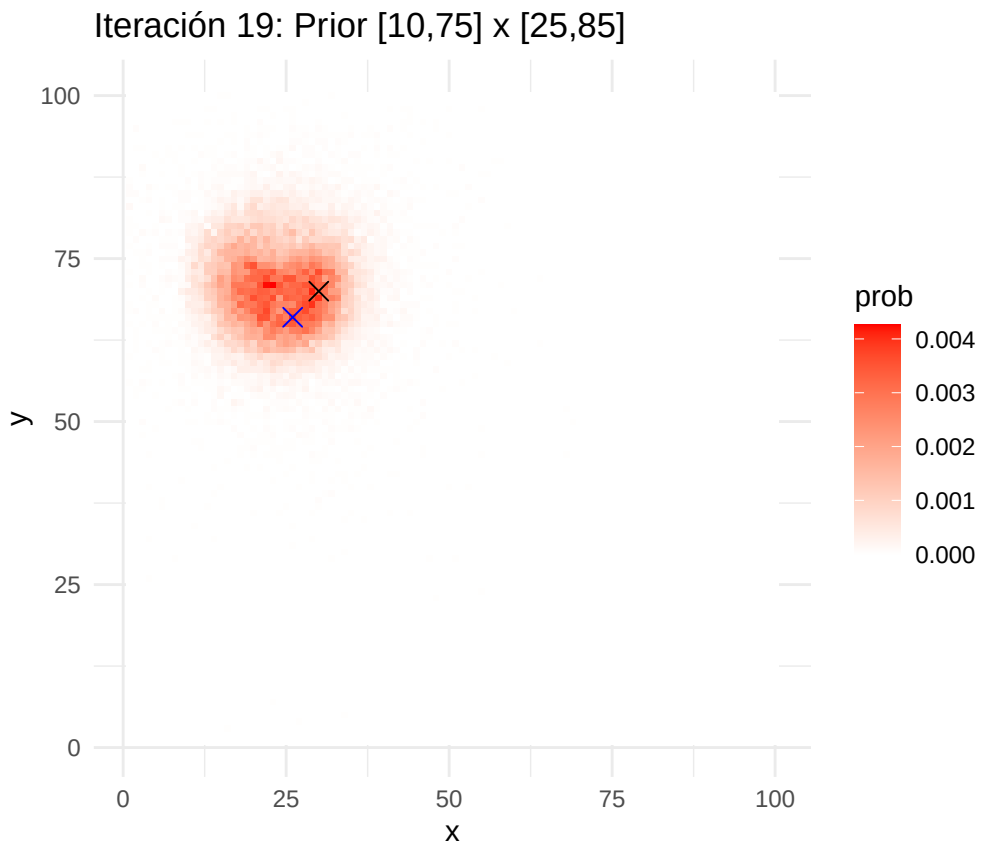

```
##
## Iteración: 16
## Coordenadas de observación: 21 70
## Intensidad observada = 0.6923
## Celda más probable = 22 71
## Probabilidad posterior  0.0053
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



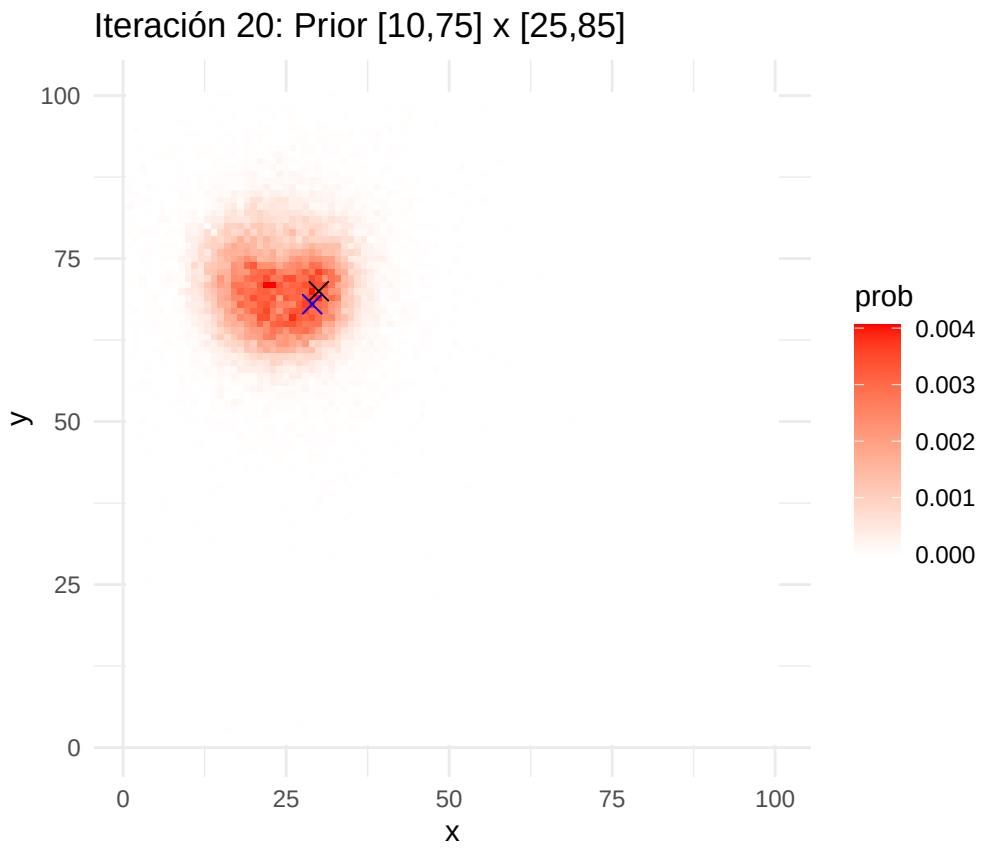

```
##
## Iteración: 17
## Coordenadas de observación: 20 74
## Intensidad observada = 0.2026
## Celda más probable = 22 71
## Probabilidad posterior  0.0049
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



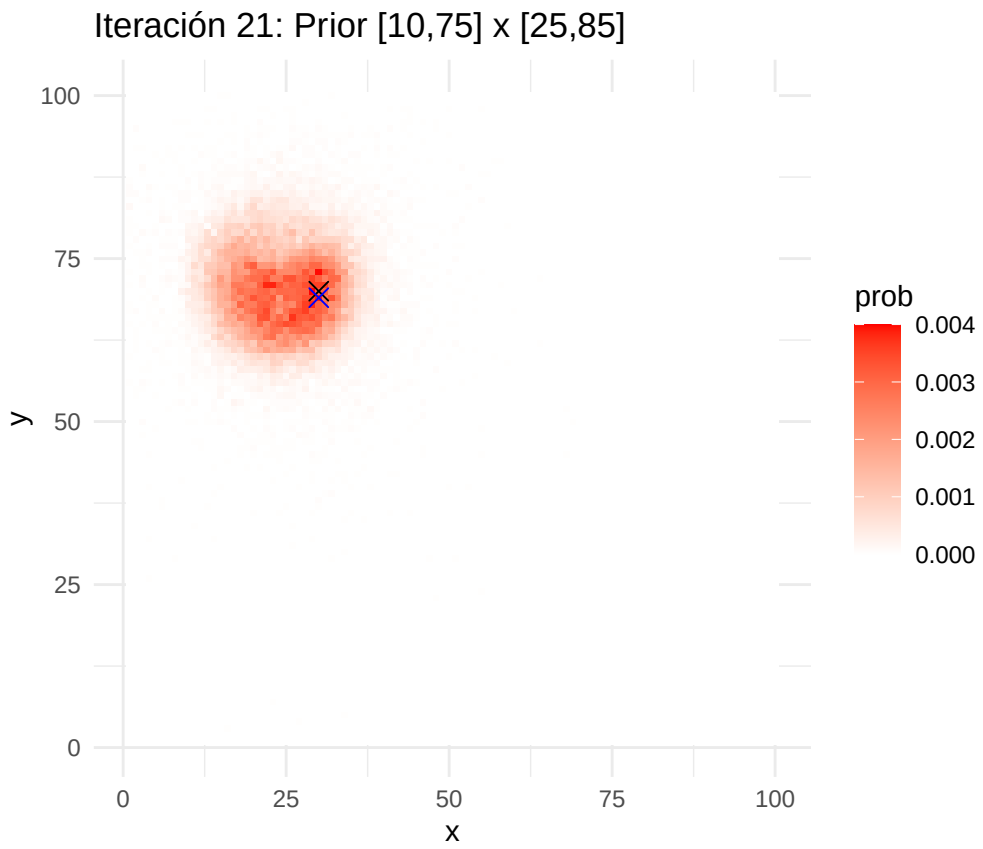

```
##
## Iteración: 18
## Coordenadas de observación: 29 70
## Intensidad observada = 0.6842
## Celda más probable = 22 71
## Probabilidad posterior  0.0045
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.1 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 19
## Coordenadas de observación: 26 66
## Intensidad observada = 0.458
## Celda más probable = 23 71
## Probabilidad posterior  0.0043
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.1 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.2 seconds.
```




```
##
## Iteración: 20
## Coordenadas de observación: 29 68
## Intensidad observada = 0.6853
## Celda más probable = 22 71
## Probabilidad posterior  0.0041
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.1 seconds.
## Chain 2 finished in 0.1 seconds.
## Chain 3 finished in 0.1 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.1 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 21
## Coordenadas de observación: 30 69
## Intensidad observada = 0.773
## Celda más probable = 29 70
## Probabilidad posterior  0.004
## Prior actual: x  [ 10 , 75 ], y  [ 25 , 85 ]
## El modelo ha localizado el objeto en la iteración 21
```

Con estos valores modelo le debería costar pero en algunas ocasiones consigue encontrarlo e ir acotando las prior.

En general podemos decir que el modelo, con el resto de valores constantes, es muy sensible a los mínimos aumentos y mínimos descensos de los valores de sigma.

2. Grupo de simulaciones.

En este grupo de simulaciones vamos a probar distintos valores de Lamba.

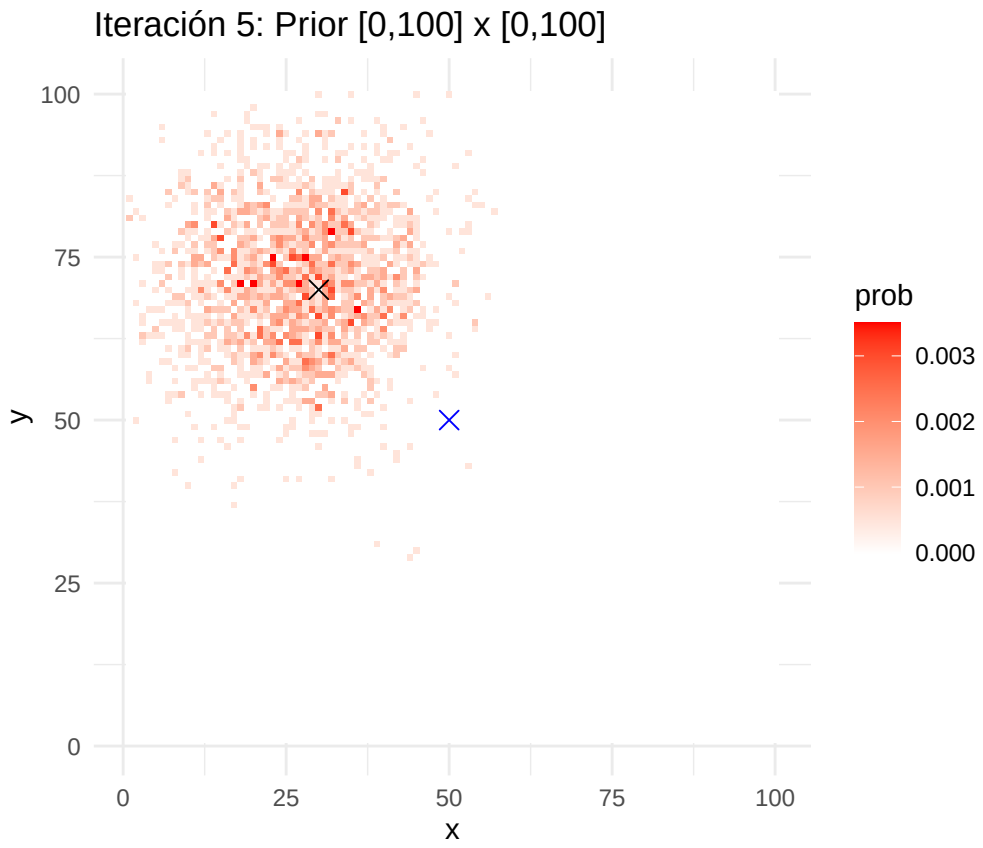
Por lo general valores más altos de lambda significan mejores detecciones a larga distancia pero peor precisión de cerca. Valores bajos serían justamente lo contrarío peores detecciones a larga distancia pero mejor precisión de cerca.

2.1

Aquí usaremos lambda = 30

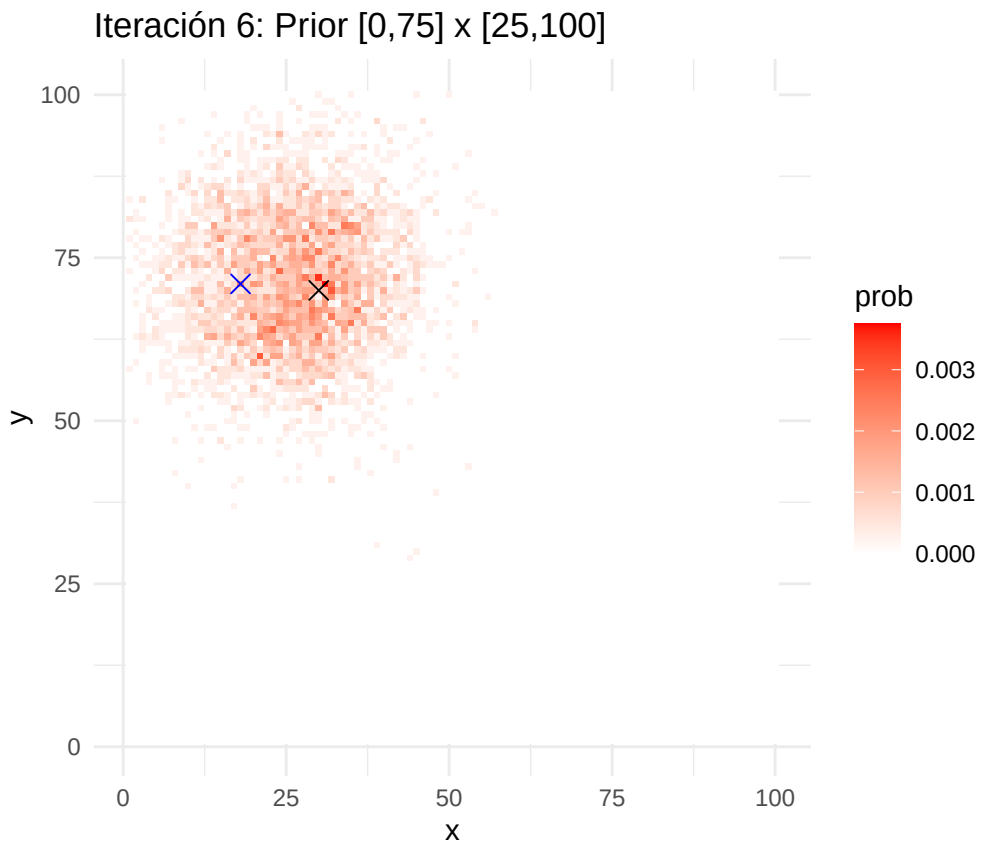
```
sim5 <- localiza_objeto(true_x = 30, true_y = 70,
                        A = 1, lambda = 30, sigma = 0.2,
                        n_obs = 25, mostrar_plot = TRUE,
                        guardar_datos = TRUE)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 5
## Coordenadas de observación: 50 50
## Intensidad observada = 0.5125
## Celda más probable = 18 71
## Probabilidad posterior  0.0035
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
```

```
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 6
## Coordenadas de observación: 18 71
## Intensidad observada = 0.7091
## Celda más probable = 31 71
## Probabilidad posterior  0.0037
## Prior actual: x   [ 0 , 75 ], y   [ 25 , 100 ]
## El modelo ha localizado el objeto en la iteración 6
```

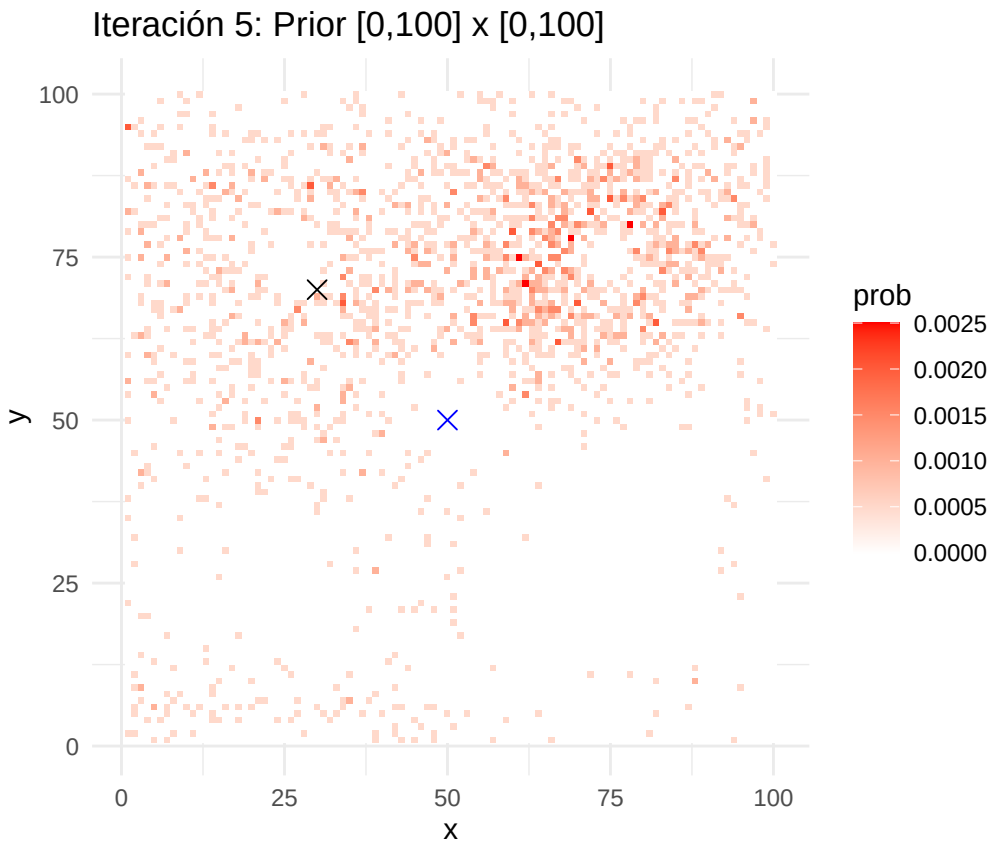
Con este valor el modelo suele quedarse muy cerca de encontrar el objeto. Llega rápidamente a las cercanías del objeto pero no tiene la precisión suficiente para dar el último paso, aunque en ocasiones si lo consigue

2.2

Lambda = 10

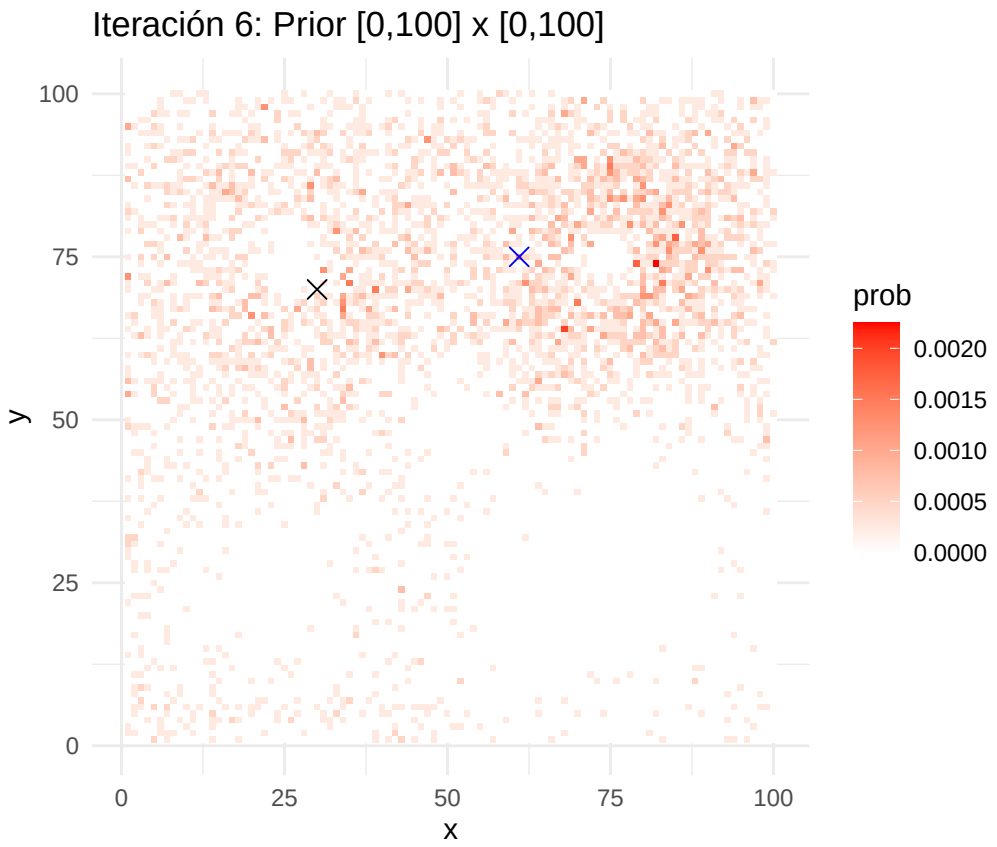
```
sim6 <- localiza_objeto(true_x = 30, true_y = 70,
                        A = 1, lambda = 10, sigma = 0.2,
                        n_obs = 25, mostrar_plot = TRUE,
                        guardar_datos = TRUE)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



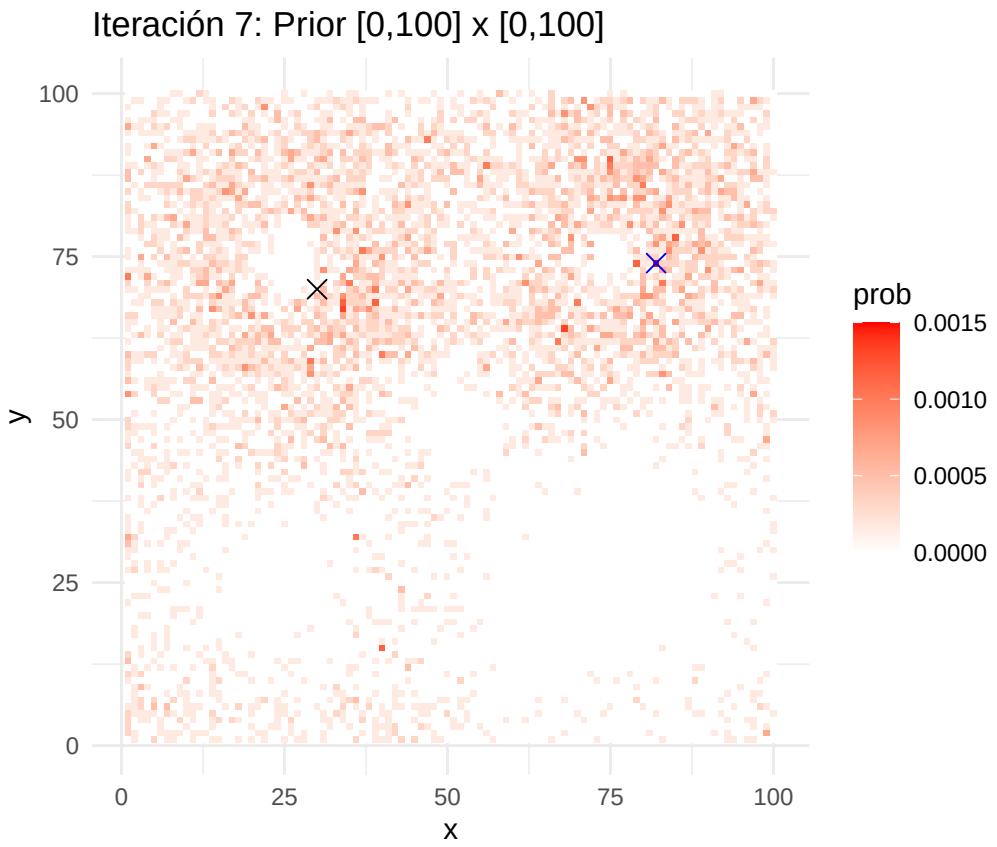

```
##
## Iteración: 5
## Coordenadas de observación: 50 50
## Intensidad observada = 0.1835
## Celda más probable = 61 75
## Probabilidad posterior  0.0025
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
```

```
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```

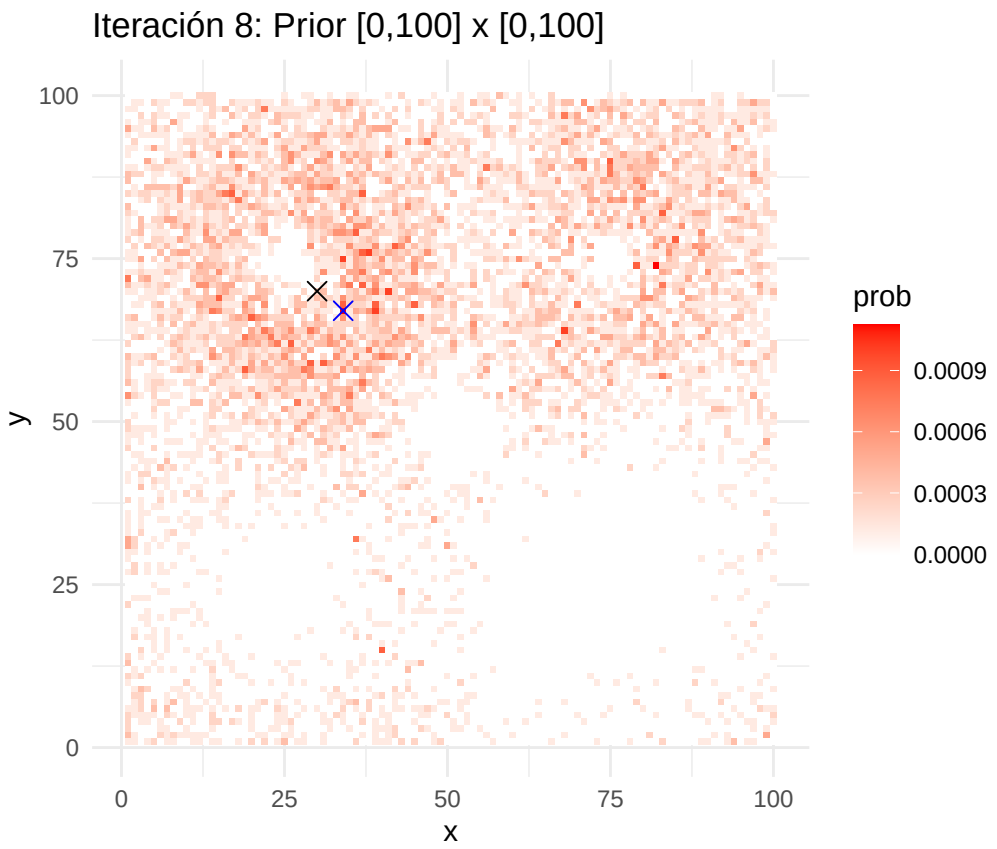



```
##
## Iteración: 6
## Coordenadas de observación: 61 75
## Intensidad observada = -0.0339
## Celda más probable = 82 74
## Probabilidad posterior  0.0022
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
```

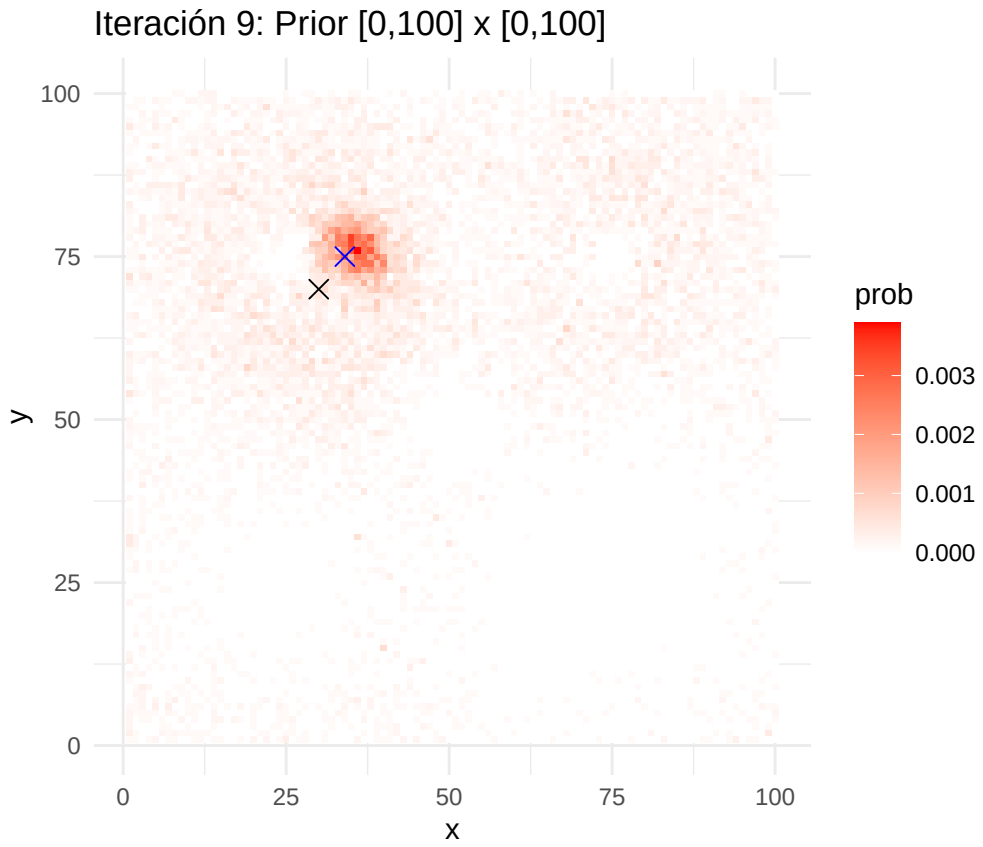
```
## Total execution time: 0.3 seconds.
```



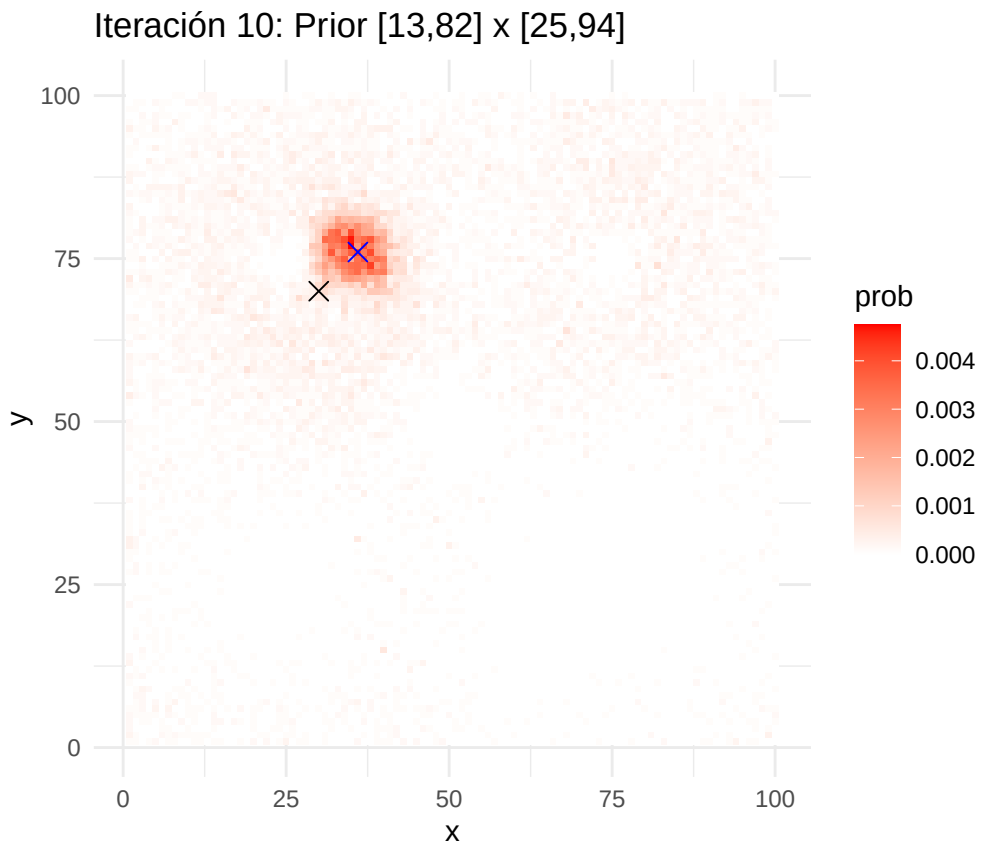

```
##
## Iteración: 7
## Coordenadas de observación: 82 74
## Intensidad observada = 0.021
## Celda más probable = 82 74
## Probabilidad posterior  0.0015
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



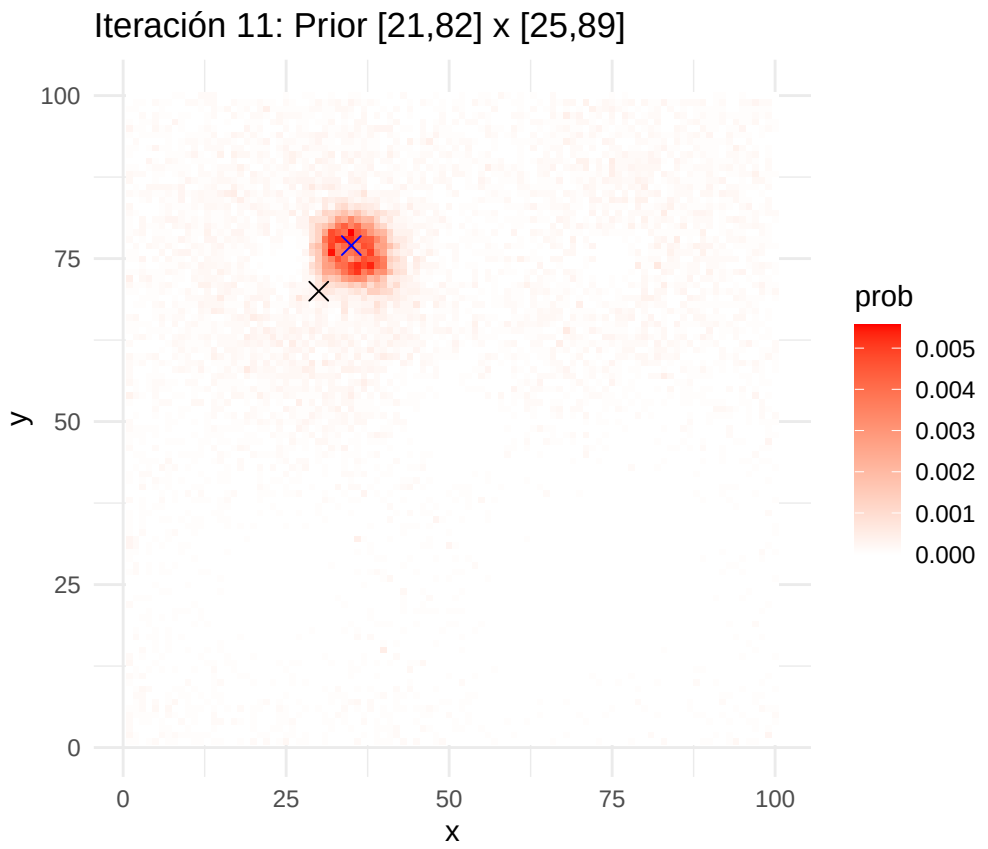

```
##
## Iteración: 8
## Coordenadas de observación: 34 67
## Intensidad observada = 0.3337
## Celda más probable = 34 67
## Probabilidad posterior  0.0011
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



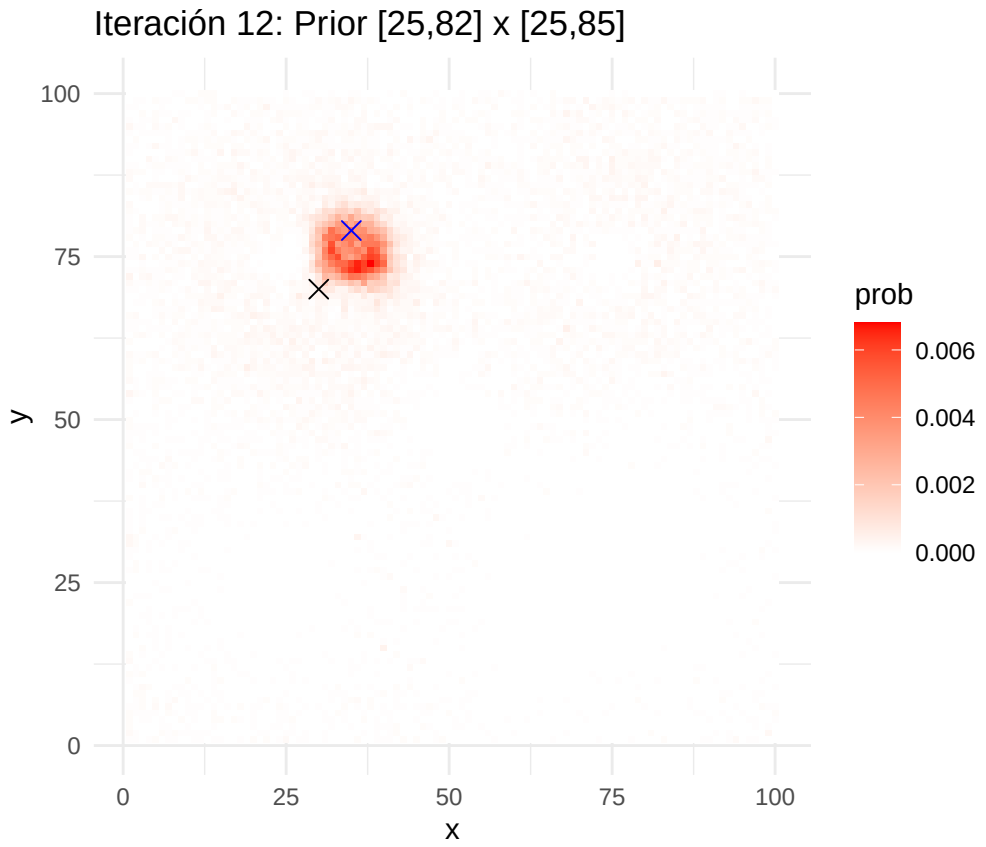

```
##
## Iteración: 9
## Coordenadas de observación: 34 75
## Intensidad observada = 0.8125
## Celda más probable = 36 76
## Probabilidad posterior  0.0039
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



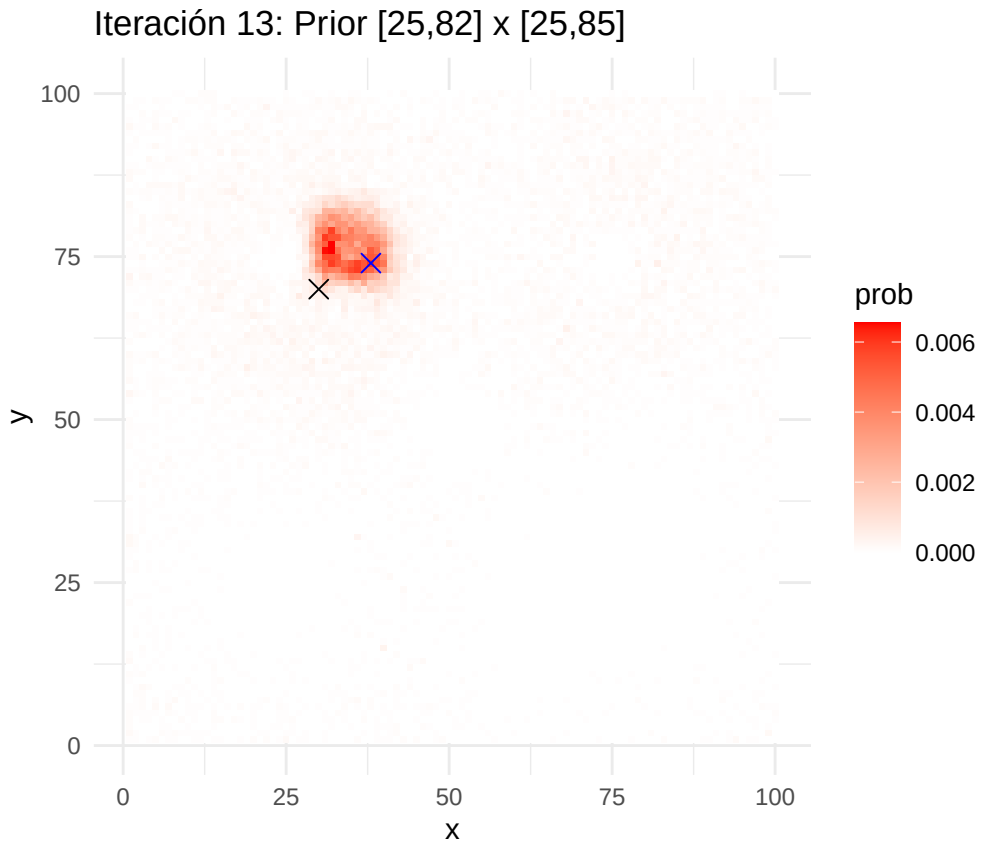

```
##
## Iteración: 10
## Coordenadas de observación: 36 76
## Intensidad observada = 0.643
## Celda más probable = 35 77
## Probabilidad posterior  0.0048
## Prior actual: x  [ 13 , 82 ], y  [ 25 , 94 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



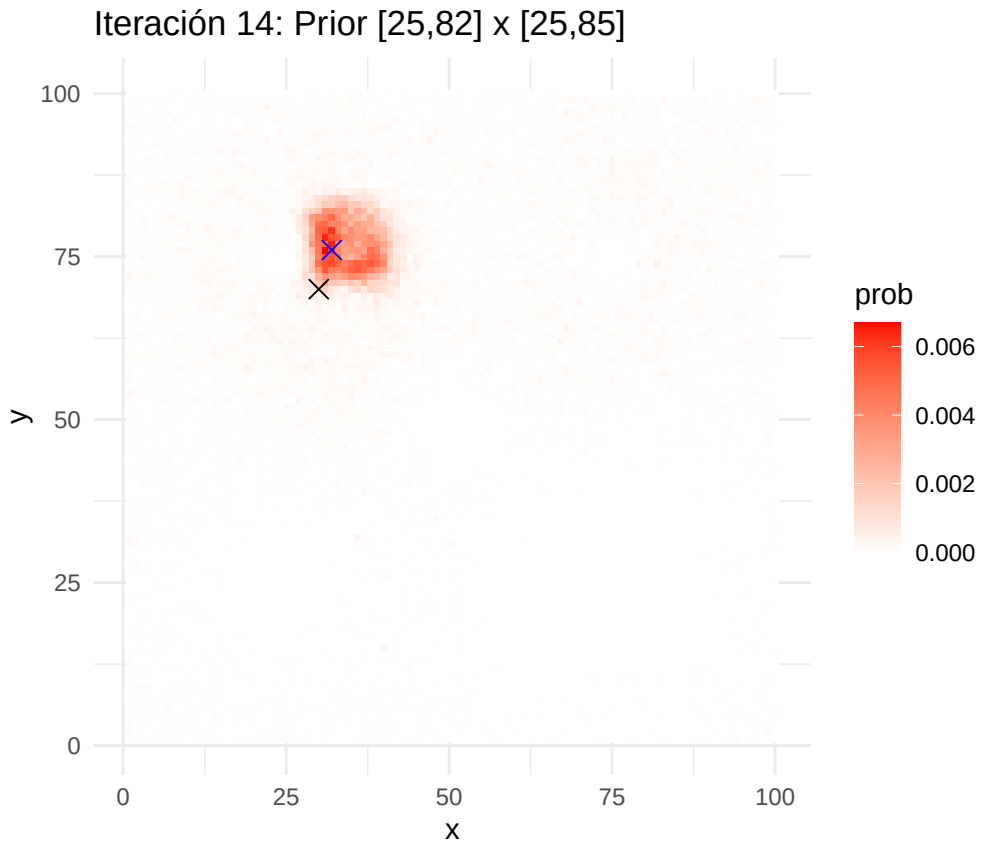

```
##
## Iteración: 11
## Coordenadas de observación: 35 77
## Intensidad observada = 0.74
## Celda más probable = 35 79
## Probabilidad posterior  0.0056
## Prior actual: x  [ 21 , 82 ], y  [ 25 , 89 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



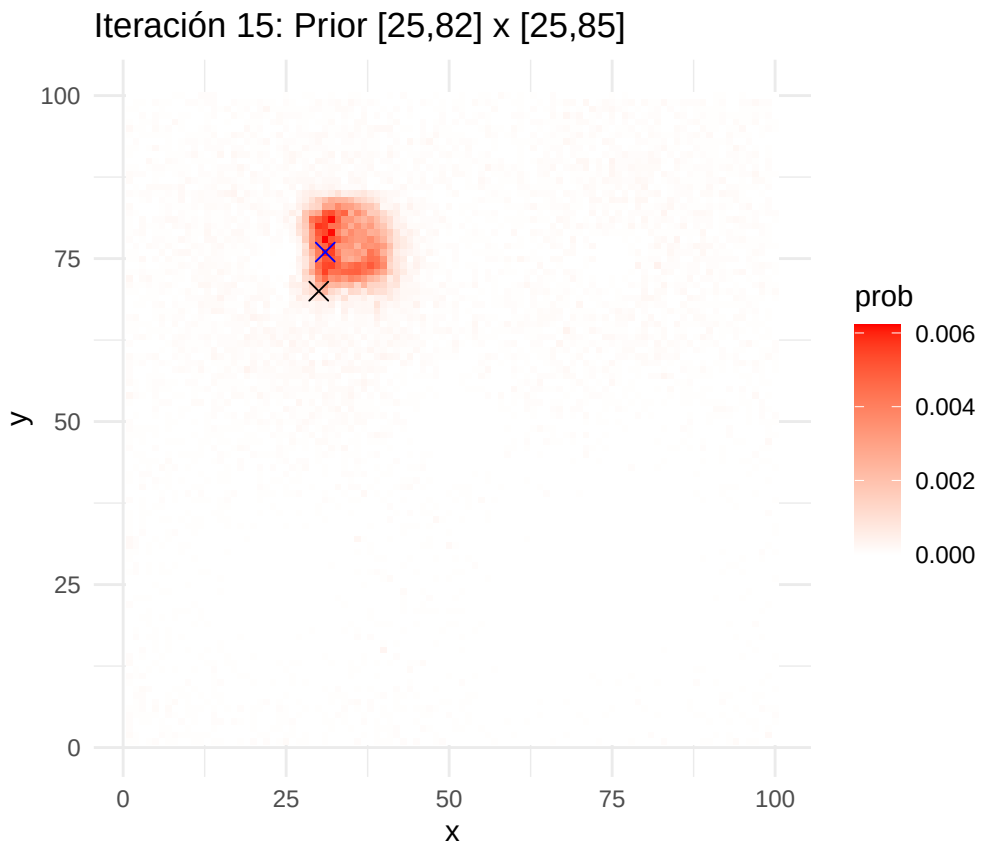

```
##
## Iteración: 12
## Coordenadas de observación: 35 79
## Intensidad observada = 0.383
## Celda más probable = 38 74
## Probabilidad posterior  0.0068
## Prior actual: x  [ 25 , 82 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```



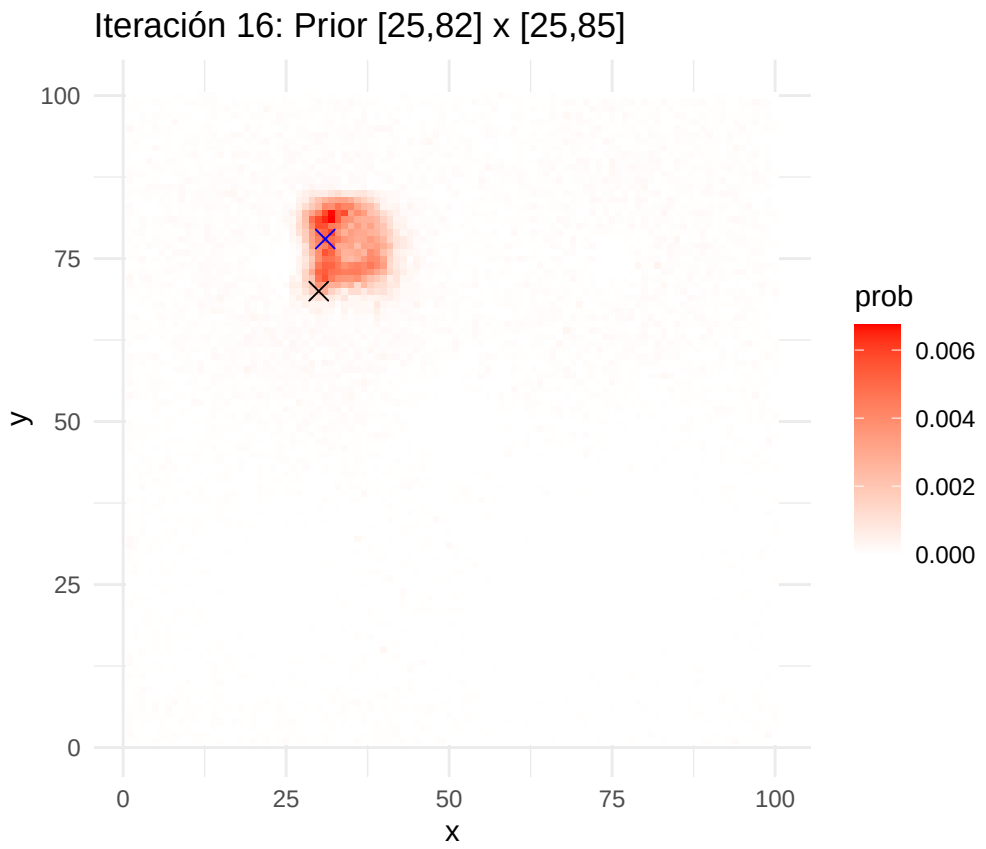

```
##
## Iteración: 13
## Coordenadas de observación: 38 74
## Intensidad observada = 0.0462
## Celda más probable = 32 76
## Probabilidad posterior  0.0066
## Prior actual: x  [ 25 , 82 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.2 seconds.
```



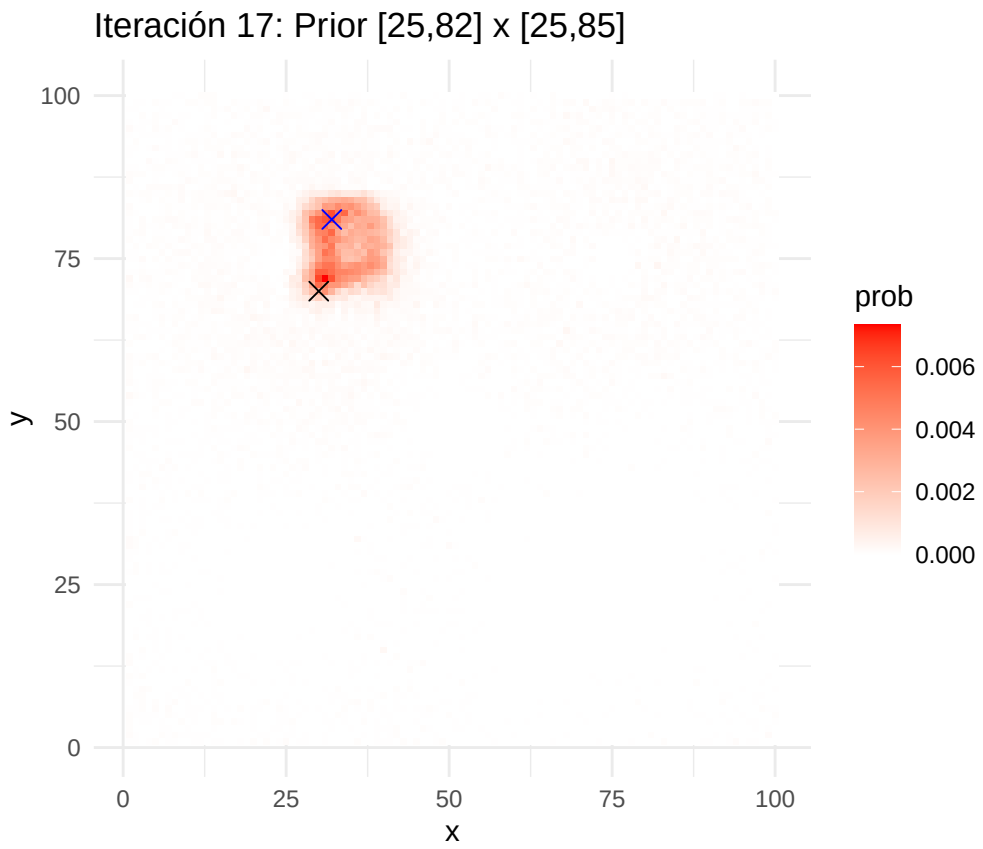

```
##
## Iteración: 14
## Coordenadas de observación: 32 76
## Intensidad observada = 0.5177
## Celda más probable = 31 76
## Probabilidad posterior  0.0067
## Prior actual: x  [ 25 , 82 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 15
## Coordenadas de observación: 31 76
## Intensidad observada = 0.4617
## Celda más probable = 31 78
## Probabilidad posterior  0.0062
## Prior actual: x  [ 25 , 82 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 16
## Coordenadas de observación: 31 78
## Intensidad observada = 0.5089
## Celda más probable = 32 81
## Probabilidad posterior  0.0068
## Prior actual: x  [ 25 , 82 ], y  [ 25 , 85 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.1 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.1 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.1 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 17
## Coordenadas de observación: 32 81
## Intensidad observada = 0.3226
## Celda más probable = 31 72
## Probabilidad posterior  0.0073
## Prior actual: x  [ 25 , 82 ], y  [ 25 , 85 ]
## El modelo ha localizado el objeto en la iteración 17
```

Aquí el modelo suele encontar con relativa facilidad el objeto, gracias a la precisión aumentada a corta distancia.

3. Grupo de simulaciones

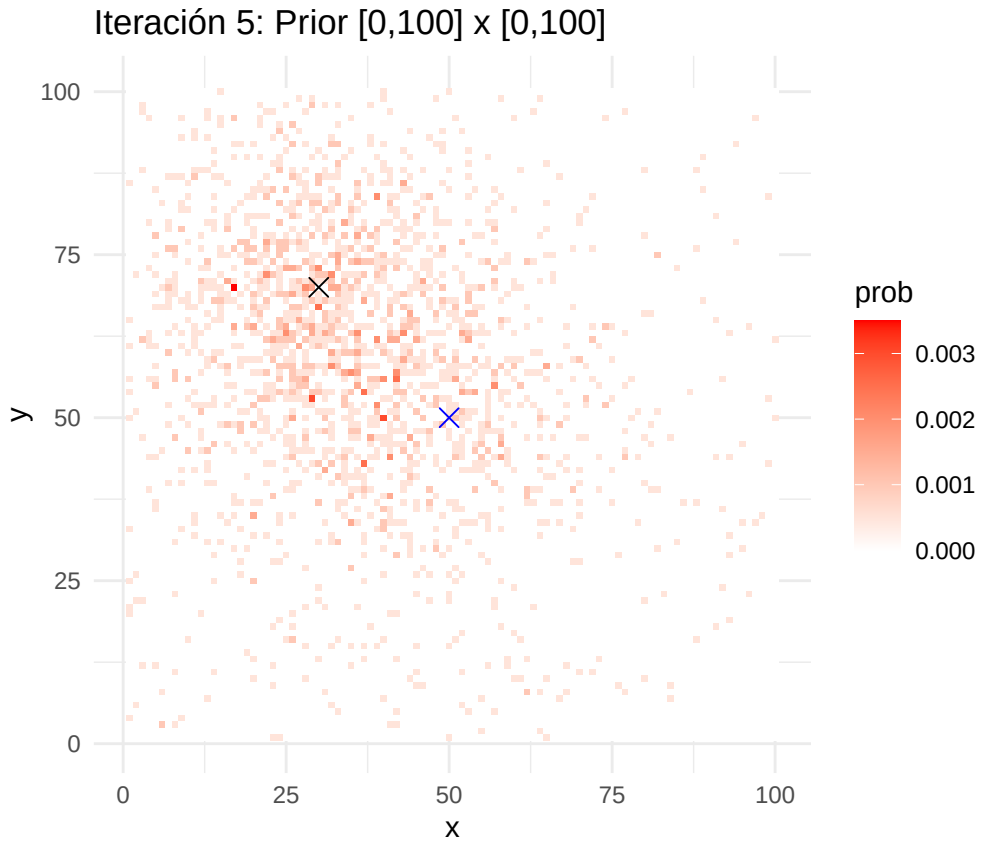
Para estas últimas simulaciones vamos a variar los valores de la amplitud de la intensidad.

3.2

Vamos a cambiar la A a 0.5

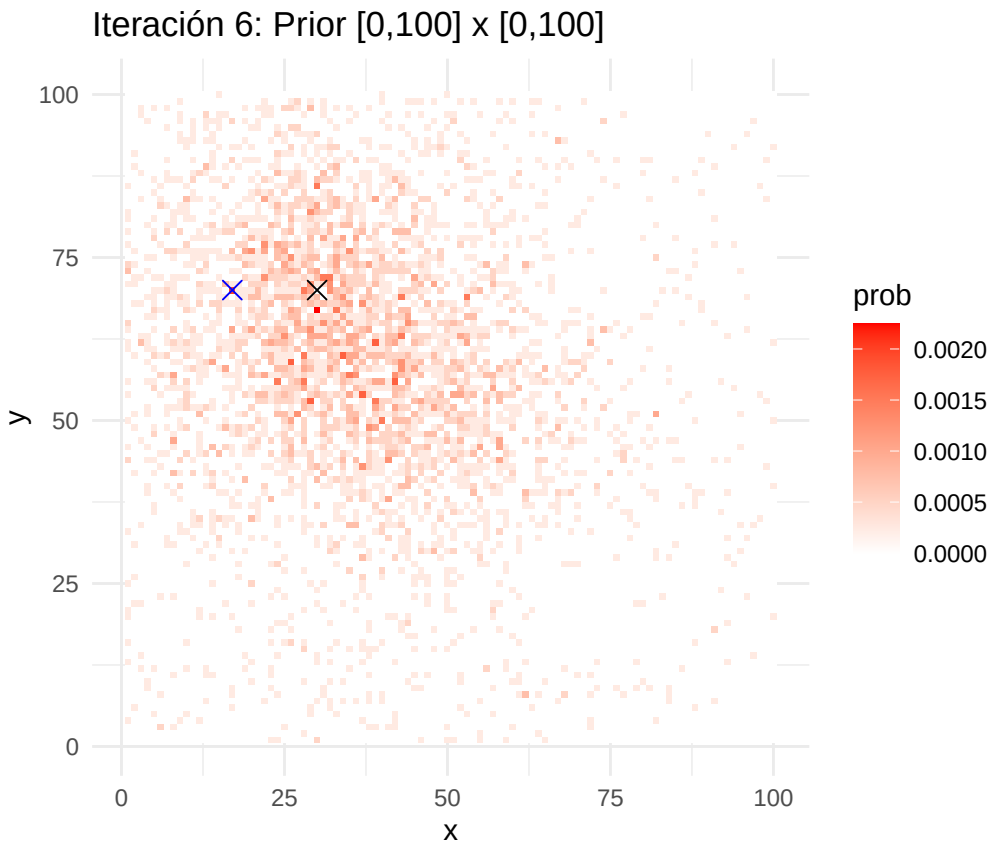
```
sim7 <- localiza_objeto(true_x = 30, true_y = 70,
                            A = 0.5, lambda = 20, sigma = 0.2,
                            n_obs = 25, mostrar_plot = TRUE,
                            guardar_datos = TRUE)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 5
## Coordenadas de observación: 50 50
## Intensidad observada = 0.426
## Celda más probable = 17 70
## Probabilidad posterior  0.0035
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
```

```
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 6
## Coordenadas de observación: 17 70
## Intensidad observada = 0.1866
## Celda más probable = 30 67
## Probabilidad posterior  0.0022
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## El modelo ha localizado el objeto en la iteración 6
```

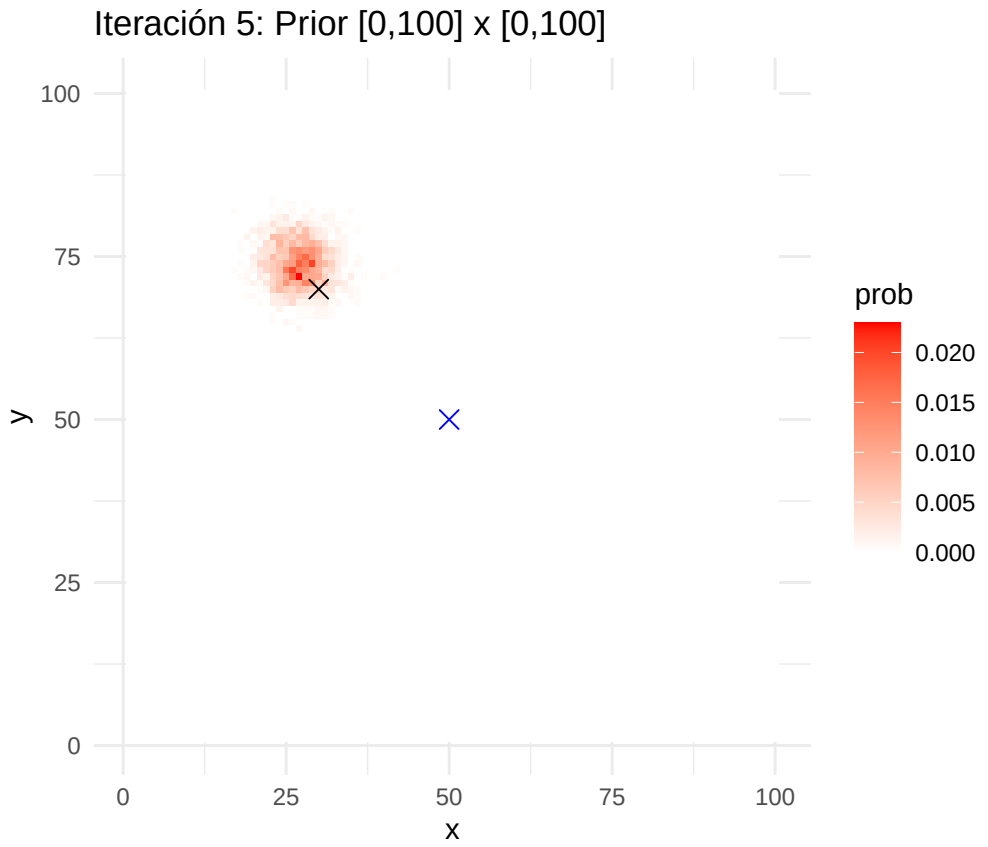
Este valor debería provocar que al modelo le costase mucho trabajo encontrar el objeto. Sin embargo, veremos como si va acotando la prior.

3.3

En esta última simulación cambiaremos el parámetro A = 2.

```
sim8 <- localiza_objeto(true_x = 30, true_y = 70,
                        A = 2, lambda = 20, sigma = 0.2,
                        n_obs = 25, mostrar_plot = TRUE,
                        guardar_datos = TRUE)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 finished in 0.0 seconds.
## Chain 2 finished in 0.0 seconds.
## Chain 3 finished in 0.0 seconds.
## Chain 4 finished in 0.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.0 seconds.
## Total execution time: 0.3 seconds.
```




```
##
## Iteración: 5
## Coordenadas de observación: 50 50
## Intensidad observada = 0.7994
## Celda más probable = 27 72
## Probabilidad posterior  0.023
## Prior actual: x  [ 0 , 100 ], y  [ 0 , 100 ]
## El modelo ha localizado el objeto en la iteración 5
```

Con este valor el modelo debería encontar de forma directa el valor.

7. Conclusiones

Me ha parecido interesante llevar a cabo este trabajo. No obstante, con la experiencia que he adquirido al hacerlo, cambiaría ciertas cosas. Para empezar, no ceñiría el modelo a RStan; intentaría desarrollarlo fuera de su marco. RStan tiene un molde que debes seguir a rajatabla. En mi opinión, este molde me ha impedido reutilizar las distribuciones a posteriori que iba generando el modelo.

Por otro lado, me habría gustado comparar este modelo con otros algoritmos que están explícitamente desarrollados para el ámbito de la búsqueda. Otros escenarios que me habría gustado simular son: si el objeto, en vez de estar estático, se moviese; si el plano tuviera restricciones y, por lo tanto, no se pudiera usar la distancia euclídea.

Asimismo, otros escenarios interesantes podrían ser aquellos en los que, además de la señal, existiesen otras pistas que pudieran utilizarse para guiar la búsqueda y ver cómo interactúan entre sí las nuevas claves.