Bay(es) Window

Michael Betancourt

April 2024

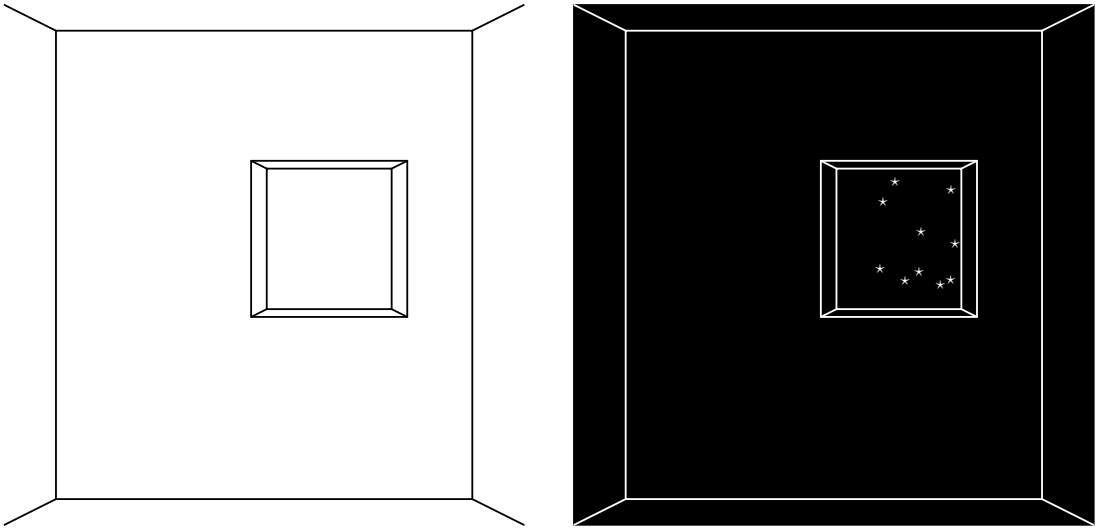
Table of contents

Exercise 22.10 of MacKay (2003) introduces the problem of inferring the dimensions of a window in the dark, using only the positions of stars seen through it. In this short case study I present a comprehensive Bayesian analysis of this problem.

1 The Room

Consider a small room with square walls, only one of which features a window that allows us to see to the outside. During the day this view allows us to fully distinguish the geometry of the

window, including not only its width and height but also its position on the wall (Figure 1a). On a moonless night, however, the only light we see is from a scattering of stars. The edges of the window are enveloped by the darkness and become indistinguishable from the unlit wall (Figure 1b).



(a) (b)

Figure 1: External light passing through a window allow us to differentiate the window from the surrounding wall. (a) During the day the bright sunlight allows us to fully characterize the window geometry but (b) during a moonless night we are limited to starlight which only partially informs the window geometry.

While the starlight does not allow us to completely determine the geometry of the window it does provide some information. Our goal is to infer the window geometries that are consistent with the stars visible on a given night.

To that end let's set up a coordinate system that describes the geometry of the room, including both the wall and the window (Figure 2). Let's say that we do know that the height and width of the wall are both $2\,L = 6$ meters. We can then denote the left, right, bottom, and top edges of the window with the variables l, r, b, and t respectively.

2 The Observational Model

The data generating process here is relatively straightforward. Given a collection of stars we observe the positions of the stars that fall within the breadth of the window. More formally we have a selection process with a latent distribution of star positions and a deterministic selection function that rejects all stars that are obscured by the wall.

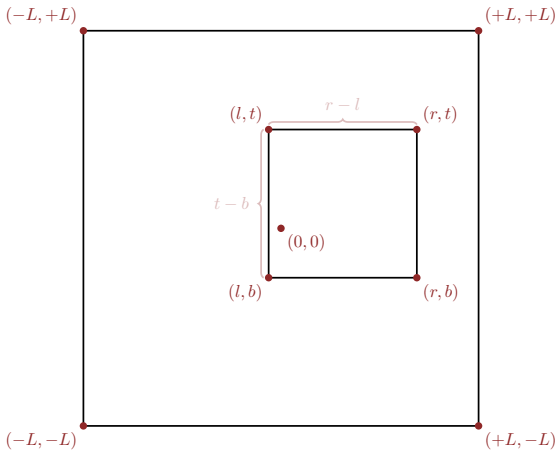


Figure 2: A natural coordinate system for this problem is centered on the center of the wall. The left l, right r, bottom b, and top t edges of the window can then be described by distances from this center.

Firstly let's assume that the horizontal and vertical positions of each star that we could possibly see are uniformly distributed across the total breadth of the wall,

$$p(x, y) = \text{uniform}(x \mid -L, +L)\,\text{uniform}(y \mid -L, +L).$$

The wall then defines a deterministic selection function for a star being observed,

$$p(z = 1 \mid x, y, l, r, b, t) = I_{(l,r)}(x)\, I_{(b,t)}(y).$$

Finally the observed star positions are modeled as

$$\begin{aligned}
&p(x, y \mid z = 1, l, r, b, t) \\
&\quad = \frac{p(x, y)\, p(z = 1 \mid x, y)}{\int \mathrm{d}x\, \mathrm{d}y\, p(x, y)\, p(z = 1 \mid x, y)} \\
&\quad = \frac{\text{uniform}(x \mid -L, +L)\,\text{uniform}(y \mid -L, +L)\, I_{(l,r)}(x)\, I_{(b,t)}(y)}{\int \mathrm{d}x\, \mathrm{d}y\, \text{uniform}(x \mid -L, +L)\,\text{uniform}(y \mid -L, +L)\, I_{(l,r)}(x)\, I_{(b,t)}(y)} \\
&\quad = \frac{\text{uniform}(x \mid -L, +L)\, I_{(l,r)}(x)}{\int \mathrm{d}x\, \text{uniform}(x \mid -L, +L)\, I_{(l,r)}(x)} \frac{\text{uniform}(y \mid -L, +L)\, I_{(b,t)}(y)}{\int \mathrm{d}y\, \text{uniform}(y \mid -L, +L)\, I_{(b,t)}(y)}
\end{aligned}$$

To simplify this further we'll need to take advantage of two facts. First the uniform probability density function is defined by an indicator function of its own,

$$\text{uniform}(z \mid a, b) = \frac{I_{(a,b)}(z)}{|b - a|}.$$

Second the product of two interval indicator functions is given by the indicator function of the intersection of the two intervals,

$$I_{(a,b)}(z)\, I_{(c,d)}(z) = I_{(a,b)\cap(c,d)}(z).$$

Putting these two properties together gives

$$\begin{aligned}
\text{uniform}(z \mid a, b)\, I_{(c,d)}(z) &= \frac{I_{(a,b)}(z)}{|b-a|}\, I_{(c,d)}(z) \\
&= \frac{I_{(a,b)}(z)\, I_{(c,d)}(z)}{|b-a|} \\
&= \frac{I_{(a,b)\cap(c,d)}(z)}{|b-a|}.
\end{aligned}$$

Because $(l, r) \subset (-L, +L)$ we have

$$(-L, +L) \cap (l, r) = (l, r)$$

and consequently

$$\begin{aligned}
\text{uniform}(x \mid -L, +L)\, I_{(l,r)}(x) &= \frac{I_{(-L,+L)\cap(l,r)}(x)}{2L} \\
&= \frac{I_{(l,r)}(x)}{2L}.
\end{aligned}$$

Similarly

$$\text{uniform}(x \mid -L, +L)\, I_{(b,t)}(y) = \frac{I_{(b,t)}(y)}{2L}.$$

Substituting these two results into the observational model for a single star finally gives

$$\begin{aligned}
&p(x, y \mid z = 1, l, r, b, t) \\
&\quad = \frac{\text{uniform}(x \mid -L, +L)\, I_{(l,r)}(x)}{\int \mathrm{d}x\, \text{uniform}(x \mid -L, +L)\, I_{(l,r)}(x)} \frac{\text{uniform}(y \mid -L, +L)\, I_{(b,t)}(y)}{\int \mathrm{d}y\, \text{uniform}(y \mid -L, +L)\, I_{(b,t)}(y)} \\
&\quad = \frac{I_{(l,r)}(x)/(2L)}{\int \mathrm{d}x\, I_{(l,r)}(x)/(2L)} \frac{I_{(b,t)}(y)/(2L)}{\int \mathrm{d}y\, I_{(b,t)}(y)/(2L)} \\
&\quad = \frac{I_{(l,r)}(x)}{\int \mathrm{d}x\, I_{(l,r)}(x)} \frac{I_{(b,t)}(y)}{\int \mathrm{d}y\, I_{(b,t)}(y)} \\
&\quad = \frac{I_{(l,r)}(x)}{|r-l|} \frac{I_{(b,t)}(y)}{|t-b|}.
\end{aligned}$$

When we observe N total stars with *independent* positions (x_n, y_n) the total observational model becomes

$$\begin{aligned} p(x_1, y_1, \ldots, x_N, y_N \mid z = 1, l, r, b, t) &= \prod_{n=1}^{N} \frac{I_{(l,r)}(x_n)}{|r - l|} \frac{I_{(b,t)}(y_n)}{|t - b|} \\ &= \frac{\prod_{n=1}^{N} I_{(l,r)}(x_n) \, I_{(b,t)}(y_n)}{|r - l|^N \, |t - b|^N}. \end{aligned}$$

3 The Likelihood Function

In theory the likelihood function for the window geometry is given by evaluating the observational probability density function on a set observed positions,

$$\begin{aligned} \mathcal{L}(l, r, b, t) &\propto p(\tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_N, \tilde{y}_N \mid z = 1, l, r, b, t) \\ &\propto \frac{\prod_{n=1}^{N} I_{(l,r)}(\tilde{x}_n) \, I_{(b,t)}(\tilde{y}_n)}{|r - l|^N \, |t - b|^N}. \end{aligned}$$

This likelihood function, however, appears to be a bit ungainly to use in practice because of the many discontinuities introduced by all of those indicator functions.

Fortunately this issue is largely an illusion of how we have written the likelihood function here; most of these indicator functions are in fact redundant and can be removed entirely. With a little bit of work we can eliminate these redundancies and simplify the likelihood function into a form that is much easier to wield in practice.

The key to this simplification is to manipulate the indicator functions over the varying star positions into indicator functions over the common window position parameters. In general the indicator function $I_{(a,b)}(z)$ gives non-zero outputs only when

$$a < z < b.$$

This sequence of inequalities, however, can also be written as three separate inequalities,

$$\begin{aligned} a &< z \\ z &< b \\ a &< b. \end{aligned}$$

In turn these separate inequalities define three indicator functions for the interval parameters a and b,

$$\begin{aligned} &I_{(-\infty,z)}(a) \\ &I_{(z,+\infty)}(b) \\ &I_{(-\infty,b)}(a). \end{aligned}$$

Consequently we can we can always write the indicator function $I_{(a,b)}(z)$ as a product of these three indicator functions,

$$I_{(a,b)}(z) = I_{(-\infty,z)}(a)\, I_{(z,+\infty)}(b)\, I_{(-\infty,b)}(a).$$

Again the first indicator function ensures that a is less than z, the second indicator function ensures that b is larger than z, and the third indicator function ensures that a is less than b which is required for (a, b) to be a well-defined interval.

This decomposition then allow us to write any product of indicator functions evaluations as

$$\begin{aligned}
\prod_{n=1}^{N} I_{(a,b)}(z_n) &= \prod_{n=1}^{N} I_{(-\infty,z_n)}(a)\, I_{(z_n,+\infty)}(b)\, I_{(-\infty,b)}(a) \\
&= \left[\prod_{n=1}^{N} I_{(-\infty,z_n)}(a)\right] \left[\prod_{n=1}^{N} I_{(z_n,+\infty)}(b)\right] \left[\prod_{n=1}^{N} I_{(-\infty,b)}(a)\right] \\
&= \left[\prod_{n=1}^{N} I_{(-\infty,z_n)}(a)\right] \left[\prod_{n=1}^{N} I_{(z_n,+\infty)}(b)\right] \left(I_{(-\infty,b)}(a)\right)^N \\
&= \left[\prod_{n=1}^{N} I_{(-\infty,z_n)}(a)\right] \left[\prod_{n=1}^{N} I_{(z_n,+\infty)}(b)\right] I_{(-\infty,b)}(a).
\end{aligned}$$

Now the overlap of the intervals $(-\infty, z_n)$ is completely determined by the smallest z_n,

$$\bigcap_{n=1}^{N} (-\infty, z_n) = (-\infty, \min(z_n)).$$

Consequently the first product reduces to

$$\begin{aligned}
\prod_{n=1}^{N} I_{(-\infty,z_n)}(a) &= I_{\cap_{n=1}^{N}(-\infty,z_n)}(a) \\
&= I_{(-\infty,\min(z_n))}(a).
\end{aligned}$$

Similarly

$$\bigcap_{n=1}^{N} (z_n, +\infty) = (\max(z_n), +\infty).$$

and

$$\begin{aligned}
\prod_{n=1}^{N} I_{(z_n,+\infty)}(b) &= I_{\cap_{n=1}^{N}(z_n,+\infty)}(b) \\
&= I_{(\max(z_n),+\infty)}(b).
\end{aligned}$$

This allows us to replace all of those indicator functions with just three indicator functions,

$$\begin{aligned}\prod_{n=1}^{N} I_{(a,b)}(z_n) &= \left[\prod_{n=1}^{N} I_{(-\infty,z_n)}(a)\right] \left[\prod_{n=1}^{N} I_{(z_n,+\infty)}(b)\right] I_{(-\infty,b)}(a) \\ &= I_{(-\infty,\min(z_n))}(a)\, I_{(\max(z_n),+\infty)}(b)\, I_{(-\infty,b)}(a).\end{aligned}$$

For real-valued z_n ties will occur with probability zero and we will have

$$\min(z_n) < \max(z_n)$$

with probability one so that the last indicator function is also redundant. Consequently we can write

$$\prod_{n=1}^{N} I_{(a,b)}(z_n) = I_{(-\infty,\min(z_n))}(a)\, I_{(\max(z_n),+\infty)}(b).$$

Applying this result twice our likelihood function takes on the much simpler form

$$\begin{aligned}\mathcal{L}(l,r,b,t) &\propto \frac{\prod_{n=1}^{N} I_{(l,r)}(\tilde{x}_n)\, I_{(b,t)}(\tilde{y}_n)}{|r-l|^N\, |t-b|^N} \\ &\propto \frac{\prod_{n=1}^{N} I_{(l,r)}(\tilde{x}_n) \prod_{n=1}^{N} I_{(b,t)}(\tilde{y}_n)}{|r-l|^N\, |t-b|^N} \\ &\propto \frac{I_{(-\infty,\min(\tilde{x}_n))}(l)\, I_{(\max(\tilde{x}_n),+\infty)}(r)}{|r-l|^N} \\ &\quad \cdot \frac{I_{(-\infty,\min(\tilde{y}_n))}(b)\, I_{(\max(\tilde{y}_n),+\infty)}(t)}{|t-b|^N}.\end{aligned}$$

The first indicator function keeps the left window boundary below all observed x positions while the second indicator function keeps the right window boundary above all observed y positions. Similarly the last two indicator functions ensure that the bottom window boundary is below, and the top window boundary is above, all observed y positions. As the number of observations N grows the denominator pulls the likelihood function closer and closer to the minimum and maximum x and y positions.

Because the likelihood function can always be written in terms of $\min(\tilde{x}_n)$, $\max(\tilde{x}_n)$, $\min(\tilde{y}_n)$, and $\max(\tilde{y}_n)$ regardless of what the individual positions are, these extreme values are *sufficient statistics* for this particular observational model.

4 Bayesian Inference

This simplified likelihood function is straightforward to implement in `Stan`, allowing us to readily quantify posterior inferences for the window geometry given observed star positions.

4.1 Setup

As always we begin by setting up our local `R` environment.

```
par(family="serif", las=1, bty="l",
    cex.axis=1, cex.lab=1, cex.main=1,
    xaxs="i", yaxs="i", mar = c(5, 5, 3, 5))

c_light <- c("#DCBCBC")
c_light_highlight <- c("#C79999")
c_mid <- c("#B97C7C")
c_mid_highlight <- c("#A25050")
c_dark <- c("#8F2727")
c_dark_highlight <- c("#7C0000")

c_light_teal <- c("#6B8E8E")
c_mid_teal <- c("#487575")
c_dark_teal <- c("#1D4F4F")

library(colormap)
disc_colors <- c("#FFFFFF", "#DCBCBC", "#C79999", "#B97C7C",
                 "#A25050", "#8F2727", "#7C0000")
```

In particular we'll need to load `RStan` and my recommended Hamiltonian Monte Carlo analysis suite.

```
library(rstan)
rstan_options(auto_write = TRUE)            # Cache compiled Stan programs
options(mc.cores = parallel::detectCores()) # Parallelize chains
parallel:::setDefaultClusterOptions(setup_strategy = "sequential")

util <- new.env()
source('mcmc_analysis_tools_rstan.R', local=util)
```

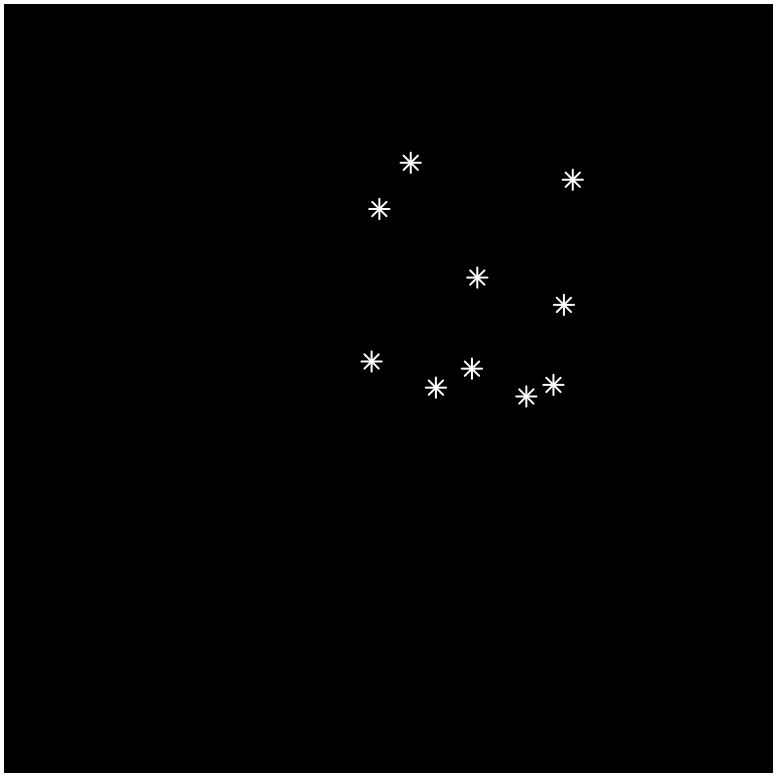
4.2 Explore Data

Let's start by reading in the observed star positions.

```
data <- read_rdump("data/stars.data.R")
```

The observed stars positions imply that the window is situated towards the top right corner of the wall but the precise edges of the windows are hard to resolve.

```
L <- 3
par(mfrow=c(1, 1), mar=c(0, 0, 0, 0))

plot(0, type='n',
     xlim=c(-L, +L), ylim=c(-L, +L),
     axes=FALSE, ann=FALSE)
rect(-L, -L, +L, L, col = "black")
points(data$xs, data$ys, col="white", pch=8)
```



4.3 Quantify Posterior Distribution

In addition to the requirement of a positive window size,

$$l < r$$
$$b < t,$$

the window geometry is constrained by the wall geometry,

$$\begin{aligned} -L < l < +L \\ -L < r < +L \\ -L < b < +L \\ -L < t < +L. \end{aligned}$$

If we don't have any additional information about the position of the window then our domain expertise is captured by the uniform prior density function subject to these constraints,

$$\begin{aligned} p(l, r, b, t) &= p(l, r)\, p(b, t) \\ &= \text{uniform}(l \mid -L, +L)\, \text{uniform}(r \mid l, +L) \\ &\quad \cdot \text{uniform}(b \mid -L, +L)\, \text{uniform}(t \mid b, +L) \end{aligned}$$

With this prior model the posterior density function completely decouples into independent posterior density functions for each of the two directions,

$$\begin{aligned} & p(l, r, b, t \mid \tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_N, \tilde{y}_N) \\ & \quad \propto p(\tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_N, \tilde{y}_N) \mid l, r, b, t)\, p(l, r, b, t) \\ & \quad \propto \frac{I_{(-\infty, \min(\tilde{x}_n))}(l)\, I_{(\max(\tilde{x}_n), +\infty)}(r)}{|r - l|^N} \\ & \qquad \cdot \frac{I_{(-\infty, \min(\tilde{y}_n))}(b)\, I_{(\max(\tilde{y}_n), +\infty)}(t)}{|t - b|^N} \\ & \qquad \cdot \text{uniform}(l \mid -L, +L)\, \text{uniform}(r \mid l, +L) \\ & \qquad \cdot \text{uniform}(b \mid -L, +L)\, \text{uniform}(t \mid b, +L) \\ & \quad \propto \frac{I_{(-\infty, \min(\tilde{x}_n))}(l)\, I_{(\max(\tilde{x}_n), +\infty)}(r)}{|r - l|^N} \\ & \qquad \cdot \text{uniform}(l \mid -L, +L)\, \text{uniform}(r \mid l, +L) \\ & \qquad \cdot \frac{I_{(-\infty, \min(\tilde{y}_n))}(b)\, I_{(\max(\tilde{y}_n), +\infty)}(t)}{|t - b|^N} \\ & \qquad \cdot \text{uniform}(b \mid -L, +L)\, \text{uniform}(t \mid b, +L) \\ & \quad \propto p(l, r \mid \tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_N, \tilde{y}_N) \\ & \qquad \cdot p(b, t \mid \tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_N, \tilde{y}_N). \end{aligned}$$

In particular we can immediately pushforward the four-dimensional posterior density function into two-dimensional, marginal posterior density functions for the horizontal and vertical geometries,

$$\begin{aligned} p(l, r \mid \tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_N, \tilde{y}_N) \propto & \frac{I_{(-\infty, \min(\tilde{x}_n))}(l)\, I_{(\max(\tilde{x}_n), +\infty)}(r)}{|r - l|^N} \\ & \cdot \text{uniform}(l \mid -L, +L)\, \text{uniform}(r \mid l, +L) \end{aligned}$$

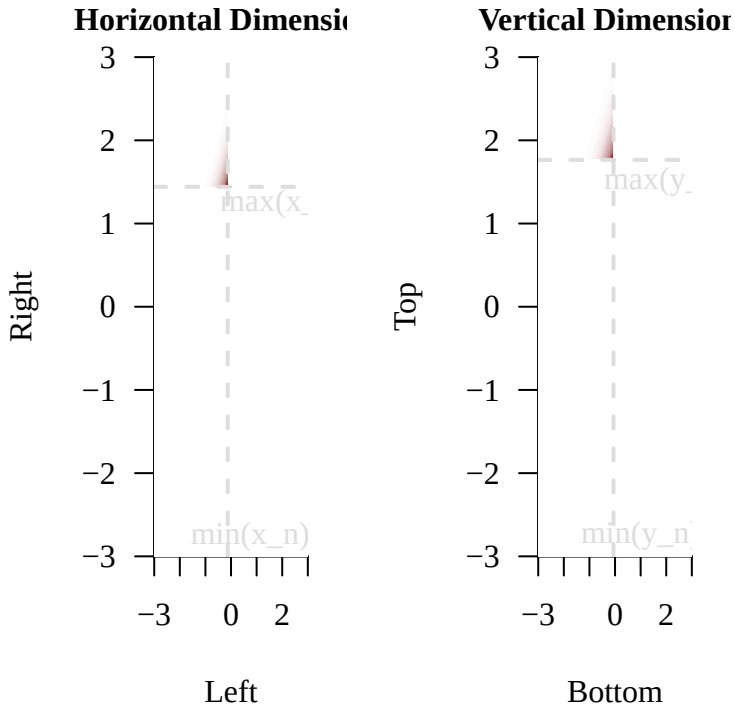
and

$$p(b, t \mid \tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_N, \tilde{y}_N) \propto \frac{I_{(-\infty, \min(\tilde{y}_n))}(b)\, I_{(\max(\tilde{y}_n), +\infty)}(t)}{|t - b|^N} \cdot \text{uniform}(b \mid -L, +L)\, \text{uniform}(t \mid b, +L).$$

Plotting these marginal posterior density functions demonstrates how our inferences push up against the constraints imposed by the minimum and maximum star positions.

```
par(mfrow=c(1, 2), mar=c(5, 5, 2, 1))

K <- 200
left_grid <- seq(-L, +L, 2 * L / (K - 1))
right_grid <- seq(-L, +L, 2 * L / (K - 1))

post_pds <- matrix(0, nrow=K, ncol=K)

for (k in 1:K) {
  for (kk in 1:K) {
    if (left_grid[k] > min(data$xs)) {
      post_pds[k, kk] <- 0
    } else if (right_grid[kk] < max(data$xs)) {
      post_pds[k, kk] <- 0
    } else {
      post_pds[k, kk] <- (right_grid[kk] - left_grid[k])**(-data$N)
    }
  }
}

disc_colors <- c("#FFFFFF", "#DCBCBC", "#C79999", "#B97C7C",
                 "#A25050", "#8F2727", "#7C0000")
cont_colors <- colormap(colormap=disc_colors, nshades=100)

image(left_grid, right_grid, post_pds, col=rev(cont_colors),
      xlim=c(-L, +L), xlab=c("Left"), ylim=c(-L, +L), ylab=c("Right"),
      main="Horizontal Dimensions")
abline(v=min(data$xs), lty=2, lwd=2, col="#DDDDDD")
text(0.75, -2.75, "min(x_n)", col="#DDDDDD")
abline(h=max(data$xs), lty=2, lwd=2, col="#DDDDDD")
text(2,  1.25, "max(x_n)", col="#DDDDDD")

K <- 200
bottom_grid <- seq(-L, +L, 2 * L / (K - 1))
```

```
top_grid <- seq(-L, +L, 2 * L / (K - 1))

post_pds <- matrix(0, nrow=K, ncol=K)

for (k in 1:K) {
  for (kk in 1:K) {
    if (bottom_grid[k] > min(data$ys)) {
      post_pds[k, kk] <- 0
    } else if (top_grid[kk] < max(data$ys)) {
      post_pds[k, kk] <- 0
    } else {
      post_pds[k, kk] <- (top_grid[kk] - bottom_grid[k])**(-data$N)
    }
  }
}

image(bottom_grid, top_grid, post_pds, col=rev(cont_colors),
      xlim=c(-L, +L), xlab=c("Bottom"), ylim=c(-L, +L), ylab=c("Top"),
      main="Vertical Dimensions")
abline(v=min(data$ys), lty=2, lwd=2, col="#DDDDDD")
text(1, -2.75, "min(y_n)", col="#DDDDDD")
abline(h=max(data$ys), lty=2, lwd=2, col="#DDDDDD")
text(2,  1.5, "max(y_n)", col="#DDDDDD")
```



If we want to extract additional information out of our posterior distribution then we'll need to resort to posterior expectation values and the Markov chain Monte Carlo estimators that approximate them.

The summary statistics $\min(\tilde{x}_n)$, $\max(\tilde{x}_n)$, $\min(\tilde{y}_n)$, and $\max(\tilde{y}_n)$ make for natural retrodictive summaries here. Conveniently we can compute the posterior predictive summary statistics directly in the `generated quantities` block.

```
fit <- stan(file="stan_programs/fit_window.stan",
            data=data, seed=8438338,
            warmup=1000, iter=2024, refresh=0)
```

Conveniently our diagnostics show no serious indications of computational infidelity.

```
diagnostics <- util$extract_hmc_diagnostics(fit)
util$check_all_hmc_diagnostics(diagnostics)
```

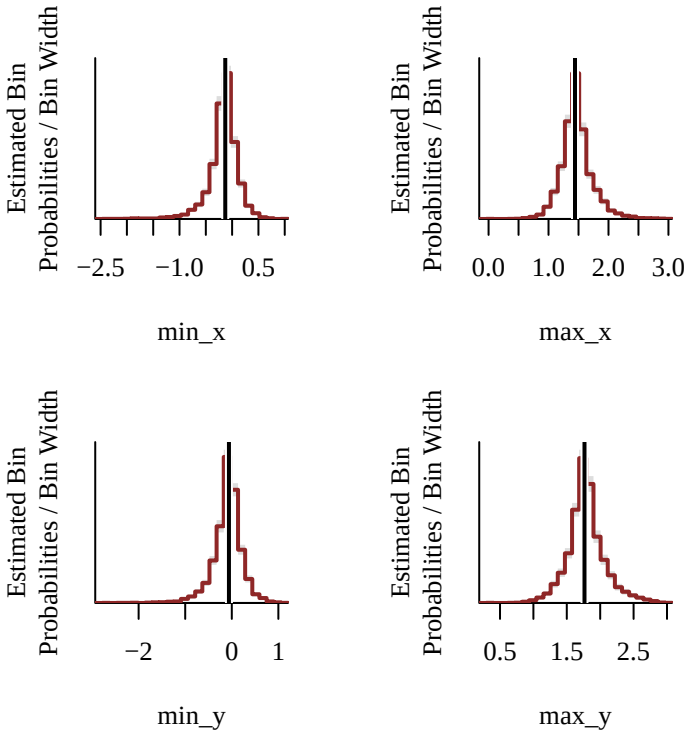
```
  All Hamiltonian Monte Carlo diagnostics are consistent with reliable
Markov chain Monte Carlo.
```

```
samples <- util$extract_expectand_vals(fit)
base_samples <- util$filter_expectands(samples,
                                       c('left', 'right',
                                         'bottom', 'top'))
util$check_all_expectand_diagnostics(base_samples)
```

```
left:
  Chain 2: Left tail hat{xi} (0.294) exceeds 0.25.


Large tail hat{xi}s suggest that the expectand might not be
sufficiently integrable.
```

None of the four summary statistics exhibit any appreciable retrodictive tension, suggesting that our modeling assumptions are adequate.

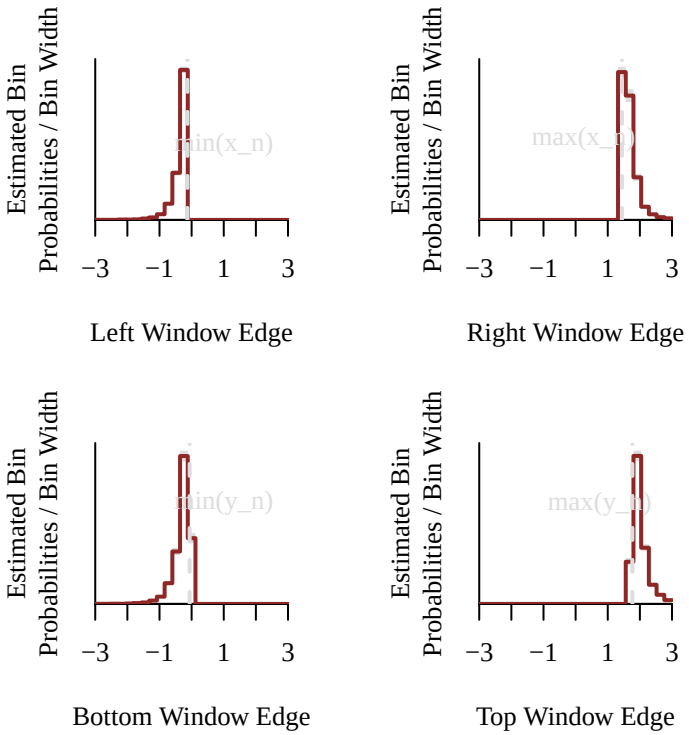
```
par(mfrow=c(2, 2), mar=c(5, 5, 2, 1))

util$plot_expectand_pushforward(samples[['min_x_pred']], 25, 'min_x',
                                baseline=min(data$xs))
util$plot_expectand_pushforward(samples[['max_x_pred']], 25, 'max_x',
                                baseline=max(data$xs))
util$plot_expectand_pushforward(samples[['min_y_pred']], 25, 'min_y',
                                baseline=min(data$ys))
util$plot_expectand_pushforward(samples[['max_y_pred']], 25, 'max_y',
                                baseline=max(data$ys))
```



The Markov chain Monte Carlo fit now allows us to examine marginal posterior inferences for each individual parameter. All four of the marginal posterior distributions concentrate towards the hard edge imposed by the observed summary statistics.

```
par(mfrow=c(2, 2), mar=c(5, 5, 2, 1))

util$plot_expectand_pushforward(samples[['left']], 25,
                                'Left Window Edge', flim=c(-L, +L))
abline(v=min(data$xs), lty=2, lwd=2, col="#DDDDDD")
text(1, 1.4, "min(x_n)", col="#DDDDDD")

util$plot_expectand_pushforward(samples[['right']], 25,
                                'Right Window Edge', flim=c(-L, +L))
abline(v=max(data$xs), lty=2, lwd=2, col="#DDDDDD")
text(0.25, 1, "max(x_n)", col="#DDDDDD")

util$plot_expectand_pushforward(samples[['bottom']], 25,
                                'Bottom Window Edge', flim=c(-L, +L))
abline(v=min(data$ys), lty=2, lwd=2, col="#DDDDDD")
text(1, 1.4, "min(y_n)", col="#DDDDDD")

util$plot_expectand_pushforward(samples[['top']], 25,
```

```
                                    'Top Window Edge', flim=c(-L, +L))
abline(v=max(data$ys), lty=2, lwd=2, col="#DDDDDD")
text(0.75, 1.5, "max(y_n)", col="#DDDDDD")
```



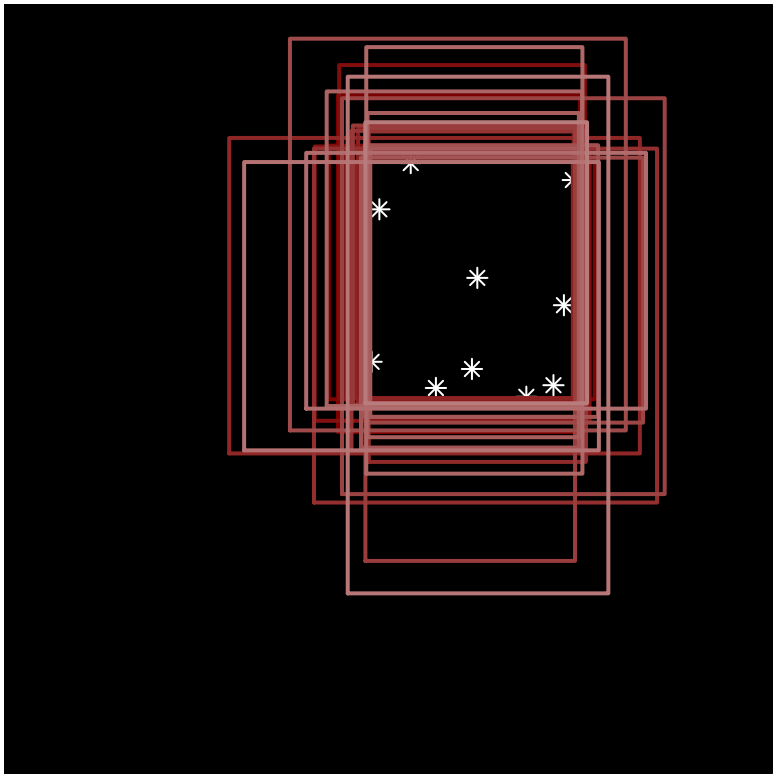
Perhaps the most useful benefit of our Markov chain Monte Carlo fit is that we can visualize the inferred window geometry directly instead of having to recreate it from its individual pieces.

First let's flatten the Markov chains into a single array for each parameter.

```
left_samples <- c(samples[['left']], recursive=TRUE)
right_samples <- c(samples[['right']], recursive=TRUE)
bottom_samples <- c(samples[['bottom']], recursive=TRUE)
top_samples <- c(samples[['top']], recursive=TRUE)
```

Subsetting these flattened arrays selects out a selection of realized window geometries consistent with the observed data. Plotting these over against the observed star positions puts our inferences into a direct geometric context.

```
plot_idxs <- seq(1, 4096, 150)

disc_colors <- c("#B97C7C", "#A25050", "#8F2727", "#7C0000")
line_colors <- colormap(colormap=disc_colors,
                        nshades=length(plot_idxs))

par(mfrow=c(1, 1), mar=c(0, 0, 0, 0))

plot(0, type='n',
     xlim=c(-L, +L), ylim=c(-L, +L),
     axes=FALSE, ann=FALSE)
rect(-L, -L, +L, +L, col = "black")

points(data$xs, data$ys, col="white", pch=8)

for(s in seq_along(plot_idxs)) {
  plot_idx <- plot_idxs[s]
  lines(c(left_samples[plot_idx], right_samples[plot_idx],
          right_samples[plot_idx], left_samples[plot_idx],
          left_samples[plot_idx]),
        c(bottom_samples[plot_idx], bottom_samples[plot_idx],
          top_samples[plot_idx], top_samples[plot_idx],
          bottom_samples[plot_idx]),
        col=line_colors[s], lwd=2)
}
```

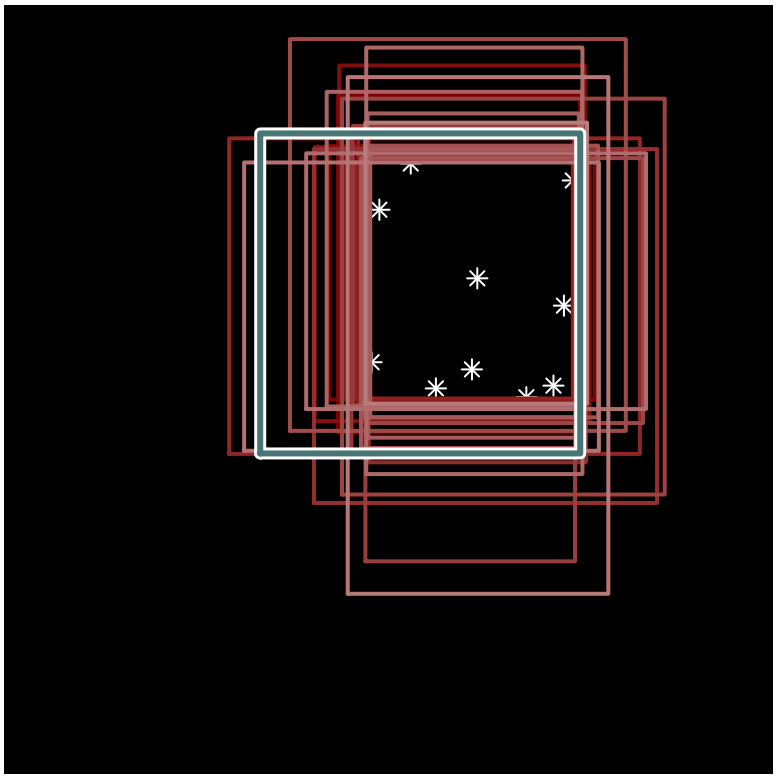


Finally let's compare our posterior inferences to the actual window geometry.

```
truth <- read_rdump("data/truth.data.R")
```

```
par(mfrow=c(1, 1), mar=c(0, 0, 0, 0))

plot(0, type='n',
     xlim=c(-L, +L), ylim=c(-L, +L),
     axes=FALSE, ann=FALSE)
rect(-L, -L, +L, +L, col = "black")

points(data$xs, data$ys, col="white", pch=8)

for(s in seq_along(plot_idxs)) {
  plot_idx <- plot_idxs[s]
  lines(c(left_samples[plot_idx], right_samples[plot_idx],
          right_samples[plot_idx], left_samples[plot_idx],
          left_samples[plot_idx]),
        c(bottom_samples[plot_idx], bottom_samples[plot_idx],
          top_samples[plot_idx], top_samples[plot_idx],
```

```
          bottom_samples[plot_idx]),
        col=line_colors[s], lwd=2)
}

lines(c(truth$left, truth$right, truth$right, truth$left, truth$left),
      c(truth$bottom, truth$bottom, truth$top, truth$top, truth$bottom),
      col="white", lwd=6)
lines(c(truth$left, truth$right, truth$right, truth$left, truth$left),
      c(truth$bottom, truth$bottom, truth$top, truth$top, truth$bottom),
      col=c_mid_teal, lwd=3)
```



Interesting in turns out that we have a bit of a void in the observed stars, with the minimum x position relatively far away from the true left window edge. Our posterior uncertainties, however, are able to account for this fluctuation and ensure that we aren't lead astray.

5 Conclusion

Inferring the dimensions of a window from the stars we see through it is an excellent demonstration of narratively generative modeling in action. Our understanding of the data generating

process directly informs an observational model and sets the context for constructing an initial prior model. Moreover the structure of that observational model immediately motivates informative summary statistics on which we can build powerful posterior retrodictive checks.

As a literal textbook exercise this problem is nice enough to admit a substantial amount of analytic simplification. That said many of most interpretable insights are still obstructed by intractable calculations and tools like Markov chain Monte Carlo are invaluable to the implementation of a rich analysis.

Acknowledgements

A very special thanks to everyone supporting me on Patreon: Adam Fleischhacker, Adriano Yoshino, Alessandro Varacca, Alexander Noll, Alexander Petrov, Alexander Rosteck, Andrea Serafino, Andrew Mascioli, Andrew Rouillard, Andrew Vigotsky, Ara Winter, Austin Rochford, Avraham Adler, Ben Matthews, Ben Swallow, Benoit Essiambre, Bertrand Wilden, Bradley Kolb, Brandon Liu, Brendan Galdo, Brynjolfur Gauti Jónsson, Cameron Smith, Canaan Breiss, Cat Shark, Charles Naylor, Charles Shaw, Chase Dwelle, Chris Jones, Christopher Mehrvarzi, Colin Carroll, Colin McAuliffe, Damien Mannion, dan mackinlay, Dan W Joyce, Dan Waxman, Dan Weitzenfeld, Daniel Edward Marthaler, Darshan Pandit, Darthmaluus , Dav Pe, David Galley, David Wurtz, Denis Vlašiček, Doug Rivers, Dr. Jobo, Dr. Omri Har Shemesh, Dylan Maher, Ed Cashin, Edgar Merkle, Eric LaMotte, Ero Carrera, Eugene O'Friel, Felipe González, Fergus Chadwick, Finn Lindgren, Florian Wellmann, Geoff Rollins, Guido Biele, Håkan Johansson, Hamed Bastan-Hagh, Hauke Burde, Hector Munoz, Henri Wallen, hs, Hugo Botha, Ian, Ian Costley, idontgetoutmuch, Ignacio Vera, Ilaria Prosdocimi, Isaac Vock, J, J Michael Burgess, jacob pine, Jair Andrade, James C, James Hodgson, James Wade, Janek Berger, Jason Martin, Jason Pekos, Jason Wong, jd, Jeff Burnett, Jeff Dotson, Jeff Helzner, Jeffrey Erlich, Jesse Wolfhagen, Jessica Graves, Joe Wagner, John Flournoy, Jonathan H. Morgan, Jonathon Vallejo, Joran Jongerling, JU, Justin Bois, Kádár András, Karim Naguib, Karim Osman, Kejia Shi, Kristian Gårdhus Wichmann, Lars Barquist, lizzie , LOU ODETTE, Luís F, Marcel Lüthi, Marek Kwiatkowski, Mark Donoghoe, Markus P., Márton Vaitkus, Matt Moores, Matthew, Matthew Kay, Matthieu LEROY, Mattia Arsendi, Maurits van der Meer, Michael Colaresi, Michael DeWitt, Michael Dillon, Michael Lerner, Mick Cooney, N Sanders, N.S. , Name, Nathaniel Burbank, Nic Fishman, Nicholas Clark, Nicholas Cowie, Nick S, Octavio Medina, Ole Rogeberg, Oliver Crook, Olivier Ma, Patrick Kelley, Patrick Boehnke, Pau Pereira Batlle, Peter Johnson, Pieter van den Berg , ptr, Ramiro Barrantes Reynolds, Raúl Peralta Lozada, Ravin Kumar, Rémi , Rex Ha, Riccardo Fusaroli, Richard Nerland, Robert Frost, Robert Goldman, Robert kohn, Robin Taylor, Ryan Grossman, Ryan Kelly, S Hong, Sean Wilson, Sergiy Protsiv, Seth Axen, shira, Simon Duane, Simon Lilburn, sssz, Stan_user, Stephen Lienhard, Stew Watts, Stone Chen, Susan Holmes, Svilup, Tao Ye, Tate Tunstall, Tatsuo Okubo, Teresa Ortiz, Theodore Dasher, Thomas Kealy, Thomas Siegert, Thomas Vladeck, Tom McEwen, Tomáš Frýda, Tony Wuersch, Virginia Fisher, Vladimir Markov, Wil Yegelwel, Will Farr, woejozney, yolhaj , yureq , Zach A, Zad Rafi and Zhengchen Cai.

References

MacKay, David J. C. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, New York.

License

A repository containing all of the files used to generate this chapter is available on GitHub.

Original Computing Environment

```
writeLines(readLines(file.path(Sys.getenv("HOME"), ".R/Makevars")))
```

```
CC=clang

CXXFLAGS=-O3 -mtune=native -march=native -Wno-unused-variable -Wno-unused-function -Wno-macr
CXX=clang++ -arch x86_64 -ftemplate-depth-256

CXX14FLAGS=-O3 -mtune=native -march=native -Wno-unused-variable -Wno-unused-function -Wno-ma
CXX14=clang++ -arch x86_64 -ftemplate-depth-256
```

```
sessionInfo()
```

```
R version 4.3.2 (2023-10-31)
Platform: x86_64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.4.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRblas.0.dylib
```

```
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlapack.dylib;

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] rstan_2.32.6       StanHeaders_2.32.7 colormap_0.1.4

loaded via a namespace (and not attached):
 [1] gtable_0.3.4       jsonlite_1.8.8     compiler_4.3.2     Rcpp_1.0.11
 [5] stringr_1.5.1      parallel_4.3.2     gridExtra_2.3      scales_1.3.0
 [9] yaml_2.3.8         fastmap_1.1.1      ggplot2_3.4.4      R6_2.5.1
[13] curl_5.2.0         knitr_1.45         tibble_3.2.1       munsell_0.5.0
[17] pillar_1.9.0       rlang_1.1.2        utf8_1.2.4         V8_4.4.1
[21] stringi_1.8.3      inline_0.3.19      xfun_0.41          RcppParallel_5.1.7
[25] cli_3.6.2          magrittr_2.0.3     digest_0.6.33      grid_4.3.2
[29] lifecycle_1.0.4    vctrs_0.6.5        evaluate_0.23      glue_1.6.2
[33] QuickJSR_1.0.8     codetools_0.2-19   stats4_4.3.2       pkgbuild_1.4.3
[37] fansi_1.0.6        colorspace_2.1-0   rmarkdown_2.25     matrixStats_1.2.0
[41] tools_4.3.2        loo_2.6.0          pkgconfig_2.0.3    htmltools_0.5.7
```

Stan
Program 1 `fit\_window.stan`

```
data {
  int<lower=1> N; // Number of observations
  real xs[N];     // Observed x positions (m)
  real ys[N];     // Observed y positions (m)
}

transformed data {
  real L = 3; // Maximum size (m)

  // Sufficient statistics
  real min_x = min(xs);
  real max_x = max(xs);
  real min_y = min(ys);
  real max_y = max(ys);
}

parameters {
  real<lower=-L, upper=min_x> left;   // Position of left window edge (m)
  real<lower=max_x, upper=+L> right;  // Position of right window edge (m)
  real<lower=-L, upper=min_y> bottom; // Position of bottom window edge (m)
  real<lower=max_y, upper=+L> top;    // Position of top window edge (m)
}

model {
  // Implicit uniform prior density function within boundaries

  // Observational model
  target += - N * ( log(right - left) + log(top - bottom));
}

generated quantities {
  real min_x_pred;
  real max_x_pred;
  real min_y_pred;
  real max_y_pred;
  {
    real xs_pred[N];
    real ys_pred[N];
    for (n in 1:N) {
      xs_pred[n] = uniform_rng(left, right);
      ys_pred[n] = uniform_rng(bottom, top);
    }

    min_x_pred = min(xs_pred);
    max_x_pred = max(xs_pred);
    min_y_pred = min(ys_pred);
    max_y_pred = max(ys_pred);
  }
}
```