Homework 4

David Lewis

March 10, 2025

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ----------------------------------------------
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
```

Question 1

Define the rhombus detector

```r
is_in_rhombus <- function(x, y, long_diagonal) {
    R <- abs(long_diagonal / 2)
    return(2 * abs(x) + 1 * abs(y) < R)
}
```

Create a Shotgun

```r
shotgun <- function(N, x1, x2, y1, y2, label_func) {
    x <- runif(N, x1, x2)
    y <- runif(N, y1, y2)
    df <- data.frame(x = x, y = y)
    df$in_dist <- apply(df, 1, function(x) label_func(x[1], x[2]))
    return(df)
}


square_shotgun <- function(N, long_diagonal, label_func = is_in_rhombus) {
    R <- abs(long_diagonal / 2)
    label_func <- function(x, y) is_in_rhombus(x, y, long_diagonal =
↪   long_diagonal)
    shotgun(N, x1 = -R, x2 = R, y1 = -R, y2 = R, label_func = label_func)
}
```

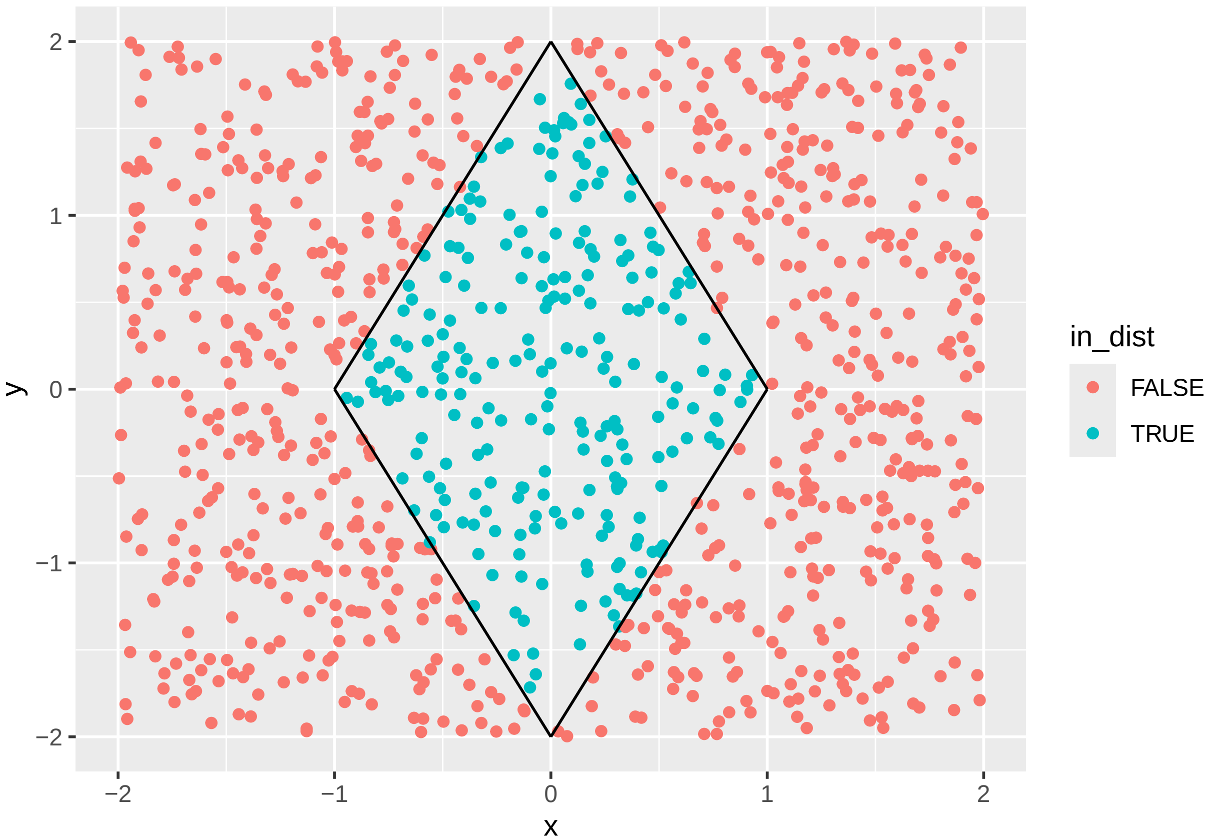
Calculate Area

- N = number of points (samples)
- LD = Long diagonal length
- \$A_{square} = LD^2
- A_{square} = (Area Inside Rhombus) + (Area Outside Rhombus)
- $1 = \frac{\text{Points Inside Rhombus}}{N} + \frac{\text{Points Outside Rhombus}}{N}$
- (Area inside Rhombus) $\approx A_{square} \cdot \frac{\text{Points Inside Rhombus}}{N}$

```r
area_rhombus <- function(N, long_diagonal = 4) {
    df <- square_shotgun(N, long_diagonal)
    ratio_of_points_in_rhombus <- nrow(df[df$in_dist, ]) / N
    area_of_square <- long_diagonal^2
    return(area_of_square * ratio_of_points_in_rhombus)
}
```

Define rhombus maker (for fun) visualization

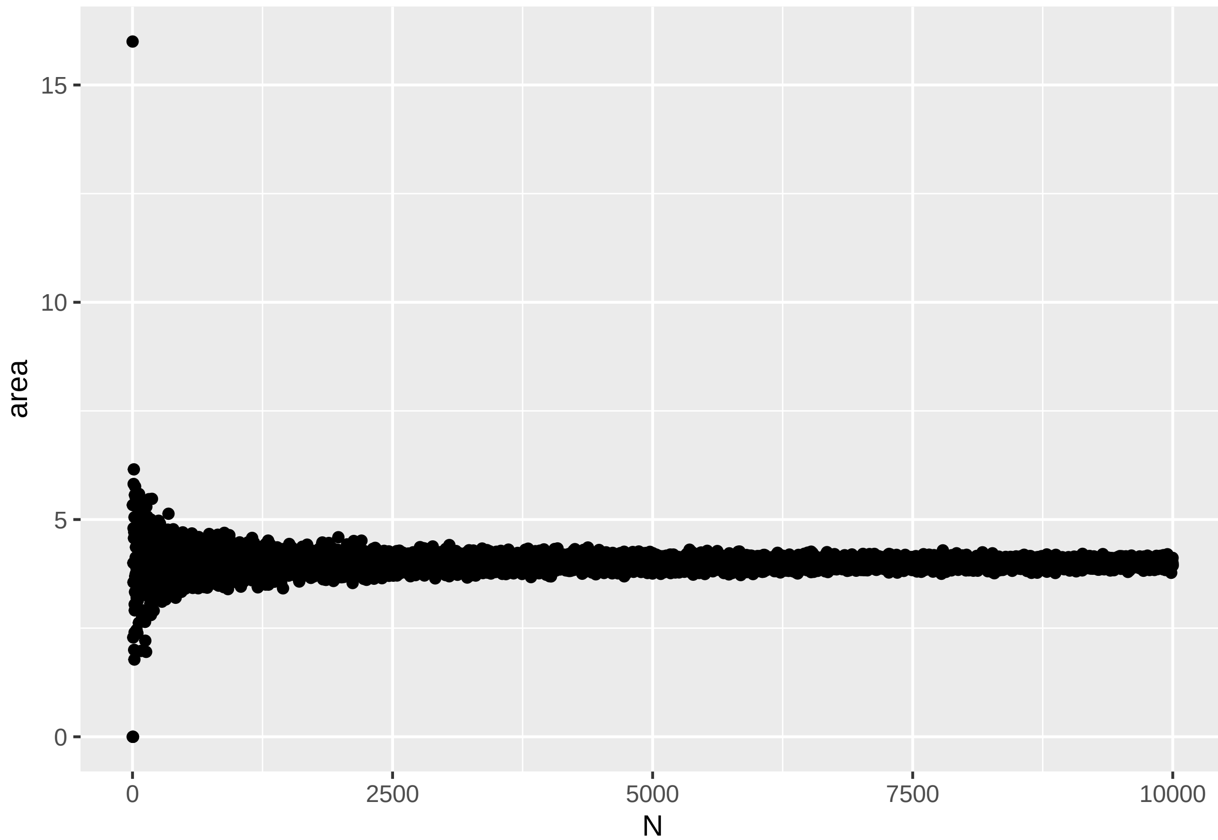
```r
rhombus <- function(long_diagonal = 4, long_diagonal_ratio = 2) {
    long_R <- long_diagonal / 2
    short_R <- (long_diagonal / long_diagonal_ratio) / 2
    list(
        left = c(-short_R, 0),
        right = c(short_R, 0),
        top = c(0, long_R),
        bottom = c(0, -long_R)
    )
}



plot_rhombus <- function(long_diagonal = 4, sampling_func =
↪   square_shotgun, N = 1000) {
    rhombus <- rhombus(long_diagonal = long_diagonal)

    ggplot(sampling_func(1000, long_diagonal)) +
        geom_point(aes(x = x, y = y, col = in_dist)) +
        annotate("segment", x = rhombus$left[1], y = rhombus$left[2], xend
        ↪   = rhombus$bottom[1], yend = rhombus$bottom[2]) +
        annotate("segment", x = rhombus$bottom[1], y = rhombus$bottom[2],
        ↪   xend = rhombus$right[1], yend = rhombus$right[2]) +
        annotate("segment", x = rhombus$right[1], y = rhombus$right[2],
        ↪   xend = rhombus$top[1], yend = rhombus$top[2]) +
        annotate("segment", x = rhombus$top[1], y = rhombus$top[2], xend =
        ↪   rhombus$left[1], yend = rhombus$left[2])
}
plot_rhombus()
```



#

Question 2

```
N <- 10000
x <- data.frame(area = sapply(1:N, area_rhombus), N = 1:N)
ggplot(data = x, aes(x = N, y = area)) +
    geom_point()
```



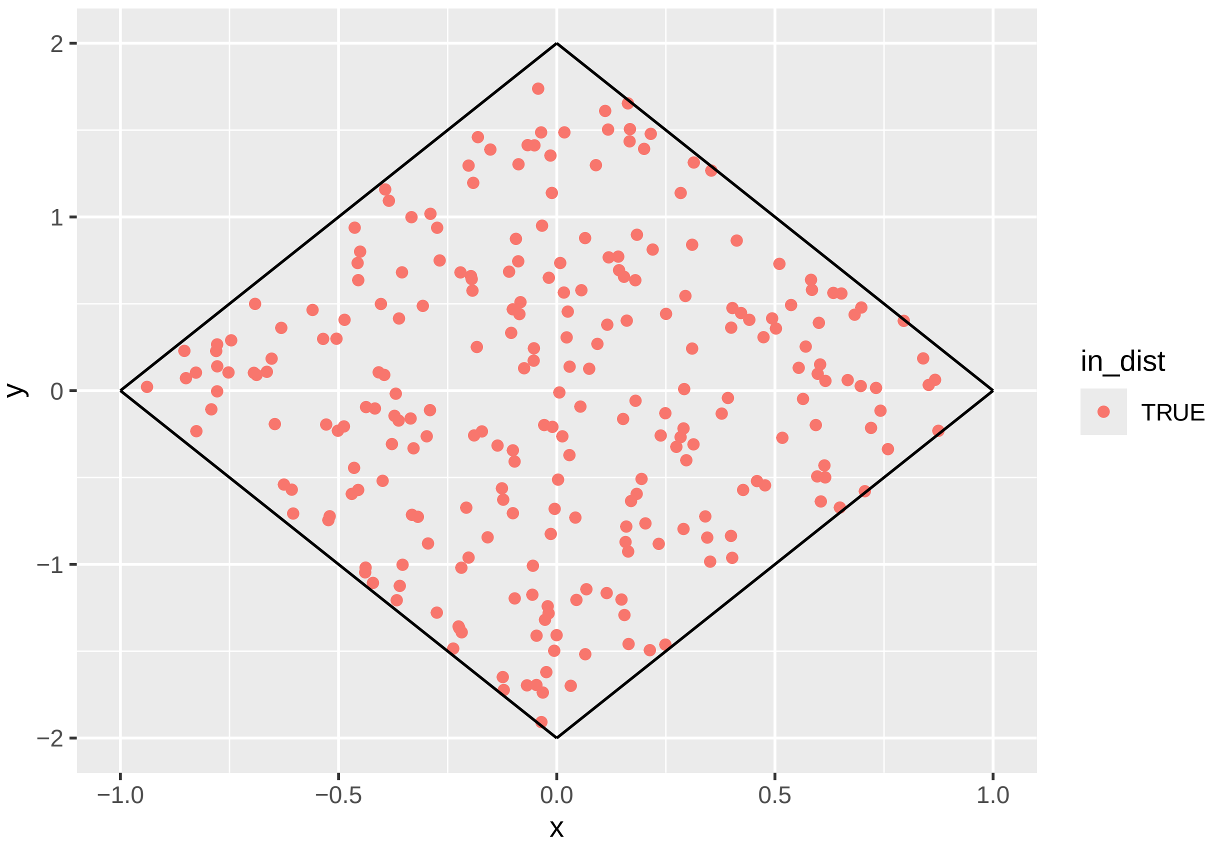

```r
area_rhombus_true <- function(long_diagonal = 4) {
    short_diagonal <- long_diagonal / 2
    ((short_diagonal) * (long_diagonal)) / 2
}
area_rhombus_true()
```

```
## [1] 4
```

The true area is 4 and the estimation algorithm does converge, though slowly.

Question 3

```
rejection_shotgun <- function(N, long_diagonal = 4, label_func =
↪   is_in_rhombus) {
    df <- square_shotgun(N, long_diagonal = long_diagonal, label_func =
↪   label_func)
    in_dist <- df[df$in_dist, ] # rejection
    return(in_dist)
}


plot_rhombus(sampling_func = rejection_shotgun)
```



Question 4

Define target and proposal functions

```
target <- function(x) dbeta(x, shape1 = 2, shape2 = 5)
proposal <- function(x) dbeta(x, shape1 = 1, shape2 = 1)
```

Sample from distribution and reject

```
sample_dist <- function(f = target, g = proposal, N = 1000) {
    ratio.f.g <- function(x) f(x) / g(x)
    M <- optimize(ratio.f.g, c(0, 1), maximum = TRUE)$objective
    accepted_samples <- 0
    n_trials <- 0

    while (length(accepted_samples) < N) {
        x <- runif(1)
        u <- runif(1)
        if (u < f(x) / (M * g(x))) {
            accepted_samples <- c(accepted_samples, x)
        }
        n_trials <- n_trials + 1
    }
    acceptance_ratio <- N / n_trials
    return(list(accepted_samples = accepted_samples, acceptance_ratio =
     ↪  acceptance_ratio, n_trials = n_trials, ratio_func = ratio.f.g))
}
output <- sample_dist()
```

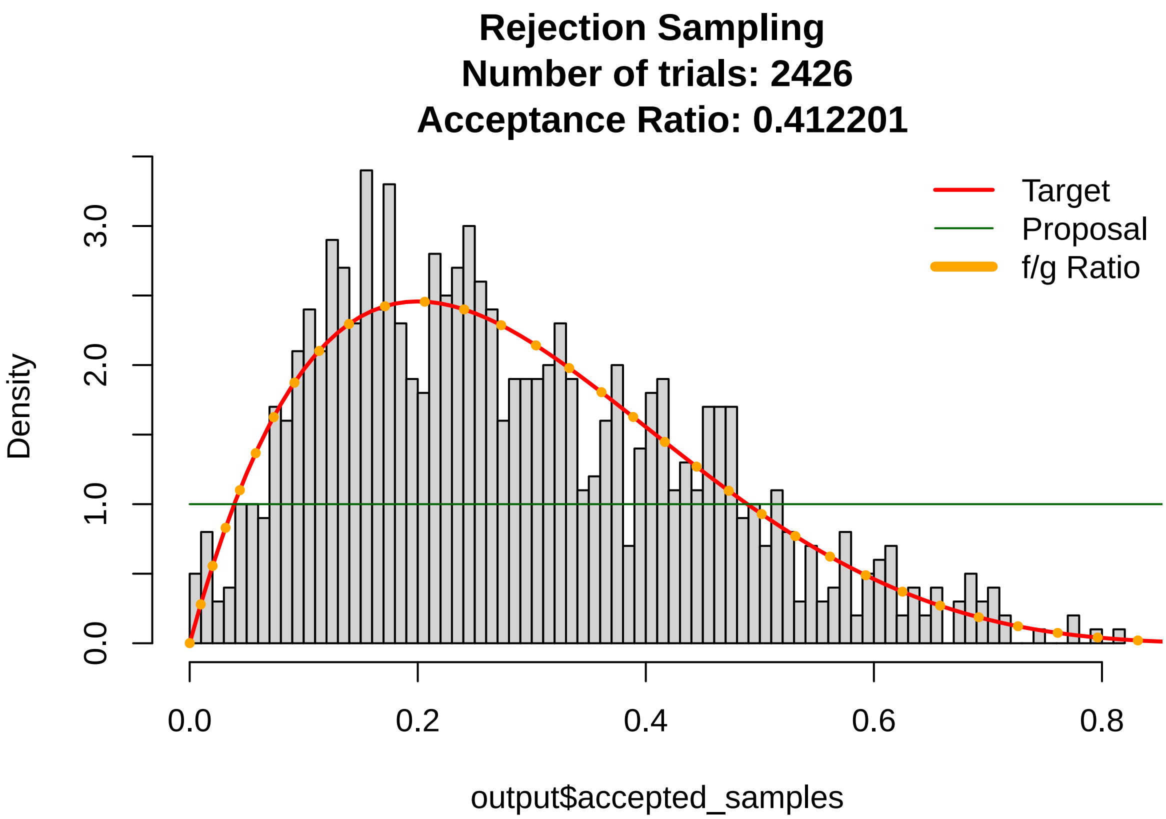
Question 5

The function used as the proposal function is the uniform function because the ratio distribution of f and g is equal to f, our target distribution. As the probabilities are optimized for this ratio distribution, to get the closest to the target distribution, the ratio distribution should be equal to the target distribution.

The acceptance ratio is: 0.4122012

Question 6

```
hist(output$accepted_samples,
    freq = FALSE, breaks = 100,
    main = sprintf("Rejection Sampling \n Number of trials: %i \n
     ↪  Acceptance Ratio: %f", output$n_trials, output$acceptance_ratio)
)
curve(target, 0, 1, add = TRUE, col = "red", lwd = 2)
curve(proposal, 0, 1, add = TRUE, col = "darkgreen", lty = 1)
curve(output$ratio_func(x), 0, 1, add = TRUE, col = "orange", lty = 3, lwd
 ↪  = 5)

legend("topright",
    legend = c("Target", "Proposal", "f/g Ratio"),
    col = c("red", "darkgreen", "orange"),
    lty = c(1), lwd = c(2, 1, 5), bty = "n"
)
```


Rejection Sampling
Number of trials: 2426
Acceptance Ratio: 0.412201
Target
Proposal
f/g Ratio
Density
0.0
1.0
2.0
3.0
0.0
0.2
0.4
0.6
0.8
output$accepted_samples