GR5065 Homework 2

Cindy Li cxl2103

Due February 21, 2022 at 4PM

1 Gross Domestic Product

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
FRED <- "https://fred.stlouisfed.org/graph/fredgraph.csv"
dataset <- read_csv(paste0(FRED, "?id=A261RL1Q225SBEA,A191RL1Q225SBEA"), na = ".") %>%
na.omit %>%
rename(quarter_startdate = DATE, GDI = A261RL1Q225SBEA, GDP = A191RL1Q225SBEA)
```

```
## Rows: 299 Columns: 3

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (2): A261RL1Q225SBEA, A191RL1Q225SBEA
## date (1): DATE
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
tail(dataset, n = 7)
```

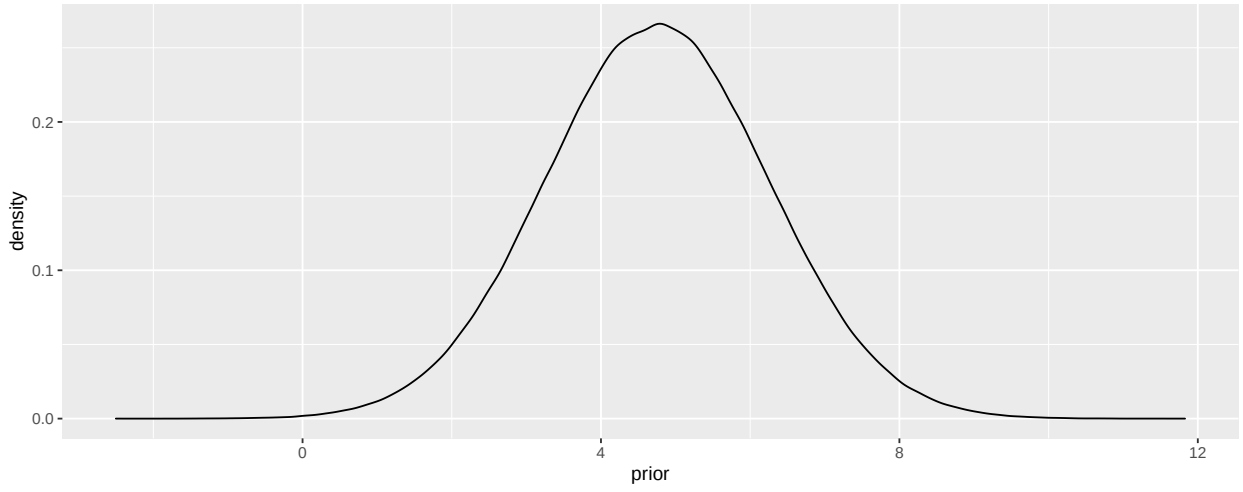
```
## # A tibble: 7 x 3
##   quarter_startdate   GDI   GDP
##   <date>            <dbl> <dbl>
## 1 2020-01-01         -0.8  -5.1
```

```
## 2 2020-04-01        -32.7 -31.2
## 3 2020-07-01         24.4  33.8
## 4 2020-10-01         19.6   4.5
## 5 2021-01-01          6.3   6.3
## 6 2021-04-01          4.3   6.7
## 7 2021-07-01          5.8   2.3
```

1.1 Normal Prior

Based on what we see from previous quarters, and if we assume that growth will not be negative (i.e. GDI & GDP growth >= 0), then I believe that the mean and therefore expectation of the growth will be 4.75%, based on the last 3 quarters (I expect it to be a little lower than the previous period because of the events near the end of 2021), and that the standard deviation will be fairly small - around 1.5%.

```
#generate a normal distribution plot
library(ggplot2)

mean <- 4.75
std <- 1.5
prior <- rnorm(10^6, mean, std) #use a large number to symbolize infinity
prior_df <- data.frame(prior)
ggplot(prior_df, aes(x = prior)) +
  geom_density()
```



1.2 Bivariate Normal Likelihood

The bivariate normal likelihood of mu (to be) evaluated at the values of GDI and GDP growth for the 4th quarter of 2021 below, assuming we will fill in GDP and GDI values:

```
mu <- rnorm(10^5, sigma_I)

likelihood <- function(mu,
              GDI = GDI, #to be filled in
              GDP = GDP, #to be filled in
              sigma_I = 7/3,
```

```
              sigma_P = 7/3,
              rho = -1/10) {
  beta <- rho * sigma_P / sigma_I
  mu_yx <- mu + beta * (GDI - mu)
  sigma_yx <- sigma_P * sqrt(1 - rho ^ 2)
  return(dnorm(GDI, mu, sigma_I) *
        dnorm(GDP, mu, sigma_yx))}
```

1.3 Simulating Reported GDI and GDP Growth

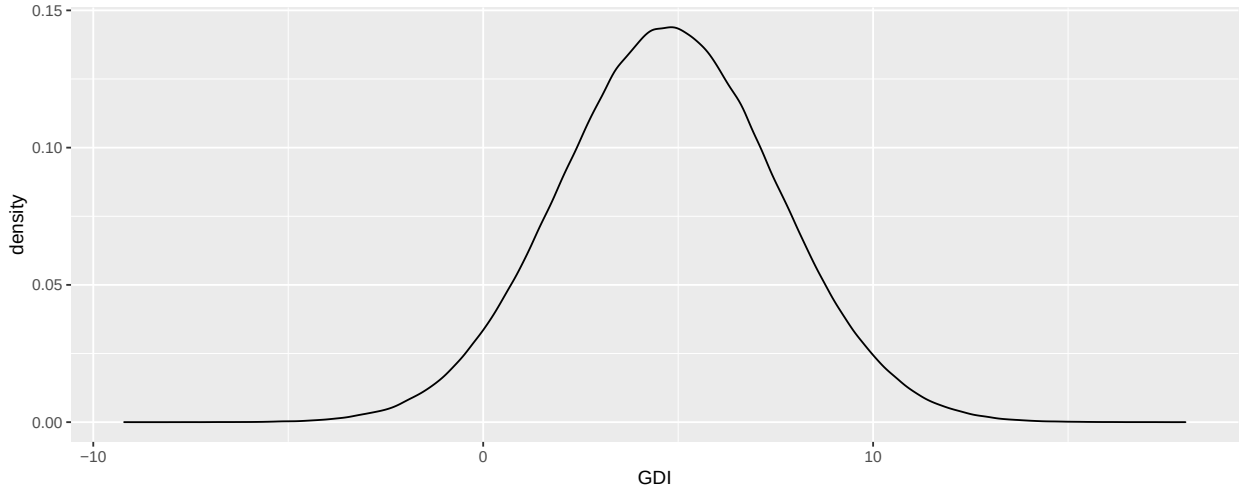
```
set.seed(20220220)
S <- 10^6

#my prior
mean <- 4.75
std <- 1.5
data <- rnorm(S, mean, std) #take 1 million draws of mu from my prior
df_mu <- data.frame(data)

#assumptions from previous sub-problem
mu <- data
sigma_I <- 7/3
sigma_P <- 7/3
rho <- -1/10
sigma <- matrix(c(sigma_I^2, sigma_I*sigma_P*rho,
                  sigma_I*sigma_P*rho, sigma_P^2),
                  ncol = 2) # Covariance matrix


GDI <- rnorm(S, mean = mu, sd = sigma_I)
GDP <- rnorm(S, mean = mu + rho * sigma_P / sigma_I * (GDI - 0),
           sd = sigma_P * sqrt( (1 + rho) * (1 - rho))) #function for y given x,
                                                        #rho, mu_x, sigma_x,
GDI_GDP <- data.frame(GDI, GDP)

sd(GDI_GDP$GDI)
```

```
## [1] 2.774879
```

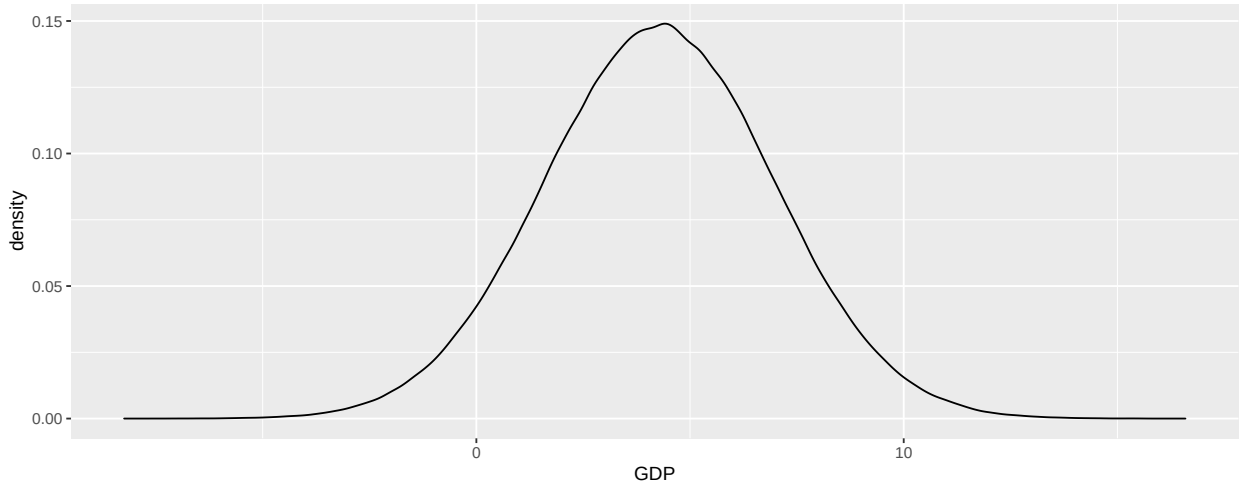
```
sd(GDI_GDP$GDP)
```

```
## [1] 2.697321
```

```
ggplot(GDI_GDP, aes(x = GDI)) +
  geom_density()
```




```
ggplot(GDI_GDP, aes(x = GDP)) +
  geom_density()
```



The estimated standard deviation for both series are ~2.77 and ~2.69, respectively - both of these are greater than 7/3 (~2.33). The difference comes from the fact that we are using two essentially randomized distributions - where the means from our prior we are using are also drawn randomly with some standard deviation. This therefore inflates our overall standard deviation. If, for instance, we lowered our initial standard deviation to 0, when we draw from our prior (mu = 4.75), the standard deviation in the simulation for reported GDP and GDI would be / get very close to 7/3.

1.4 Posterior

```
sigma_I <- 7/3
sigma_P <- 7/3
rho <- -1/10
GDI <- 'TBD' #to be filled in
GDP <- 'TBD' #to be filled in
mu <- rnorm(10^6, sigma_I)
```

```r
prior <- prior

likelihood <- function(mu, GDI, GDP,
                       sigma_I = 7/3,
                       sigma_P = 7/3,
                       rho = -1/10) {
  beta <- rho * sigma_P / sigma_I
  mu_yx <- mu + beta * (GDI - mu)
  sigma_yx <- sigma_P * sqrt(1 - rho ^ 2)
  return(dnorm(GDI, mu, sigma_I) *
        dnorm(GDP, mu, sigma_yx))}

numerator <- prior * likelihood

denominator <- integrate(numerator, lower = -Inf, upper = Inf, y = GDP,
                         mean_x = mu_GDI, mean_y = mu_GDP, sd_x = sigma_I,
                         sd_y = sigma_P, rho = rho)$value
#OR

denominator <- dnorm(y = GDP, mean = mu_GDP, sd = sigma_P)
```

1.5 Gross Domestic Output

Both GDI and GDP (the estimates) are subject to measurement error, especially in the 'after three months' immediate future - however, they tend to have errors in opposite directions.

GDO takes the average between the two which may balance out any over/under estimation produced by the two independent measures. Conceptually, it takes in 2 data-points as its data / likelihood function (and all previous GDP/GDI as a prior) - and can therefore produce a posterior given more data than using just GDI or GDP alone. Essentially, it is combining additional data to form a posterior, relative to GDI and GDP, which only contribute one additional data point to their respective posteriors. We could also consider it more step-wise, where if you input one data point at a time, the current GDP/GDI would serve as part of the prior for your second data point in one quarter. In any case, it is using more information to predict future GDP*.

On the other hand, GDP and GDI are subject to their own distribution and parameters, so using just GDP (or GDI) growth alone to estimate the posterior distribution of GDP* would seem like it is more comparing apples to apples.

2 Bowling

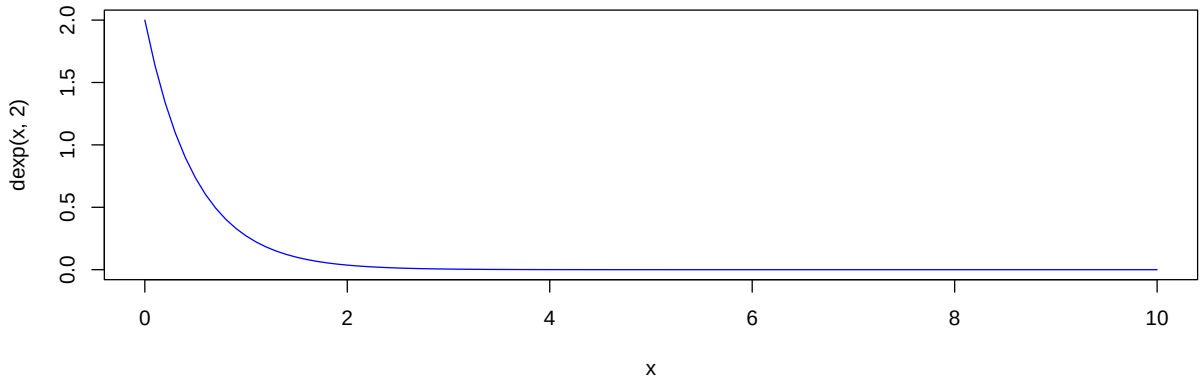
```r
# defines Pr(x, n = 10, theta = 1) and Omega as 0:10
source(file.path("..", "..", "Week04", "bowling.R"))
```

2.1 Prior

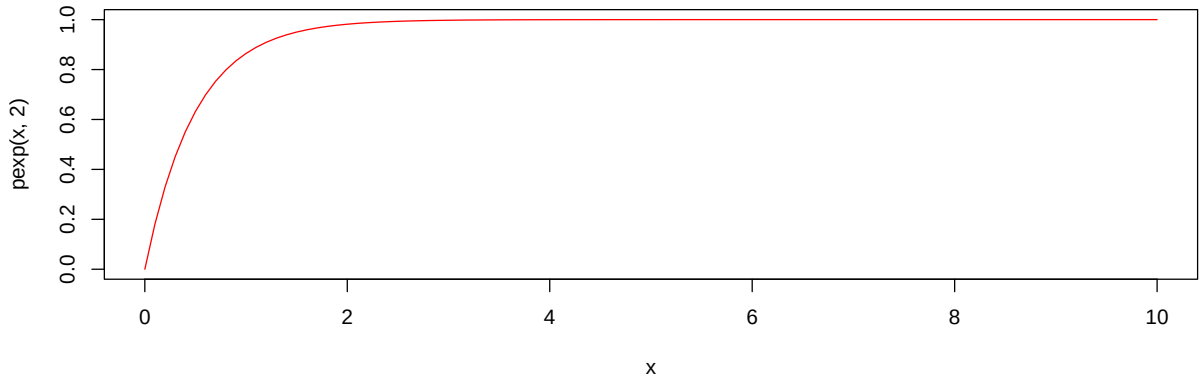
I would expect a woman bowling in the Olympics to be very good, and therefore theta to be quite small - somewhere between 0 and 1. I would therefore set the expectation for theta to be 0.5, somewhere in between. This would likely set expectation for the number of pins knocked down to be close to 9 or 10.

If E(theta) = 0.5, then rate = 1/0.5 = 2

```r
curve(dexp(x, 2), from = 0, to = 10, col = 'blue') #PDF
```




```r
curve(pexp(x, 2), from = 0, to = 10, col = 'red') # CDF
```



2.2 Marginal Probability of a Frame

```r
# defines Pr(x, n = 10, theta = 1) and Omega as 0:10
source(file.path("..", "..", "Week04", "bowling.R"))

theta <- 0:999
mu <- 0.5

joint_Pr_frame <- array(NA, dim = c(length(theta), length(Omega), length(Omega)),
                        dimnames = list(theta, Omega, Omega))
for (y in theta) {
  prior <- dexp(y, mu) # scalar
  cPr_x_1 <- Pr(Omega, n = 10, theta = y) # vector of size 11
```

```
  cPr_x_2 <- t(sapply(Omega, FUN = function(x_1) { # 11x11 matrix
    Pr(Omega, n = 10 - x_1, theta = y)
  }))
  joint_Pr_frame[y + 1, , ] <- prior * cPr_x_1 * cPr_x_2 # element wise multiplication
}
dim(joint_Pr_frame)
```

```
## [1] 1000   11   11
```

```
sum(joint_Pr_frame)
```

```
## [1] 1.270747
```

```
marginal_Pr_frame <- apply(joint_Pr_frame, MARGIN = 2:3, FUN = sum)
dim(marginal_Pr_frame) #confirms that it is 11 x 11
```

```
## [1] 11 11
```

```
marginal_Pr_frame #shown below
```

```
##              0           1           2           3           4           5
## 0  0.001514610 0.001636375 0.001780065 0.001952379 0.002163149 0.002427473
## 1  0.001854988 0.002017933 0.002213348 0.002452387 0.002752182 0.003140637
## 2  0.002318150 0.002542753 0.002817517 0.003162147 0.003608745 0.004213928
## 3  0.002969407 0.003290511 0.003693315 0.004215376 0.004922946 0.005947032
## 4  0.003923333 0.004404122 0.005027388 0.005872344 0.007095673 0.009068968
## 5  0.005395709 0.006160551 0.007197826 0.008700310 0.011125463 0.016016198
## 6  0.007832406 0.009154588 0.011071110 0.014167562 0.020421360 0.000000000
## 7  0.012279449 0.014861399 0.019039248 0.027496944 0.000000000 0.000000000
## 8  0.021693133 0.027841554 0.040337524 0.000000000 0.000000000 0.000000000
## 9  0.047422466 0.069103362 0.000000000 0.000000000 0.000000000 0.000000000
## 10 0.671997750 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
##              6           7           8           9          10
## 0  0.002769941 0.003233918 0.003905116 0.004986642 0.00716237
## 1  0.003666963 0.004428438 0.005655590 0.008124681 0.00000000
## 2  0.005089626 0.006501137 0.009341933 0.000000000 0.00000000
## 3  0.007598216 0.010922769 0.000000000 0.000000000 0.00000000
## 4  0.013044608 0.000000000 0.000000000 0.000000000 0.00000000
## 5  0.000000000 0.000000000 0.000000000 0.000000000 0.00000000
## 6  0.000000000 0.000000000 0.000000000 0.000000000 0.00000000
## 7  0.000000000 0.000000000 0.000000000 0.000000000 0.00000000
## 8  0.000000000 0.000000000 0.000000000 0.000000000 0.00000000
## 9  0.000000000 0.000000000 0.000000000 0.000000000 0.00000000
## 10 0.000000000 0.000000000 0.000000000 0.000000000 0.00000000
```

```
sum(marginal_Pr_frame) #sum above 1
```

```
## [1] 1.270747
```

The sum of all the probabilities is 1.27. The sum should theoretically be 1, and computation accounts for the difference (we can not get theta to infinitely small sizes) - we are basically using large increments of theta - in this "infinite" range - so if we were to theoretically "add" up each tiny slice of the area under the curve, we could get close to 1; since we are starting E(theta) = 0.5, we look at increments of 0.5 + 1, etc. If we increased mu, our sum would be slightly larger, and vice versa. The way we would get very close to 1 is through integration, which we have not done above.

2.3 Data

Six matrices are presented below for the 6 games for Tun Meziou - in Palembang - Indonesia 2019.

```
game_1 <- matrix(c(
3, 7, NA,
10, NA, NA,
8, 1, NA,
9, 1, NA,
7, 3, NA,
8, 2, NA,
8, 1, NA,
9, 1, NA,
10, NA, NA,
7, 3, 8),
nrow = 10, ncol = 3, byrow = TRUE)

game_2 <- matrix(c(
9, 1, NA,
9, 1, NA,
9, 1, NA,
10, NA, NA,
4, 6, NA,
8, 1, NA,
7, 2, NA,
6, 3, NA,
9, 1, NA,
7, 3, 8),
nrow = 10, ncol = 3, byrow = TRUE)

game_3 <- matrix(c(
7, 0, NA,
7, 1, NA,
9, 0, NA,
7, 2, NA,
8, 2, NA,
8, 0, NA,
8, 2, NA,
7, 2, NA,
7, 3, NA,
10, 8, 2),
nrow = 10, ncol = 3, byrow = TRUE)

game_4 <- matrix(c(
6, 3, NA,
9, 1, NA,
```

```
10, NA, NA,
7, 2, NA,
10, NA, NA,
6, 3, NA,
8, 2, NA,
9, 1, NA,
7, 3, NA,
8, 1, NA),
nrow = 10, ncol = 3, byrow = TRUE)

game_5 <- matrix(c(
9, 1, NA,
10, NA, NA,
10, NA, NA,
8, 2, NA,
8, 2, NA,
10, NA, NA,
10, NA, NA,
7, 2, NA,
7, 1, NA,
8, 2, 9),
nrow = 10, ncol = 3, byrow = TRUE)

game_6 <- matrix(c(
8, 2, NA,
7, 3, NA,
5, 4, NA,
7, 3, NA,
10, NA, NA,
9, 0, NA,
7, 1, NA,
7, 2, NA,
9, 1, NA,
6, 0, NA),
nrow = 10, ncol = 3, byrow = TRUE)
```

2.4 Posteriors

```
source(file.path("..", "..", "Week04", "bowling.R"))

game <- game_1
theta <- seq(from = 0.01, to = 5, by = 0.01)
x <- rexp(500, 2)
prior <- dexp(x, 1/theta)
likelihood <- sapply(theta, FUN = function(y) {
  sum(Pr(x = game[ ,1], n = 10, theta = y),
      (Pr(x = game[ ,2], n = 10 - game[ ,1], theta = y)), na.rm = TRUE)
  })

numerator <- prior + likelihood
denominator <- sum(numerator)
```

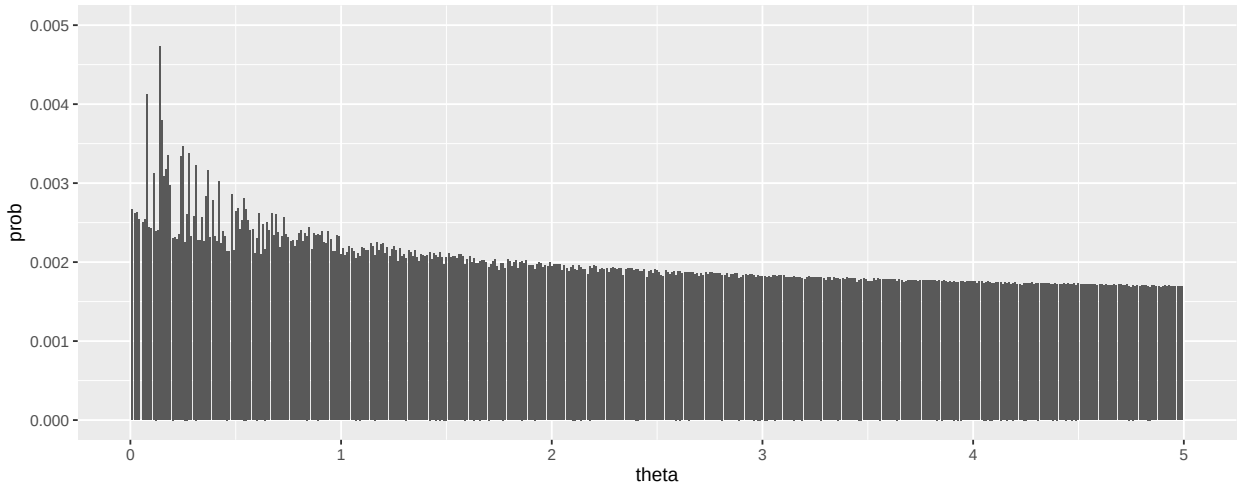
```
sum(numerator / denominator) #check that it = 1
```

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

```
ggplot(tibble(theta,
              prob = numerator / denominator)) +
  geom_col(aes(x = theta, y = prob)) + xlim(0, 5) + ylim(0, 0.005)
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```

```
## Warning: Removed 1 rows containing missing values (geom_col).
```

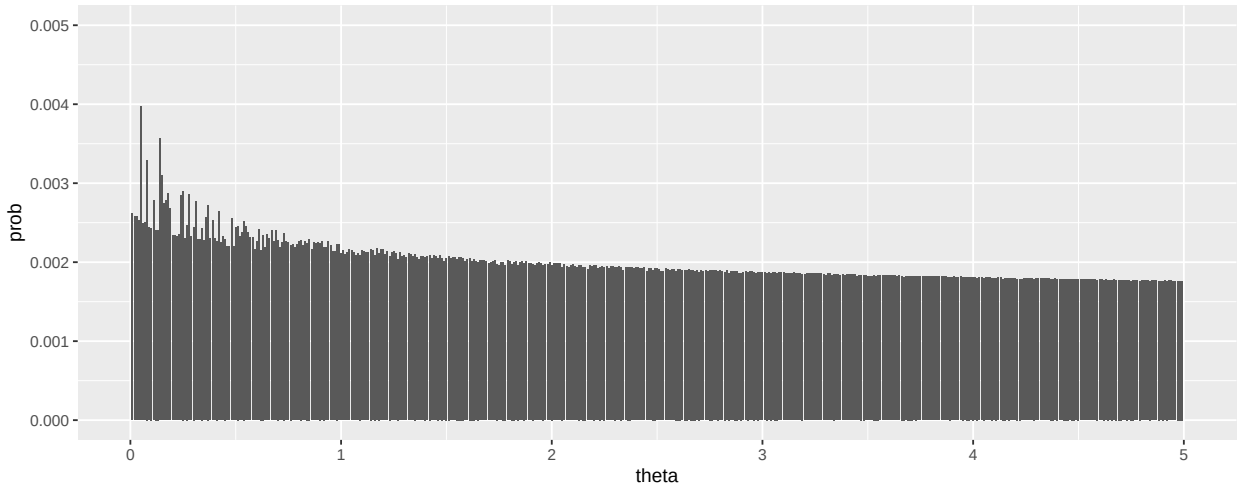



```
game2 <- rbind(game_1, game_2)

theta <- seq(from = 0.01, to = 5, by = 0.01)
prior <- dexp(x, 1/theta)
likelihood <- sapply(theta, FUN = function(y) {
  sum(Pr(x = game2[ ,1], n = 10, theta = y),
      (Pr(x = game2[ ,2], n = 10 - game2[ ,1], theta = y)), na.rm = TRUE)
  })

numerator <- prior + likelihood
denominator <- sum(numerator)

sum(numerator / denominator) #check that it = 1
```

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

```
ggplot(tibble(theta,
              prob = numerator / denominator)) +
  geom_col(aes(x = theta, y = prob)) + xlim(0, 5) + ylim(0, 0.005)
```

```
## Warning: Removed 1 rows containing missing values (geom_col).
```

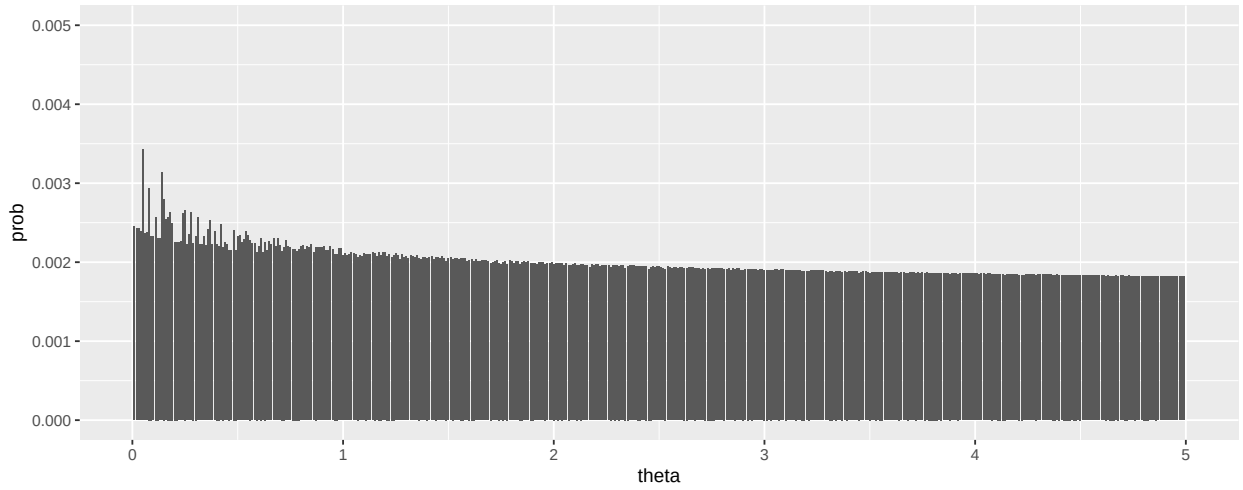



```
game3 <- rbind(game_1, game_2, game_3)

theta <- seq(from = 0.01, to = 5, by = 0.01)
prior <- dexp(x, 1/theta)
likelihood <- sapply(theta, FUN = function(y) {
  sum(Pr(x = game3[ ,1], n = 10, theta = y),
      (Pr(x = game3[ ,2], n = 10 - game3[ ,1], theta = y)), na.rm = TRUE)
  })

numerator <- prior + likelihood
denominator <- sum(numerator)

sum(numerator / denominator) #check that it = 1
```

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

```
ggplot(tibble(theta,
              prob = numerator / denominator)) +
  geom_col(aes(x = theta, y = prob)) + xlim(0, 5) + ylim(0, 0.005)
```

```
## Warning: Removed 1 rows containing missing values (geom_col).
```

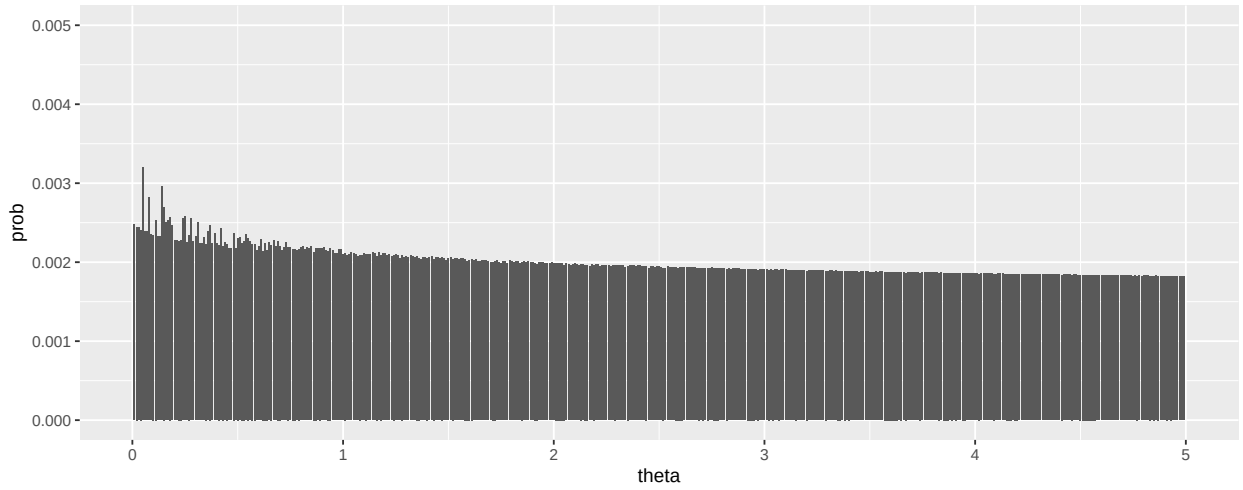



```
game4 <- rbind(game_1, game_2, game_3, game_4)

theta <- seq(from = 0.01, to = 5, by = 0.01)
prior <- dexp(x, 1/theta)
likelihood <- sapply(theta, FUN = function(y) {
  sum(Pr(x = game4[ ,1], n = 10, theta = y),
      (Pr(x = game4[ ,2], n = 10 - game4[ ,1], theta = y)), na.rm = TRUE)
  })

numerator <- prior + likelihood
denominator <- sum(numerator)

sum(numerator / denominator) #check that it = 1
```

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

```
ggplot(tibble(theta,
              prob = numerator / denominator)) +
  geom_col(aes(x = theta, y = prob)) + xlim(0, 5) + ylim(0, 0.005)
```

```
## Warning: Removed 1 rows containing missing values (geom_col).
```

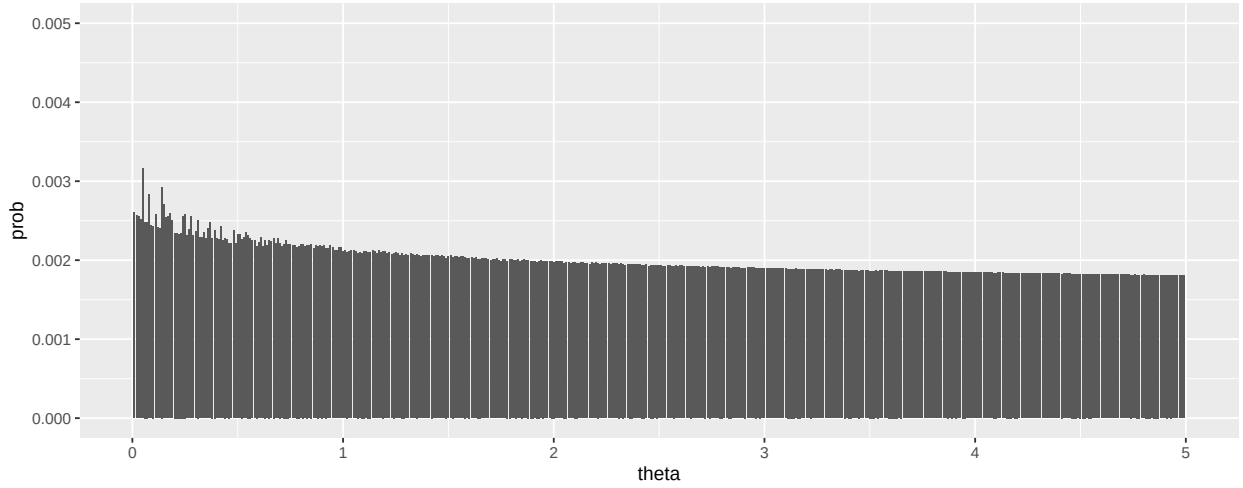



```
game5 <- rbind(game_1, game_2, game_3, game_4, game_5)

theta <- seq(from = 0.01, to = 5, by = 0.01)
prior <- dexp(x, 1/theta)
likelihood <- sapply(theta, FUN = function(y) {
  sum(Pr(x = game5[ ,1], n = 10, theta = y),
      (Pr(x = game5[ ,2], n = 10 - game5[ ,1], theta = y)), na.rm = TRUE)
  })

numerator <- prior + likelihood
denominator <- sum(numerator)

sum(numerator / denominator) #check that it = 1
```

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

```
ggplot(tibble(theta,
              prob = numerator / denominator)) +
  geom_col(aes(x = theta, y = prob)) + xlim(0, 5) + ylim(0, 0.005)
```

```
## Warning: Removed 1 rows containing missing values (geom_col).
```

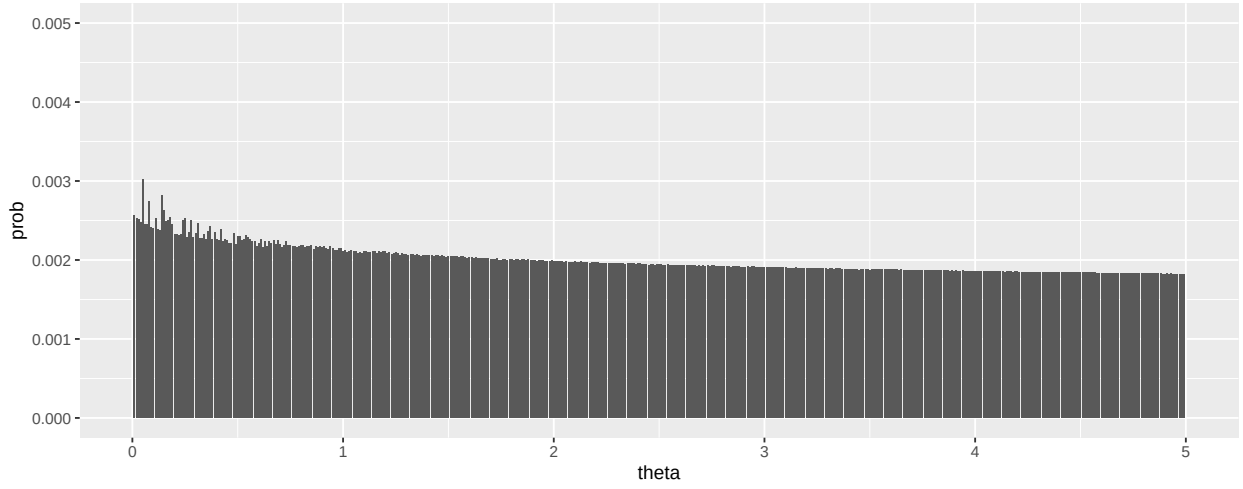



```
game6 <- rbind(game_1, game_2, game_3, game_4, game_5, game_6)
theta <- seq(from = 0.01, to = 5, by = 0.01)
prior <- dexp(x, 1/theta)
likelihood <- sapply(theta, FUN = function(y) {
  sum(Pr(x = game6[ ,1], n = 10, theta = y),
      (Pr(x = game6[ ,2], n = 10 - game6[ ,1], theta = y)), na.rm = TRUE)
  })

numerator <- prior + likelihood
denominator <- sum(numerator)

sum(numerator / denominator) #check that it = 1
```

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

```
ggplot(tibble(theta,
              prob = numerator / denominator)) +
  geom_col(aes(x = theta, y = prob)) + xlim(0, 5) + ylim(0, 0.005)
```

```
## Warning: Removed 1 rows containing missing values (geom_col).
```



This distribution was quite different than what I initially thought. However, we do see that the peak of the distribution is near 0, suggesting that the theta is most likely between somewhere in the 0.25-0.5 region. However, the probabilities do not drop as much after theta=2 than I expected - it is a much less steep curve than I thought it would be. I suppose it would take someone who is rolling strikes all the time to produce a curve that has a high likelihood of an extremely low theta. We also see the peaks smooth out after more data is added.

2.5 Simulate Score Distribution

```
#write a function called score
x <- games
score <- function(A) {
  stopifnot(class(A)=="matrix")
  if (A[j,1] = 10)   return(10 + A[j+1,1] + A[j+1,2] + A[j+2,1] + A[j+2,2])  # strike
  if (A[j,2] + game[j,1] = 10) return(10 + game[j+1,1] + game[j+1,2]) # spare
  if (game[j,1] + game[j,2] <= 10) return(game[j,1] + game[j,2]) # single roll
  if (game[10, 1] = 10 or game[10, 1] + game [10,2] = 10)
    return (game [10, 1] + game[10, 2] + game[10,3]) #last frame
  return(score)
}
#above shows potential logic for the scoring function

#take 1000 draws from exponential prior for theta
draws <- rexp(1000, rate = 1)
draws

#simulate 1 entire game of bowling for each of the 1000 thetas
for (y in draws) {
  x_1 <- Pr(Omega, n=10, theta = y)
  x_2 <- t(sapply(Omega, FUN=function(x_1) {
    Pr(Omega, n = 10 - x_1, theta = y)
  }))
  game <- rbind(x_1, x_2)

#score each game
scores <- apply(game, 2, score)

#histogram of scores
score <- scores
hist(score)

mode(score)
```