Lab 5 | Jon Lee

Code

Lab 5 | BINF 6310 | Spring 2020 | Jon Lee

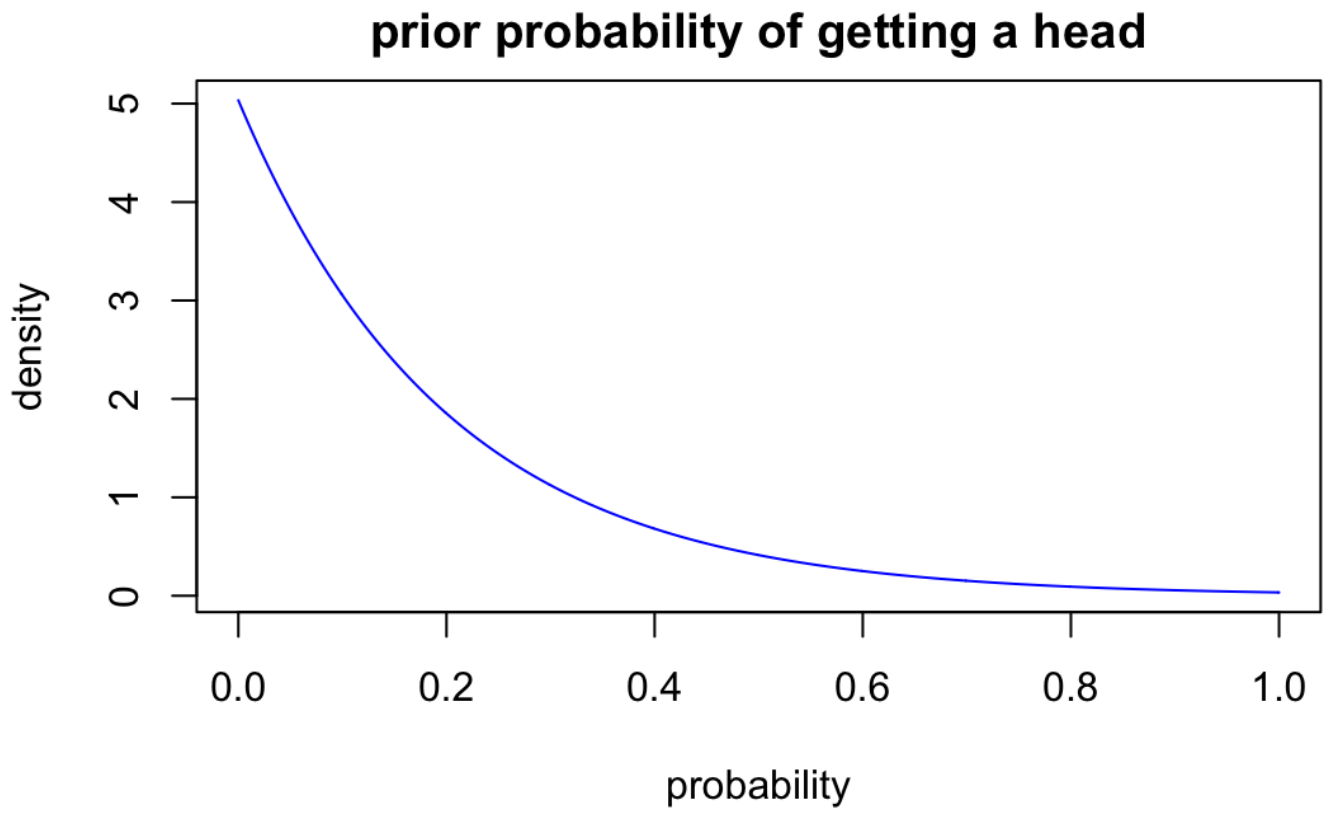
(1A) Plot the prior graph for a situation for a coin where the prior belief for p(head) is represented by the following R code:

```
dexp(x, rate = 5) / 0.9932621
```

for values of 0 ≤ x ≤ 1 and 0 otherwise. (We choose the denominator to make the Integral between 0 and 1 sum to 1).

Hide

```
probs <- seq(0, 1, 0.001)
prior <- dexp(probs, rate = 5)/0.9932621

#this <- dbeta(probs, prior, prior)
#plot(probs, this, xlim = c(0, 1), ylim = c(0, 2), type = "l")

plot(probs, prior, type = "l", xlab = "probability", ylab = "density", main = "prior pro
bability of getting a head", col = "blue")
```



(1B) Calculate the posterior graph with both the Metropolis algorithm and grid approximation for a case with 14 heads and 10 tails (where x = prob(head)). Show the two methods roughly agree. Compare these to a plot with a posterior for new data of 14 heads and 10 tails with a prior with beta(40,40).

Hide

```
#for piror p = dexp(x, rate = 5) / 0.9932621
#metropolis algorithm with addition of 14 heads and 10 tails
numIterations <- 20000

postDist_m1 <- vector(length = numIterations)
piO_m1 <- 0.5

for(i in 1:numIterations)
{
    pO_m1 <- (dexp(piO_m1, rate = 5)/0.9932621)*dbinom(14, 24, piO_m1)
    piN_m1 <- piO_m1+rnorm(1, 0, sd = 0.01)

    if(piN_m1 > 1)
        piN_m1 = 1;
    if(piN_m1 < 0)
        piN_m1 = 0;

    pN_m1 <- (dexp(piN_m1, rate = 5)/0.9932621)*dbinom(14, 24, piN_m1)
    ratio_m1 <- pN_m1/pO_m1

    if(ratio_m1 > 1 || ratio_m1 >= runif(1))
        piO_m1 = piN_m1;

    postDist_m1[i] = piO_m1
}

hist_m1 <- hist(postDist_m1, breaks = 200, plot = FALSE)

#grid approximation with addition of 14 heads and 10 tails
postDist_g1 <- vector()
Vals_g1 <- hist_m1$mids

i_g1 <- 1
sum_g1 <- 0

for(x in Vals_g1)
{
    #postDist_g1[i_g1] <- dbeta(p, 10, 10)*dbinom(14, 24, p)
    postDist_g1[i_g1] <- (dexp(x, rate = 5)/0.9932621)*dbinom(14, 24, x)
    sum_g1 <- sum_g1 + postDist_g1[i_g1]
    i_g1 <- i_g1 + 1
}

gridApprox_1 <- postDist_g1/sum_g1
```

Hide

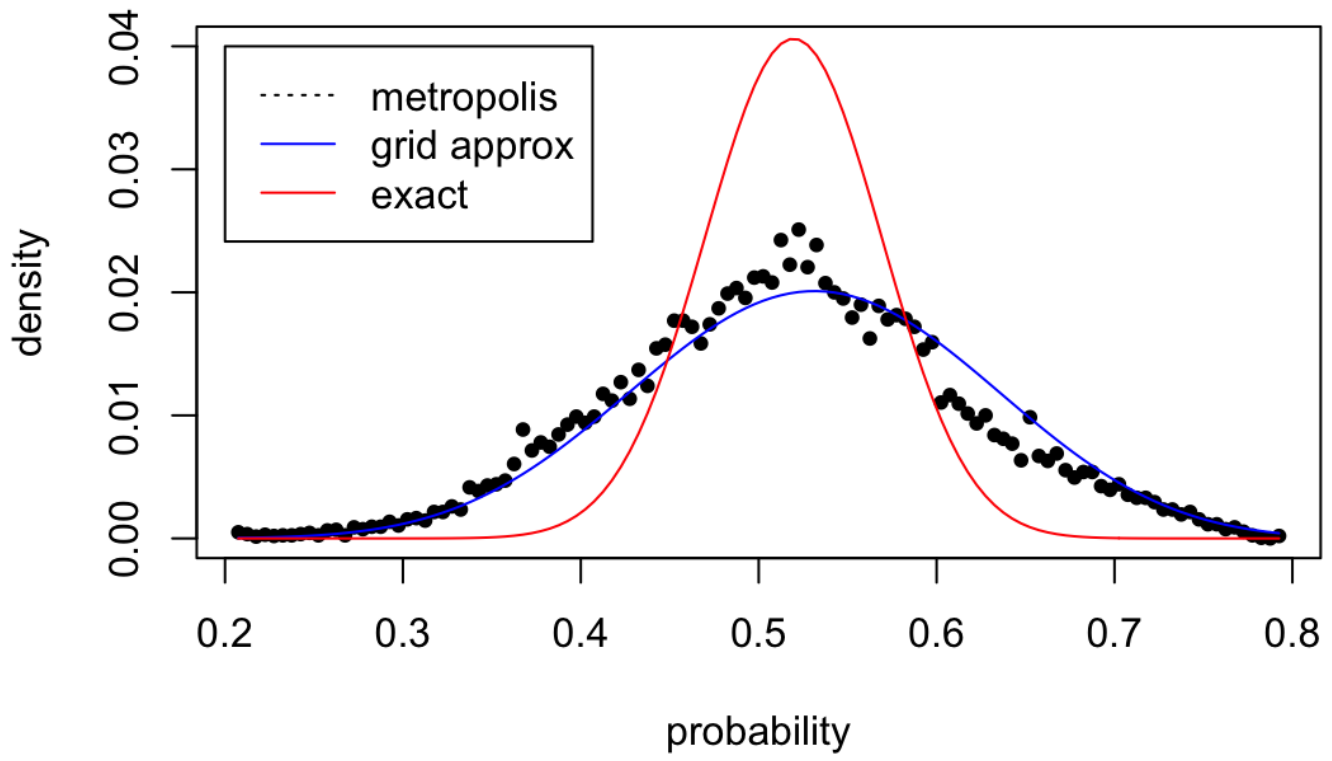
```
#grid approximation with addition of 14 heads and 10 tails; add 24 flips
postDist_g1 <- vector()
Vals_g1 <- hist_m1$mids

i_g1 <- 1
sum_g1 <- 0

for(x in Vals_g1)
{
    postDist_g1[i_g1] <- (dexp(x, rate = 5)/0.9932621)*dbinom(14, 24, x)
    sum_g1 <- sum_g1 + postDist_g1[i_g1]
    i_g1 <- i_g1 + 1
}

gridApprox_1 <- postDist_g1/sum_g1
```

Hide

```
#plot metropolis
plot(hist_m1$mids, hist_m1$counts/numIterations, xlab = "probability", ylab = "density",
pch = 20, ylim = c(0,0.04))
#plot grid approximation
lines(hist_m1$mids, gridApprox_1, col = "blue")
```

Hide

```
#plot exact
exSum_1 <- sum(dbeta(hist_m1$mids, 54, 50))
lines(hist_m1$mids, (dbeta(hist_m1$mids, 54, 50))/exSum_1, col = "red")
#legned
legend(0.2, 0.04, legend = c("metropolis", "grid approx", "exact"), col = c("black", "bl
ue", "red"), lty = c(3, 1, 1))
```



(So for the observation of 14 heads and 10 tails you will end up with a graph with three plots superimposed: (i) the Metropolis algorithm with an exp prior, (ii) grid approximation with an exp prior and (iii) exact analytical solution from a beta(40,40) prior make the plots different colors so you can visualize them…)

Answer: The metropolis and the grid approximation graphs are not as narrow as the exact analytical graph. Even thorugh they are more disperse, the center of all 3 do seem to line up.

(1C) Repeat the above calculation but for a case of 583 heads and 417 tails. (You may need to adjust your model step parameters to try and get the grid and Metropolis graphs to match up). How do the three posterior curves relate to each other now? Why does this plot look different than the plot in (1B)?

Hide

```
#for piror p = dexp(x, rate = 5) / 0.9932621
#metropolis algorithm with addition of 583 heads and 417 tails; add 1000 flips
postDist_m2 <- vector(length = numIterations)
piO_m2 <- 0.5

for(i in 1:numIterations)
{
    pO_m2 <- (dexp(piO_m2, rate = 5)/0.9932621)*dbinom(583, 1000, piO_m2)
    piN_m2 <- piO_m2+rnorm(1, 0, sd = 0.001)

    if(piN_m2 > 1)
        piN_m2 = 1;
    if(piN_m2 < 0)
        piN_m2 = 0;

    pN_m2 <- (dexp(piN_m2, rate = 5)/0.9932621)*dbinom(583, 1000, piN_m2)
    ratio_m2 <- pN_m2/pO_m2

    if(ratio_m2 > 1 || ratio_m2 >= runif(1))
        piO_m2 = piN_m2;

    postDist_m2[i] = piO_m2
}

hist_m2 <- hist(postDist_m2, breaks = 200, plot = FALSE)
```

Hide

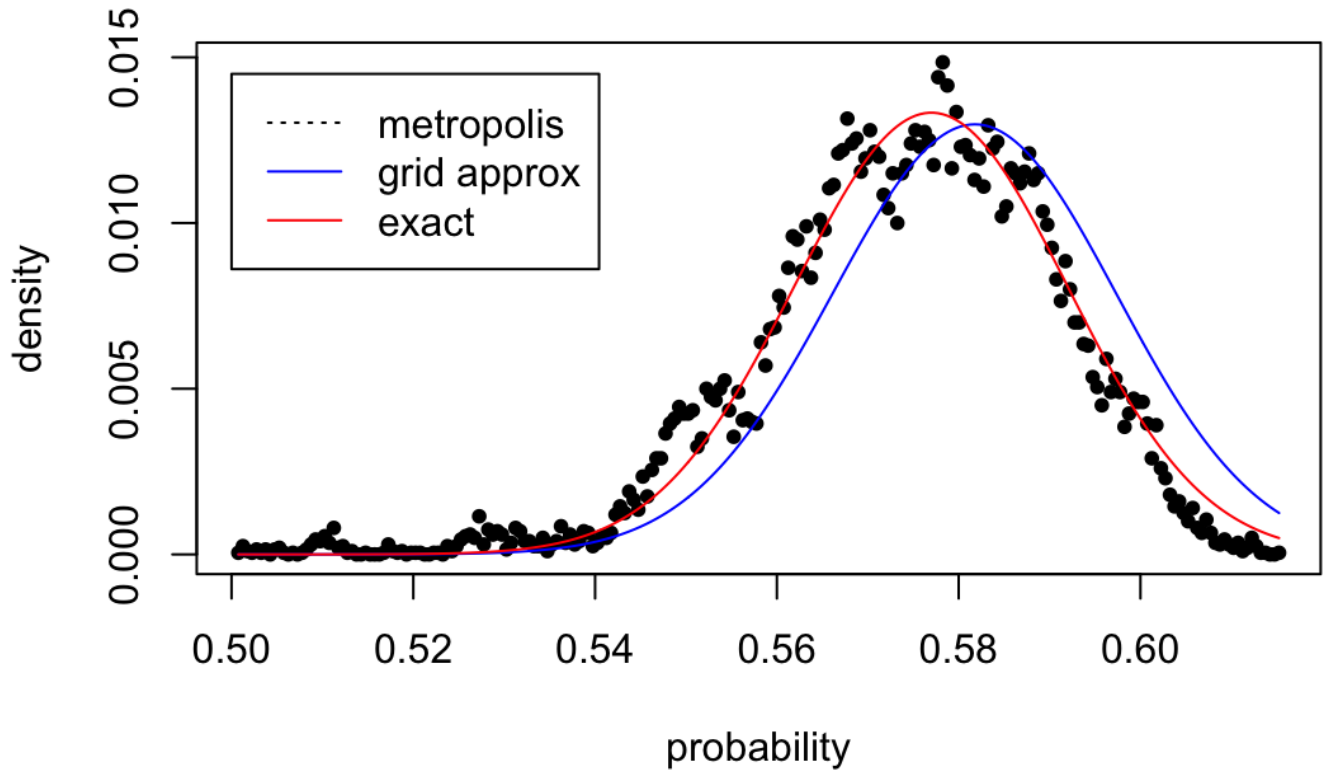
```
#grid approximation with addition of 583 heads and 417 tails; add 1000 flips
postDist_g2 <- vector()
Vals_g2 <- hist_m2$mids

i_g2 <- 1
sum_g2 <- 0

for(x in Vals_g2)
{
    postDist_g2[i_g2] <- (dexp(x, rate = 5)/0.9932621)*dbinom(583, 1000, x)
    sum_g2 <- sum_g2 + postDist_g2[i_g2]
    i_g2 <- i_g2 + 1
}

gridApprox_2 <- postDist_g2/sum_g2
```

Hide

```
#plot metropolis
plot(hist_m2$mids, hist_m2$counts/numIterations, pch = 20, xlab = "probability", ylab =
"density")
#plot grid approximation
lines(hist_m2$mids, gridApprox_2, col = "blue")
```

Hide

```
#plot exact
exSum_2 <- sum(dbeta(hist_m2$mids, 623, 457))
lines(hist_m2$mids, (dbeta(hist_m2$mids, 623, 457))/exSum_2, col = "red")
#legend
legend(0.5, 0.0145, legend = c("metropolis", "grid approx", "exact"), col = c("black",
"blue", "red"), lty = c(3, 1, 1))
```



Answer: All 3 graphs now are almos identical, the is some uncertainty on the middle of all 3 peaks, but the distributions are much more narrow and exact. The plots look so much different than before because of the addition of more data. More data helps to increase confidence in our algorithms and narrow the prediction.

HINTS: Watch your x-axis and y-axis scales here… Make sure all your curves are evaluated on the same set of x sequences. If someDistribution holds the results of your Metropolis walks, you can use…

```
myHist <- hist(someDist, breaks = 200, plot = FALSE)
```

to bin those results and then myHist$mids will hold the values for the x-axis for all your graphs.

Make sure the sum of each curve is equal to 1 when you graph them… So for the above histogram, you could plot…

```
plot(myHist$mids, myHist$counts/length(someDist), main = “Plot title here”)
```

You can add ylim() and xlim() as parameters to zoom in or zoom out as appropriate for that graph.

Then to add another distribution (that can be evaluated as a function ranging from 0 to 1) to the same graph:

```
aSum <- sum(nextDist(myHist$mids))
lines( myHist$mids, nextDist(myHist$mids)/aSum, col = "red")
```

(See lecture 7, slide 5 for an example of scaling…)