Lab 3: Intro to Decision Theory

Rebecca C. Steorts and Xingyu Yan

Task 1

```
# set seed
set.seed(123)

# data
sum_x = 1
n = 30
# prior parameters
a = 0.05; b = 1
# posterior parameters
an = a + sum_x
bn = b + n - sum_x
th = seq(0,1,length.out = 100)
like = dbeta(th, sum_x+1,n-sum_x+1)
prior = dbeta(th,a,b)
post = dbeta(th,sum_x+a,n-sum_x+b)
```

We now consider the loss function.

```
# compute the loss given theta and c
loss_function = function(theta, c){
  if (c < theta){
    return(10*abs(theta - c))
  } else{
    return(l = abs(theta - c))
  }
}
```

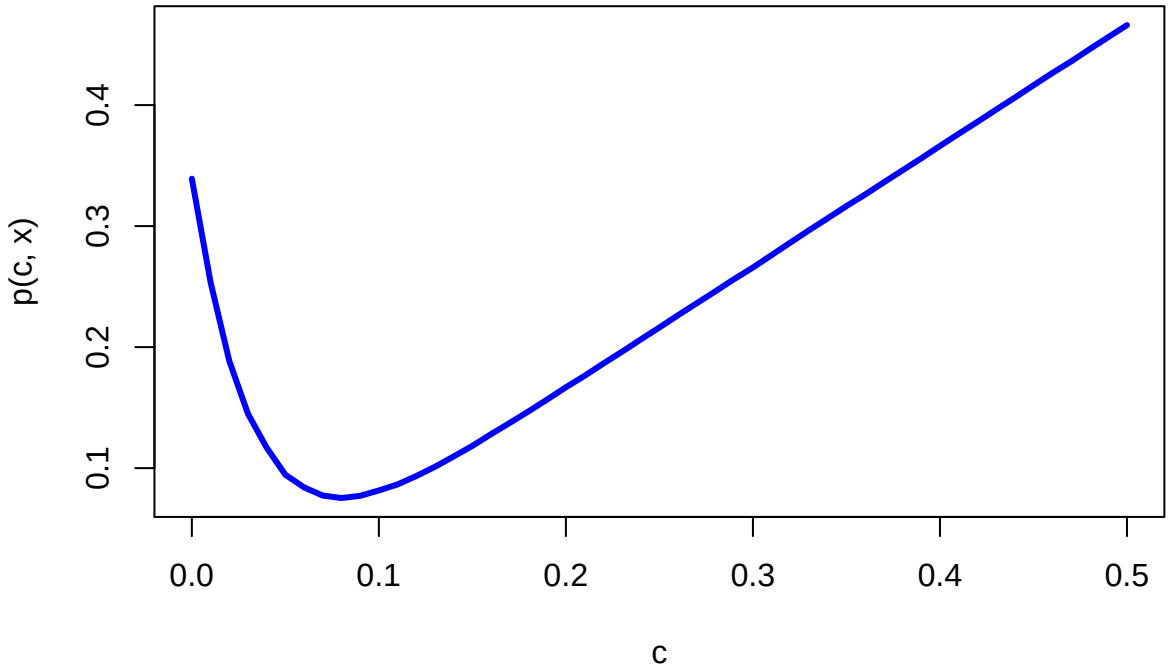
We now write a function **posterior_risk** which is a function of c, parameters a_prior and b_prior for the prior distribution of θ, the summation of x_i sum_x, the number of observations n, and also the number of random draws s.

```
# compute the posterior risk given c
# s is the number of random draws
posterior_risk = function(c, a_prior, b_prior, sum_x, n, s = 30000){
  # randow draws from beta distribution
  a_post = a_prior + sum_x
  b_post = b_prior + n - sum_x
  theta = rbeta(s, a_post, b_post)
  loss <- apply(as.matrix(theta),1,loss_function,c)
  # average values from the loss function
  risk = mean(loss)
}
# a sequence of c in [0, 0.5]
c = seq(0, 0.5, by = 0.01)
post_risk <- apply(as.matrix(c),1,posterior_risk, a, b, sum_x, n)
head(post_risk)
```

```
## [1] 0.33917940 0.25367603 0.18868962 0.14489894 0.11693106 0.09453471
```

We then look at the Posterior expected loss (posterior risk) for disease prevelance versus c.

```
# plot posterior risk against c
plot(c, post_risk, type = 'l', col='blue',
    lwd = 3, ylab ='p(c, x)' )
```




```
# minimum of posterior risk occurs at c = 0.08
(c[which.min(post_risk)])
```

```
## [1] 0.08
```

Task 2

We now consider task 2. We set $a = 0.05, 1, 0.05$ and $b = 1, 2, 10$. If we have different prior, the posterior risk is minimized at different c values. The optimal c depends on not only the data, but also the prior setting.

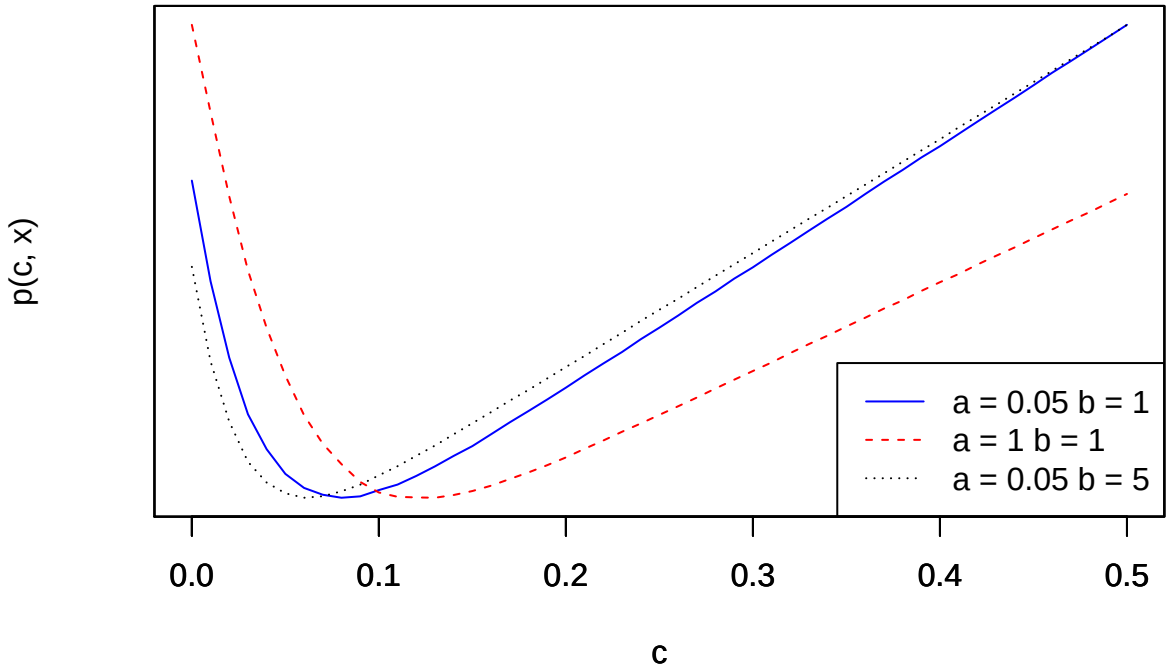
```
# set prior
as = c(0.05, 1, 0.05); bs = c(1, 1, 10)
post_risk = matrix(NA, 3, length(c))

# for each pair of a and b, compute the posterior risks
for (i in 1:3){
  a_prior = as[i]
  b_prior = bs[i]

  post_risk[i,] = apply(as.matrix(c), 1, posterior_risk, a_prior, b_prior, sum_x, n)
}

plot(c, post_risk[1,], type = 'l', col='blue', lty = 1, yaxt = "n", ylab = "p(c, x)")
par(new = T)
plot(c, post_risk[2,], type = 'l', col='red', lty = 2, yaxt = "n", ylab = "")
```

```
par(new = T)
plot(c, post_risk[3,], type = 'l', lty = 3, yaxt = "n", ylab = "")
legend("bottomright", lty = c(1,2,3), col = c("blue", "red", "black"),
       legend = c("a = 0.05 b = 1", "a = 1 b = 1", "a = 0.05 b = 5"))
```



Note there is a more automated solution but this is the most simple one and is completely correct.

Task 3

The Bayes procedure always picks c to be a little bigger than $\bar{x}$.

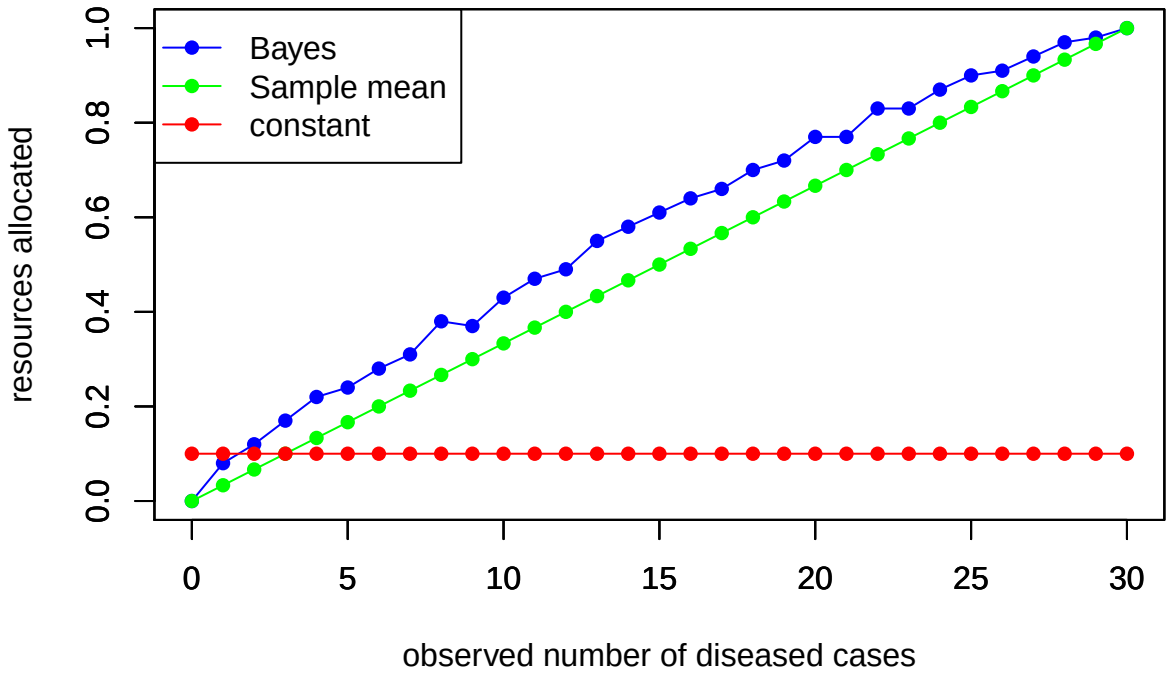
```
sum_xs = seq(0, 30)
min_c = matrix(NA, 3, length(sum_xs))

# find_optimal_C finds the optimal c under Bayes procedure
# function of sum of x, parameters for prior, number of observations, and number of random draws
find_optimal_C = function(sum_x, a_prior, b_prior, n, s = 500){
  c = seq(0, 1, by = 0.01)
  post_risk =  apply(as.matrix(c), 1, posterior_risk, a_prior, b_prior, sum_x, n, s)
  c[which.min(post_risk)]
}

min_c[1,] = apply(as.matrix(sum_xs), 1, find_optimal_C, a, b, n)
# find optimal c under sample mean
min_c[2,] = (sum_xs)/n
# constant c
min_c[3,] = 0.1

# plot
plot(sum_xs, min_c[1,], col='blue',type = 'o',pch = 16,
     ylab = "resources allocated", xlab = 'observed number of diseased cases',
```

```
     ylim = c(0,1))
par(new = T)
plot(sum_xs, min_c[2,], type = 'o', col='green',
     pch = 16, ylab = "", xlab = '', ylim = c(0,1))
par(new = T)
plot(sum_xs, min_c[3,], type = 'o',col = 'red',
     pch = 16, ylab = "", xlab = '', ylim = c(0,1))
legend("topleft", lty = c(1,1,1), pch = c(16,16,16),
       col = c("blue", "green", "red"),
       legend = c("Bayes", "Sample mean", "constant"))
```



Task 4

For all θ, the Bayes procedure has the lower frequentist risk than the sample mean.

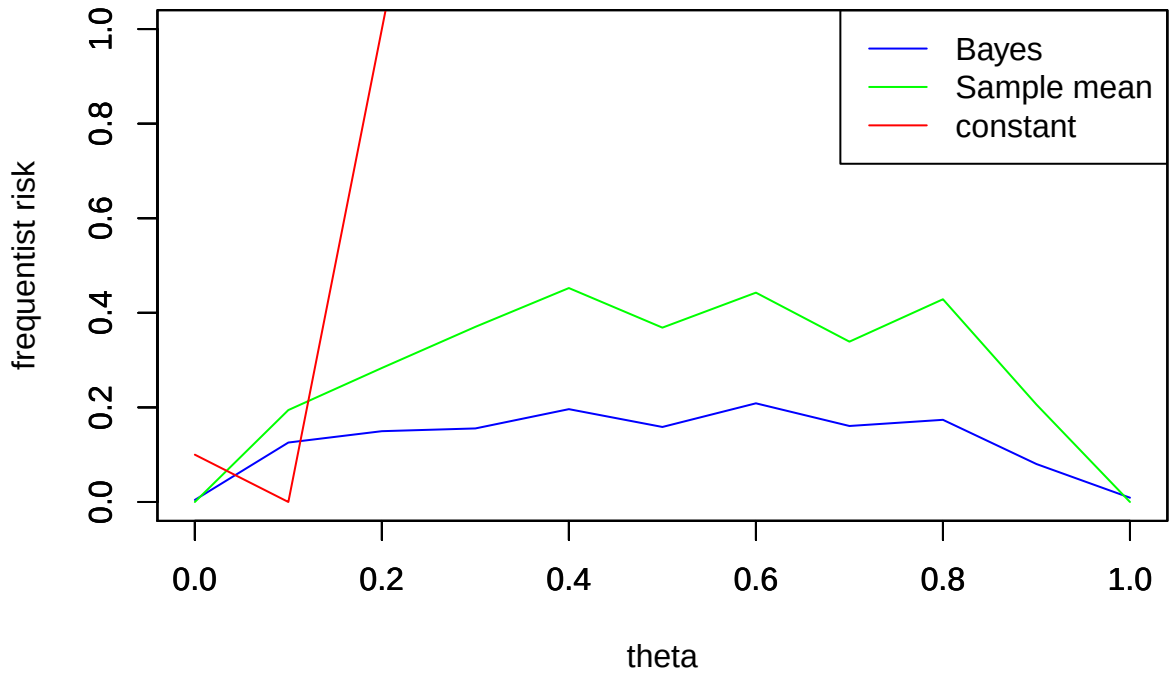
```
thetas = seq(0, 1, by=0.1)

# frequentist risk for the 3 estimators given a theta
frequentist_risk = function(theta){
  sum_xs = rbinom(100, 30, theta)
  Bayes_optimal = apply(as.matrix(sum_xs), 1, find_optimal_C, a, b, n, s = 100)
  mean_c = sum_xs / 30

  loss1 = apply(as.matrix(Bayes_optimal), 1, loss_function, theta = theta)
  loss2 = apply(as.matrix(mean_c), 1, loss_function, theta = theta )
  risk1 = mean(loss1)
  risk2 = mean(loss2)
  risk3 = loss_function(theta, 0.1)
  return(c(risk1, risk2, risk3))
}
```

```r
# given a sequance a theta, compute frequentist risk for each theta
R = apply(as.matrix(thetas), 1, frequentist_risk)

# plot
plot(thetas, R[1,], col='blue', type = "l",
     ylab = "frequentist risk", xlab = 'theta',ylim = c(0,1))
par(new = T)
plot(thetas, R[2,], type = 'l', col='green',
     ylab = "", xlab = '', ylim = c(0,1))
par(new = T)
plot(thetas, R[3,], type = 'l',col = 'red',
     ylab = "", xlab = '', ylim = c(0,1))
legend("topright", lty = c(1,1,1), col = c("blue", "green", "red"),
       legend = c("Bayes", "Sample mean", "constant"))
```



Please see a few remarks about Task 4 that will help you with interpreting the plot.

1. In this part of the problem, if we don't run enough samples, we can run into issues on the boundary.
2. It's important that we have enough random samples. What is enough. Probably around 30,000.
3. What should you observe? You should see that the Bayes estimator has lower risk than the sampler mean uniformly if you are able to draw enough random samples
4. In the event that you aren't able to draw enough random samples (perhaps you are running short on time or your computer is slow), then for a smaller number of samples, you'll observe that the Bayes estimator has slightly higher risk than the sampler mean. However, this is misleading here to report as you have not run enough samples and you should run more.