Q5

Varun Nayyar

30/04/2019

Bayesian Updating

You're on a cityrail train and you see that it's numbered 4031. You know that the train numbering cannot exceed 10,000 due to it's typefacing and you assume that each number is unique and in sequence.

- Explain why the likelihood P(X|N) is proportional to 1/N for 4031 <= N < 10000.

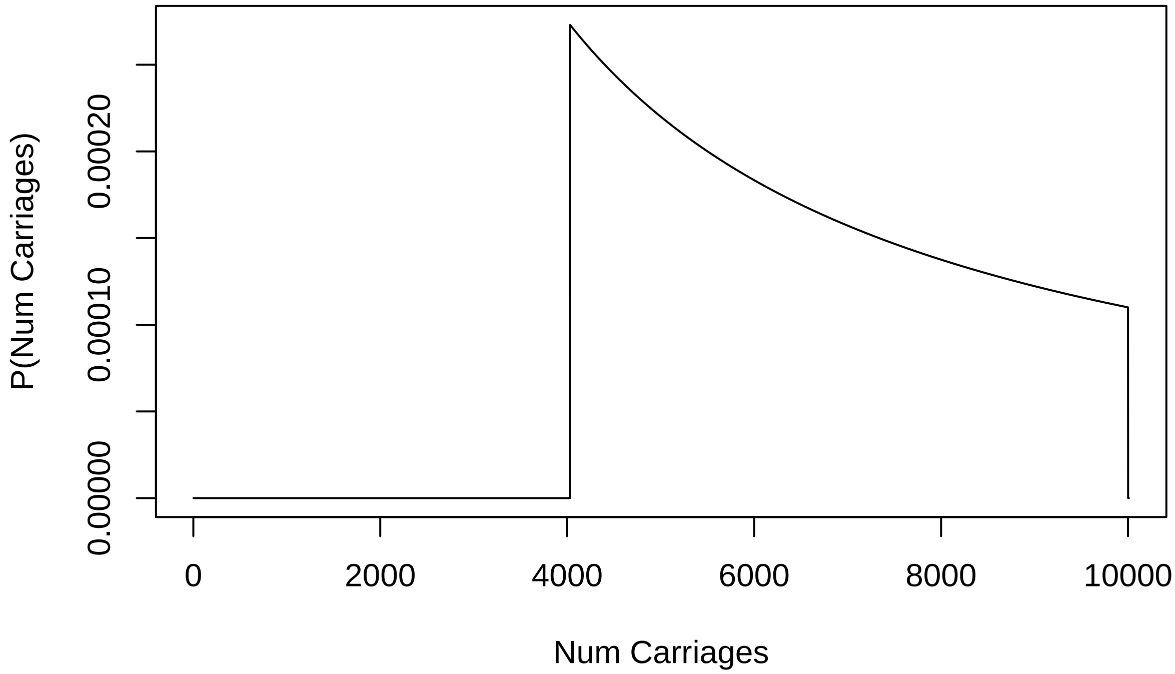
Consider the situation where there are 5000 carriages The probability of seeing number 4031 is 1/5000. Similarly, if there are 10,000 carriages the chance of seeing number 4031 is 1/10,000

Additionally, if there were only 2000 trains, you could never see a carriage numbered 4031, hence probability is 0. Since you know there can be no more than 10,000 carriages, you know any situation beyond that is also 0.

- Using a convenient prior, plot the posterior, remembering to normalize. Report median and mean.

We just use an implicit uniform prior here. This can be changed quite easily actually!

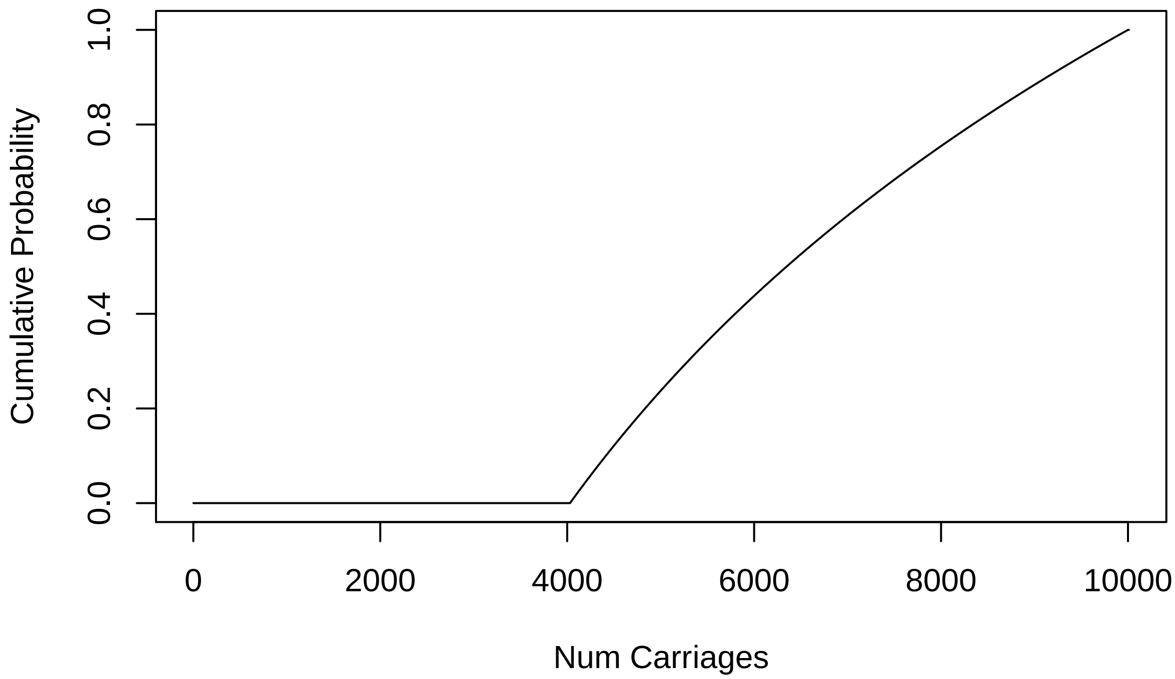
```r
obs = 4031
maxn = 10000
cityrail = function (obs, maxn){
  posterior = 1/(obs:maxn)
  # note we need the posterior to sum to 1.
  # so this is necessary!
  normalisation = sum(posterior)
  # pad with 0s from the front so it looks sensible.
  # also pad with 10 0s after for pizzaz
  c(rep(0, obs-1), posterior/normalisation, rep(0, 10))
}


post1 = cityrail(obs, maxn)
numc = 1:length(post1)

plot(numc, post1, "l",
     ylab="P(Num Carriages)", xlab="Num Carriages")
```



```
# This is a discrete distribution
# so the mean is the sum of carriages * probability(carriages)
# These are distributions
sum(numc * post1)
```

```
## [1] 6569.502
```

```
# lets plot the cumulative distribution
plot(numc, cumsum(post1), "l",
     xlab="Num Carriages", ylab="Cumulative Probability")
```



The median is the halfway point. We're gonna use this again, so let's make a function

```
medianp = function(posterior){
  # takes a discrete posterior and returns the median
  cdf_posterior = cumsum(posterior)
  # we want to find the crossing point of the cdf at 0.5
  # this can be done with sum(cdf <0.5).
  # the +1 is for the 1 indexing of R. (Not necessary in Python)
  median = sum(cdf_posterior < 0.5) + 1
  # this is not strictly accurate since the real median is between
  # `median` and `median-1` as defined above. Good enough approximation
  return(median)
}
medianp(post1)
```

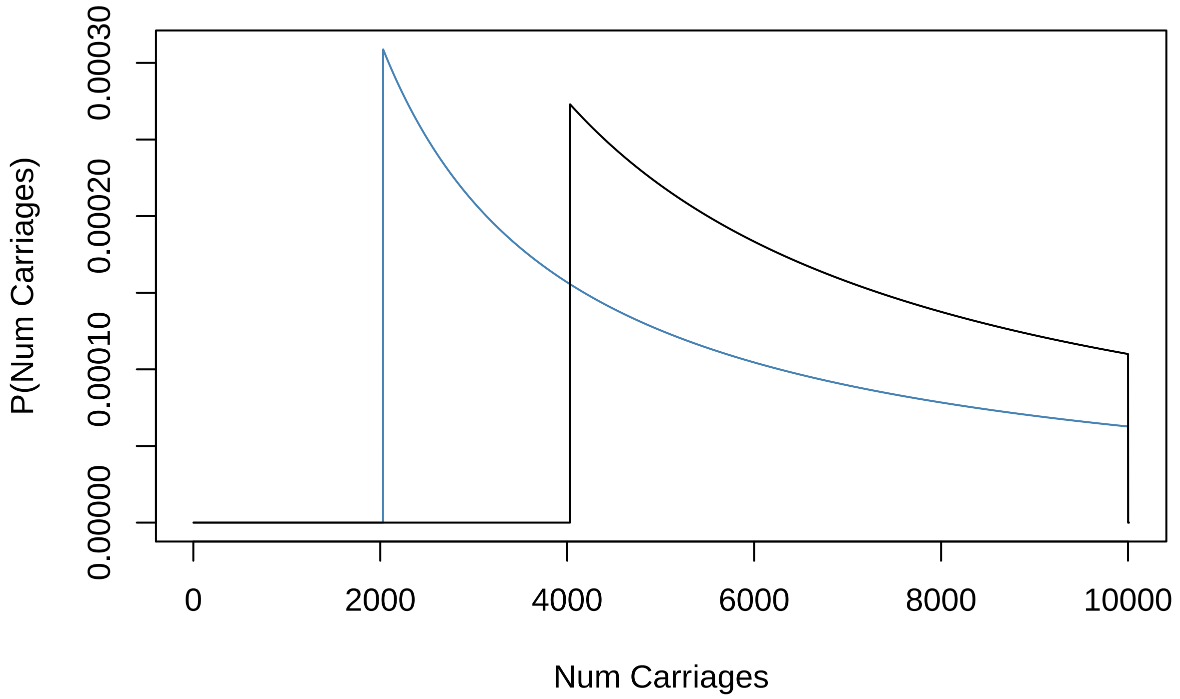
```
## [1] 6349
```

Hence we get median - 6349 < mean - 6569.502. This is expected due to the left skew of the distribution

- Now you've seen a carriage numbered 2031. Make use of Bayesian Updating to get a new posterior (this can be done by multiplying the two posteriors element wise, taking care to normalise).

First let's plot both

```
post2 = cityrail(2031, maxn)

plot(numc, post2, "l", col="steelblue",
     ylab="P(Num Carriages)", xlab="Num Carriages")
lines(numc, post1)
```

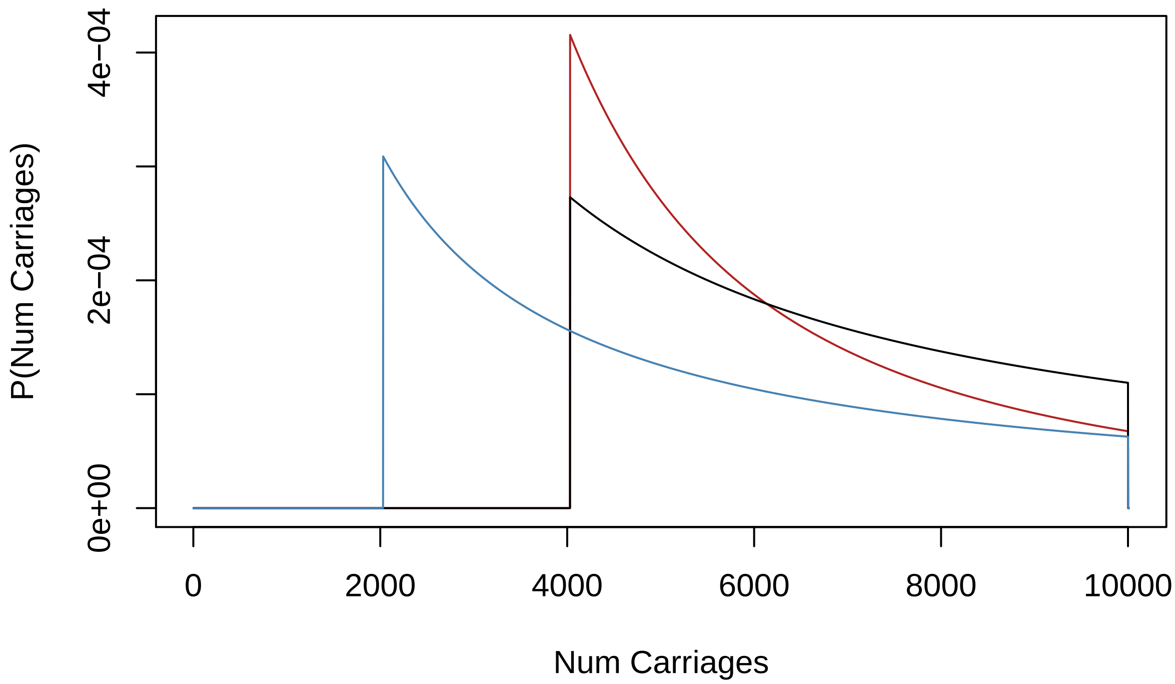


As expected the 2031 is further to the left and more probability to the left. Intuitively, if there are only 2031 carriages, there's a high probability of having seen 2031 than if there were 10000.

```
multiplypost = function(first, second){
  # assume they're already normalised
  newpost = first * second
  newpost / sum(newpost)
}
outpost = multiplypost(post1, post2)
```

I.e. since each event is independent, bayesian updating is a simple multiplcation (this doesn't hold as obviously for continuous distributions)

```
plot(numc, outpost, "l", col="firebrick",
     ylab="P(Num Carriages)", xlab="Num Carriages")
lines(numc, post1)
lines(numc, post2, col="steelblue")
```



Red is the posterior. We can see that with observations 2031 and 4031, we know that there must be at least 4031 carriages. Note how our posterior of 4031 vs combined has shifted to the left as seeing another small number has made it more likely to have fewer carriages

```
c(sum(numc * outpost), medianp(outpost))
```

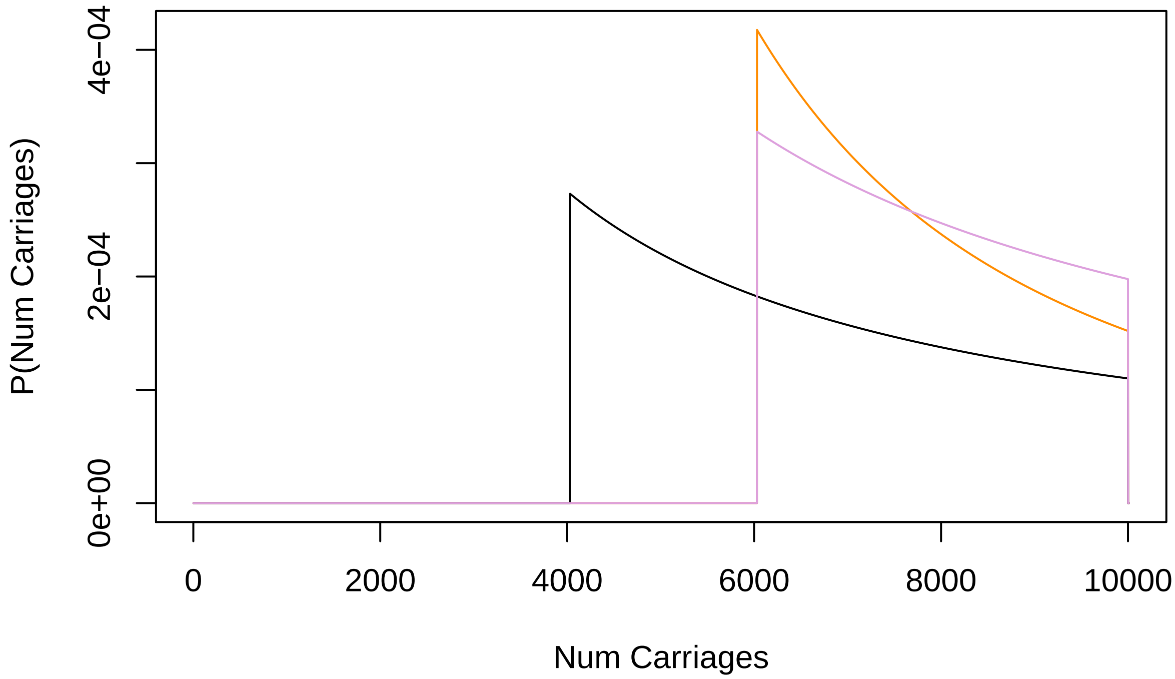
```
## [1] 6135.475 5745.000
```

Now the difference between median and mean has increased significantly

- What if the carriage number was 6031 instead? Report median and mean.

Similarly

```
post3 = cityrail(6031, maxn)
outpost6031 = multiplypost(post1, post3)

plot(numc, outpost6031, "l", col="dark orange",
     ylab="P(Num Carriages)", xlab="Num Carriages")
lines(numc, post1)
lines(numc, post3, col="plum")
```



And we see a similar situation. orange is the final posterior

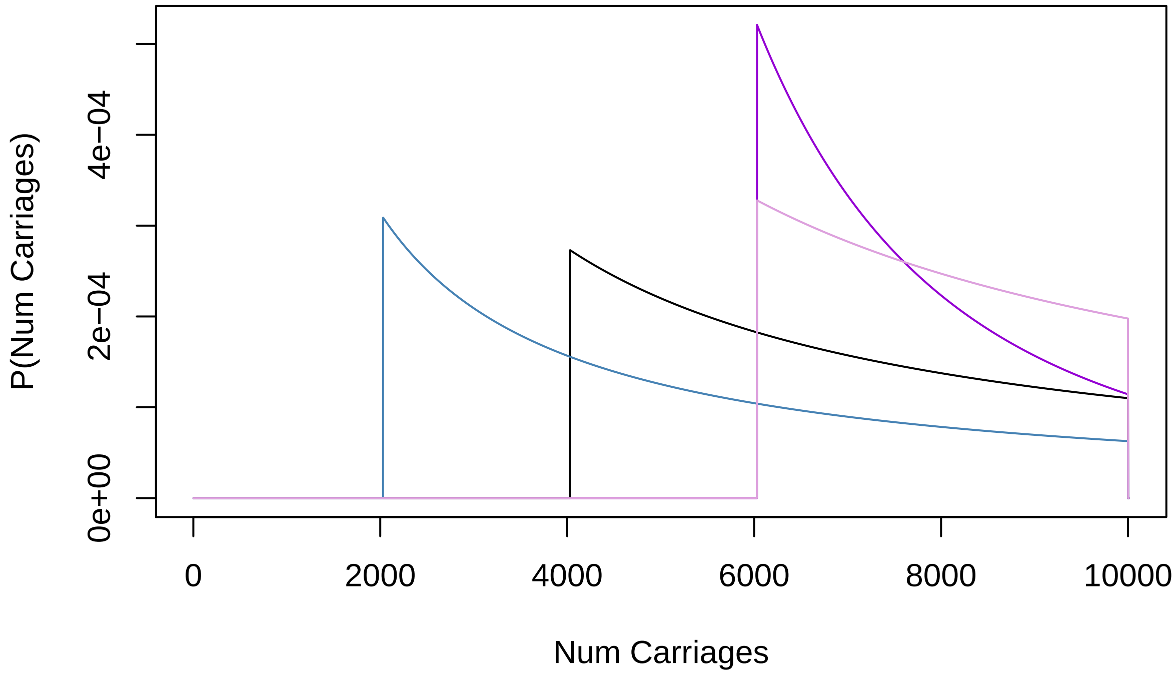
```
c(sum(numc * outpost6031), medianp(outpost6031))
```

```
## [1] 7683.654 7524.000
```

We see here since we're squeezing the posterior into a smaller area, the median and mean are less far apart despite similar shapes.

- Now with all 3 carriage observations, what's your estimate of the number of carriages in Sydney

```
# we can repeatedly multiply posteriors via bayesian updating
outpostAll = multiplypost(outpost, post3)


plot(numc, outpostAll, "l", col="darkviolet",
     ylab="P(Num Carriages)", xlab="Num Carriages")
lines(numc, post1)
lines(numc, post2, col="steelblue")
lines(numc, post3, col="plum")
```



we can see we've become much surer about the number of carriages (violet line)

Estimates:

```
c(sum(numc * outpostAll), medianp(outpostAll))
```

```
## [1] 7523.924 7303.000
```

- (Bonus) Suggest a method that would work if you didn't know what the upper bound on the number of trains would be. (Hint the right prior is needed)

For those of you who remeber your maths, the sequence 1/x for x=1..N doesn't actually converge.

```
sum(1/(1:1000))
```

```
## [1] 7.485471
```

```
sum(1/(1:10000))
```

```
## [1] 9.787606
```

```
sum(1/(1:1000000))
```

```
## [1] 14.39273
```

This means that normalisation is impossible!

This is not the case for something like 1/N^2

```
sum(1/(1:10000)^2)
```

```
## [1] 1.644834
```

```r
sum(1/(1:1000000)^2)
```

```
## [1] 1.644933
```

In this case we can use a prior to save us! Choosing anything that converges, like P(N) = .999^N or P(N)= 1/N will give a posterior that can be normalised.

The code above doesn't assume infinite support, so I leave this as an exercise for the reader