CoinToss

Manpreet S. Katari (with a few extra annotations from Kris)

Fall 2019

Probability

Coin Toss

Here's a simple simulation where we will try to empirically prove that the more trials we have, the closer we will get to the real mean.

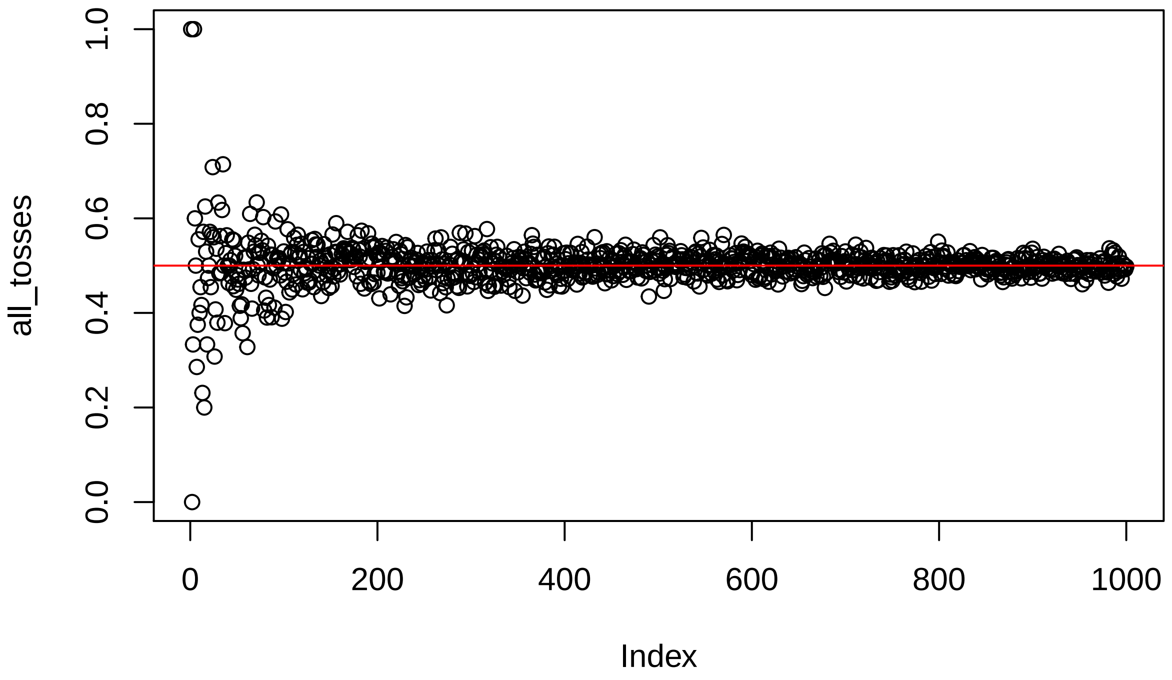
Framework:

- A single coin toss is a single Bernoulli trial for a random variable with a binary outcome, $x = 0, 1$. We define 1 as a "success", and 0 as a "failure".
- We also define a certain probability of success, p. For a fair coin, $p = 0.5$.
- The Binomial distribution comprises a set of Bernoulli trials. Here we will sample from a Binomial distribution and find the total number of successes out of many independent trials.
- The proportion of successes should converge on the true value of p as the number of samples increases.

Setup:

- Since the coin has only two outcomes (H and T), we will create a variable with two values: **1 for H** and **0 for T**.
- Then we will use the `sample()` function 1000 times, increasing the sample size from 1 to 1000. At each iteration, we will keep track of the proportion of heads we get.
- Finally, we will plot the cumulative average as the total number of trials increases.

```r
toss = c(1,0) # a single Bernoulli trial

# generate samples
all_tosses = numeric()
for ( i in 1:1000 ) { # gradually increase the number of coin flips from 1 to 1000

  sample_toss = sample(toss, i, replace = T)
  all_tosses[i] = mean(sample_toss) # the proportion of successes in i coin flips
}

# plot the results and draw a red line across 0.5

plot(all_tosses)
abline(h=0.5, col="red")
```

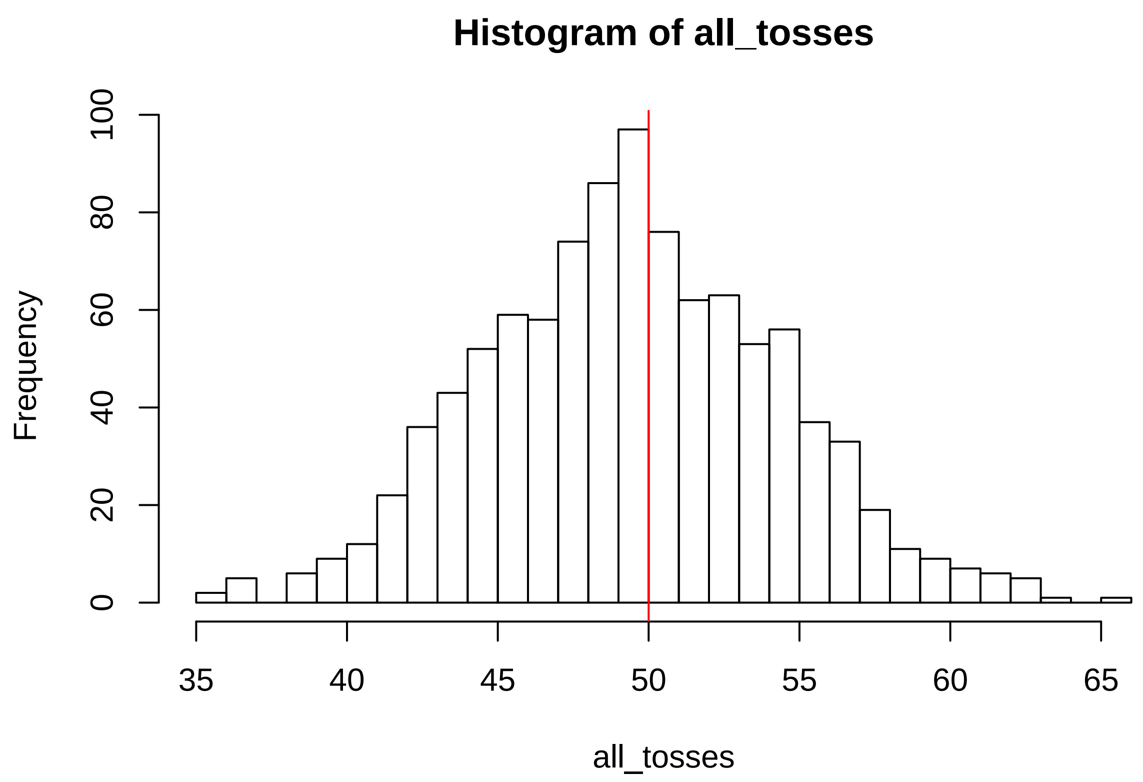


This graph demonstrates that there is much more variability and uncertainty when there are smaller number of events. As the number of events increases, the mean outcome (proportion of heads) gets closer to the true probability, $p = 0.5$.

Now let's change the question - instead of asking about the proportion of times we get heads as we increase the number of coin flips, let's ask,

- **How many Heads will we get if we flip the coin 100 times?**
- We will do this 1000 times and look at the distribution of values we get using a histogram.

```
set.seed(0) # this will make sure everyone's sample() function
            # gives same results.

all_tosses = numeric()
num_flips=100 # number of coin flips per iteration
              # (sample size = 100)

for ( i in 1:1000 ) { # repeat the sampling 1000 times
                      # (1000 random samples of 100 flips)

  # since heads are 1, we can count heads by simply
  # summing up the total number of 1's.
  sample_toss = sample(toss, num_flips, replace = T)
  all_tosses[i] = sum(sample_toss)

}

hist(all_tosses, breaks=25)
abline(v=50, col="red")
```



As expected, we see that values close to 50 are the most frequent.

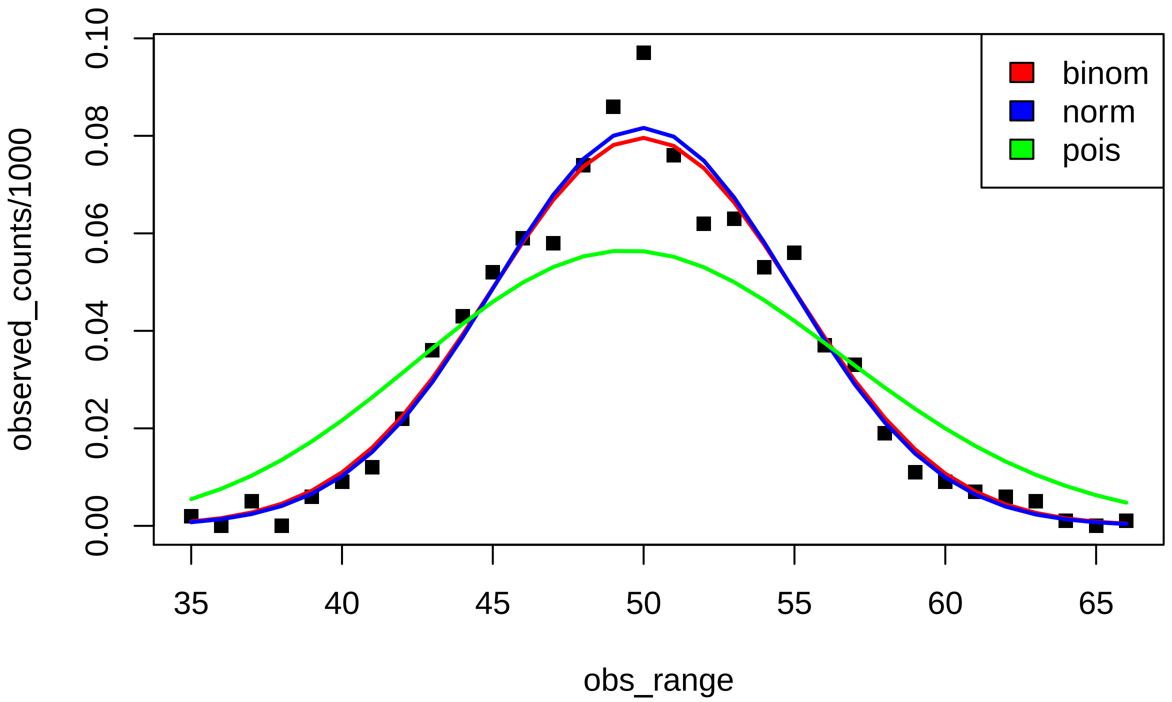
Let's draw a theoretical line and compare it to our data. Since this is a binomial distribution, we can use one of the helper functions to calculate the theoretical values of the line.

```
observed_counts = numeric()
obs_range=range(all_tosses)[1]:range(all_tosses)[2]

for (i in 1:length(obs_range)) {
  observed_counts[i] = sum(all_tosses==obs_range[i])
}
observed_counts
```

```
##  [1]  2  0  5  0  6  9 12 22 36 43 52 59 58 74 86 97 76 62 63 53 56 37 33
## [24] 19 11  9  7  6  5  1  0  1
```

```
# Compare with theoretical curves fo binomial, normal, and Poisson distributions
# Binomial(x, n, p)
theor_binom = dbinom(obs_range, num_flips,
                     mean(all_tosses)/num_flips)
# Normal(x,mean,sd)
theor_norm = dnorm(obs_range,mean=mean(all_tosses),
                   sd=sd(all_tosses))
# Poisson(x,lambda) -- only one parameter! mean = sd
theor_pois = dpois(obs_range, lambda = mean(all_tosses))
```

```
plot(obs_range, observed_counts/1000,
     type="p", col="black", pch=15)

lines(obs_range, theor_binom, type="l", col="red", lwd=2)
lines(obs_range, theor_norm, type="l", col="blue", lwd=2)
lines(obs_range, theor_pois, type="l", col="green", lwd=2)
legend("topright",legend = c("binom","norm","pois"),
       fill=c("red","blue","green"))
```

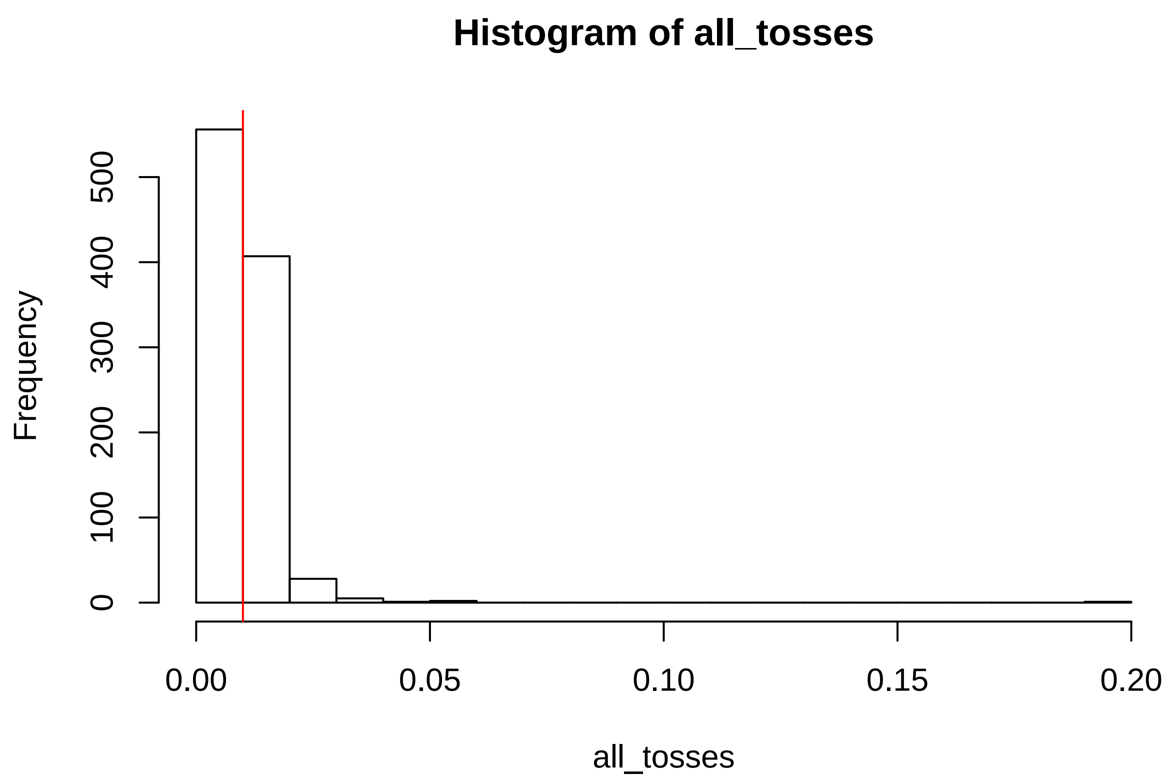


Here, the black and red lines nearly overlap. Thus, the binomial looks very similar to the normal distribution. The Poisson distribution doesn't fit the data as well.

What if the probability is very low?

Let's repeat the same exercise as above, but changing p to a very low value.

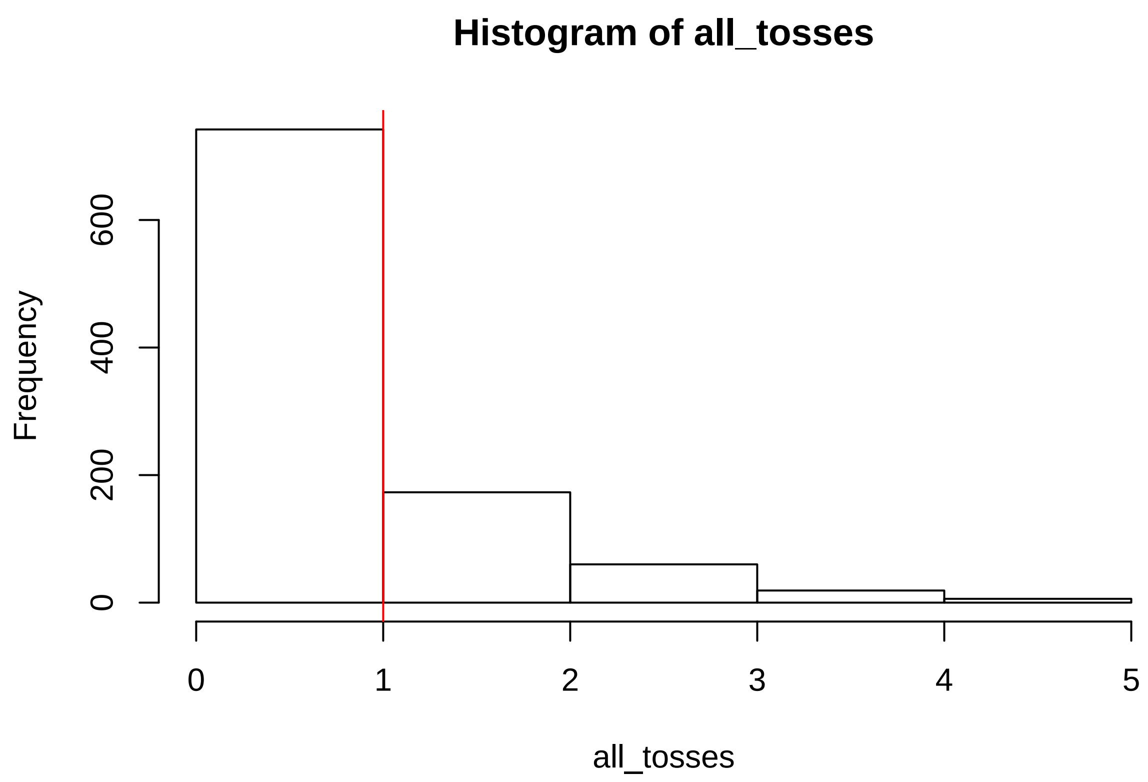
```
toss = c(1,rep(0,99)) # p = 0.01 (1 H, 99 T per 100)

all_tosses = numeric()
for ( i in 1:1000) {
  sample_toss = sample(toss,i, replace = T)
  all_tosses[i] = mean(sample_toss)
}

hist(all_tosses, breaks=25)
abline(v=.01, col="red")
```



Again, we see that there is much more variability and uncertainty when the number of events is smaller. As the number of events increases, the mean of the observations converges on the true mean.

Now let's change the question again and ask, **How many Heads will we get if we flip the coin 100 times?** As before, we will do this 1000 times and look at the distribution of values we get using a histogram.

```r
all_tosses = numeric()
num_flips=100 # make 100 coin flips per sample
for ( i in 1:1000) { # 1000 samples of 100 coin flips each

  # since heads are 1, we can simply sum up the 1's
  sample_toss = sample(toss,num_flips, replace = T)
  all_tosses[i] = sum(sample_toss)

}

hist(all_tosses, breaks=5)
abline(v=1, col="red")
```



Let's draw a theoretical line and compare it to our data. Since this is a binomial distribution, we can use one of the helper functions to calculate the theoretical values of the line.

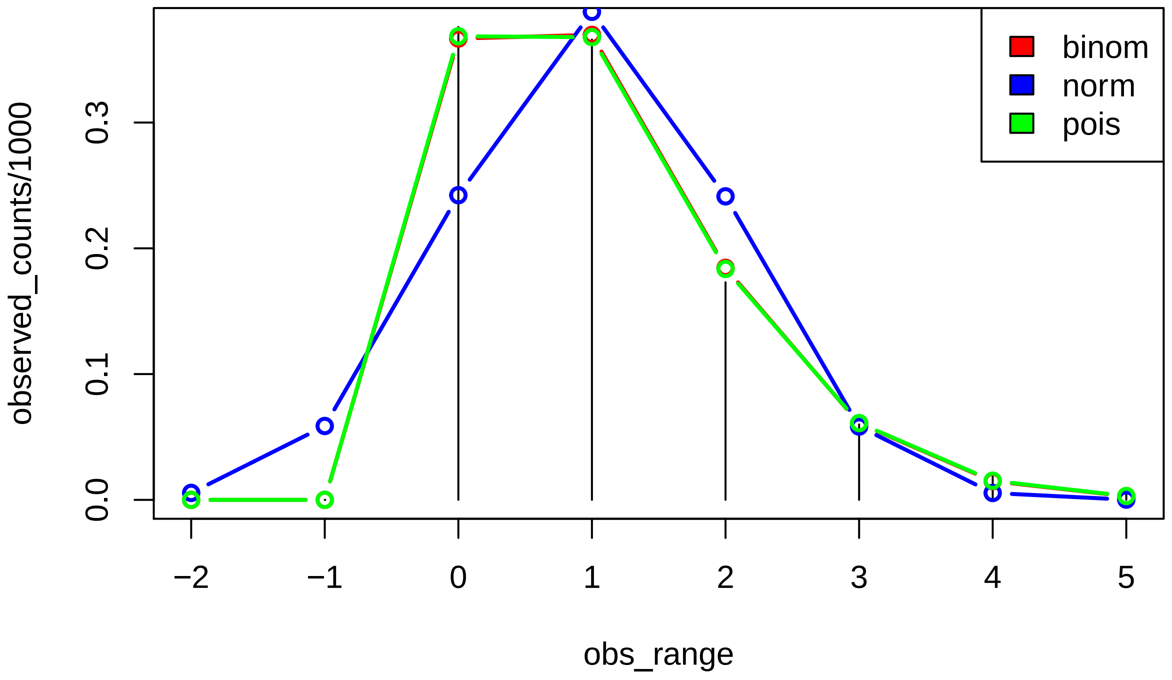
```
observed_counts = numeric()
#obs_range=range(all_tosses)[1]:range(all_tosses)[2]

obs_range=-2:range(all_tosses)[2]


for (i in 1:length(obs_range)) {
  observed_counts[i] = sum(all_tosses==obs_range[i])
}
observed_counts
```

```
## [1]   0   0 376 366 173  60  19   6
```

```
# Compare with theoretical curves fo binomial, normal, and Poisson distributions
# Binomial(x, n, p)
theor_binom = dbinom(obs_range, num_flips,
                     mean(all_tosses)/num_flips)
# Normal(x,mean,sd)
theor_norm = dnorm(obs_range,mean=mean(all_tosses),
                   sd=sd(all_tosses))
# Poisson(x,lambda) -- only one parameter! mean = sd
theor_pois = dpois(obs_range, lambda = mean(all_tosses))
```

```
plot(obs_range, observed_counts/1000,
     type="h", col="black", pch=15)

lines(obs_range,theor_binom, type="b", col="red", lwd=2)
lines(obs_range, theor_norm, type="b", col="blue", lwd=2)
```

```r
lines(obs_range, theor_pois, type="b", col="green", lwd=2)
legend("topright",legend = c("binom","norm","pois"),
       fill=c("red","blue","green"))
```



Here, we see that when the probability is small, the Poisson is a better fit to the data. Does it make sense to have a negative value in this case?