Unit01

K Iwasaki

September 3, 2017

Computing Probabilities of Binomial Probability Model

```
# calcualte the probability for w = 1, which is 1 success in 5 trials
dbinom(x = 1, size = 5, prob = .6)
```

```
## [1] 0.0768
```

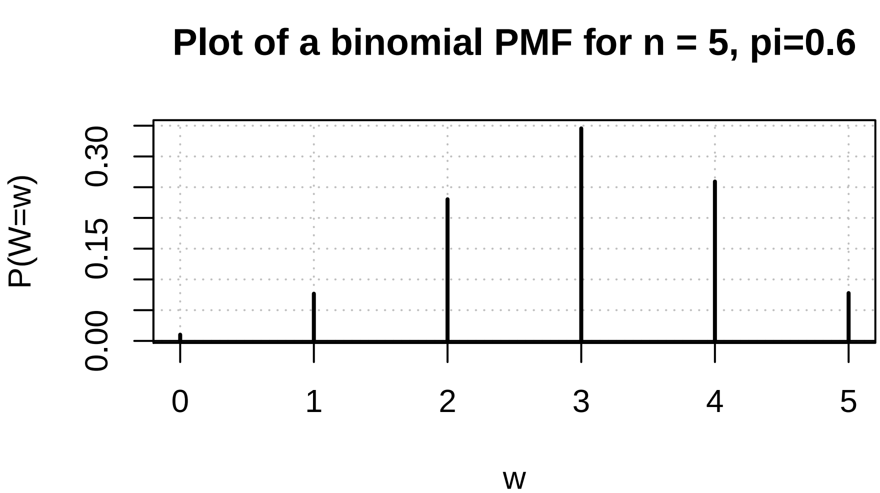
```
# calculate the probabilities for w = 0, 1, 2, 3, 4, 5
dbinom(x = 0:5, size = 5, prob = .6)
```

```
## [1] 0.01024 0.07680 0.23040 0.34560 0.25920 0.07776
```

```
pmf = dbinom(x = 0:5, size = 5, prob = .6)
pmf.df = data.frame(w = 0:5, prob = round(x = pmf, digits = 4))
pmf.df
```

```
##   w   prob
## 1 0 0.0102
## 2 1 0.0768
## 3 2 0.2304
## 4 3 0.3456
## 5 4 0.2592
## 6 5 0.0778
```

```
plot(x = pmf.df$w, y = pmf.df$prob, type = "h", xlab = "w",
     ylab ="P(W=w)", main = "Plot of a binomial PMF for n = 5, pi=0.6",
     panel.first = grid(col="gray", lty = "dotted"),
     lwd = 2)
abline(h=0)
```

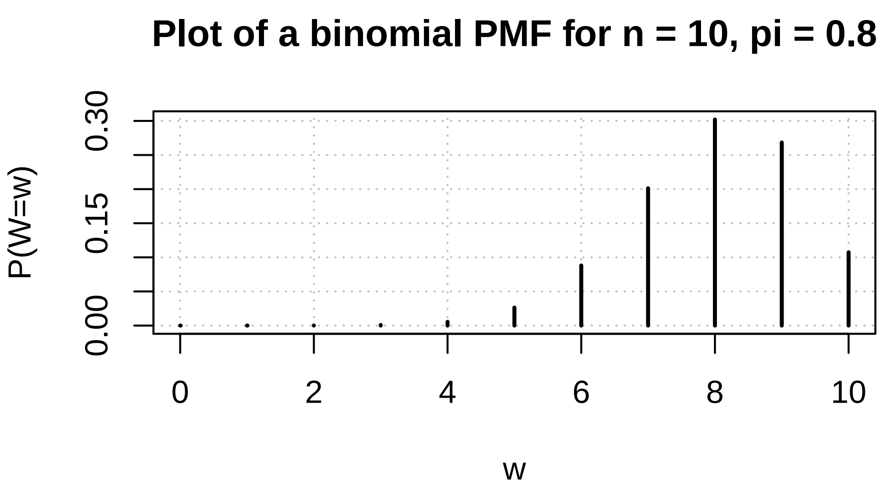


Repeat the implementation in R exercise using pi = 0.2, n = 10. What about pi = 0.8, n = 10? Submit your R script.

```
# pi = 0.8, n = 10
pmf = dbinom(x = 0:10, size = 10, prob = .8)
pmf.df = data.frame(w = 0:10, prob = round(x = pmf, digits = 4))
pmf.df
```

```
##     w   prob
## 1   0 0.0000
## 2   1 0.0000
## 3   2 0.0001
## 4   3 0.0008
## 5   4 0.0055
## 6   5 0.0264
## 7   6 0.0881
## 8   7 0.2013
## 9   8 0.3020
## 10  9 0.2684
## 11 10 0.1074
```

```
plot(x = pmf.df$w, y = pmf.df$prob, type = "h", xlab = "w", ylab = "P(W=w)",
     main = "Plot of a binomial PMF for n = 10, pi = 0.8",
     panel.first = grid(col = "gray", lty = "dotted"), lwd = 2
     )
```



Simulating a Binomial Probability Model

```
set.seed(4848)
bin5 = rbinom(n = 1000, size = 5, prob = 0.6)
bin5[1:20]
```

```
##  [1] 3 2 4 1 3 1 3 3 3 4 3 3 3 2 3 1 2 2 5 2
```

```r
mean(bin5)
```

```
## [1] 2.991
```

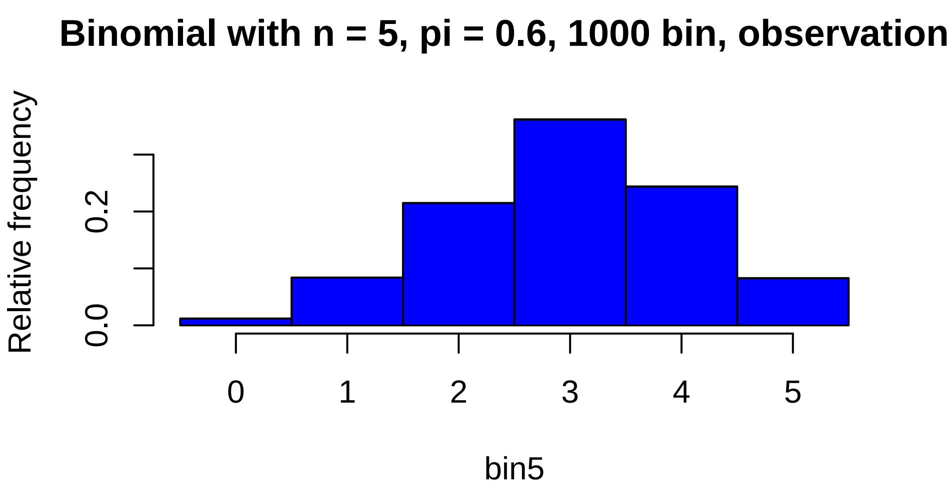
```r
var(bin5)
```

```
## [1] 1.236155
```

```r
table(x = bin5)
```

```
## x
##   0   1   2   3   4   5
##  12  84 215 362 244  83
```

```r
hist(x = bin5, main = "Binomial with n = 5, pi = 0.6, 1000 bin, observations",
     col = "blue", probability = TRUE, breaks = -0.5:5.5, ylab = "Relative frequency")
```



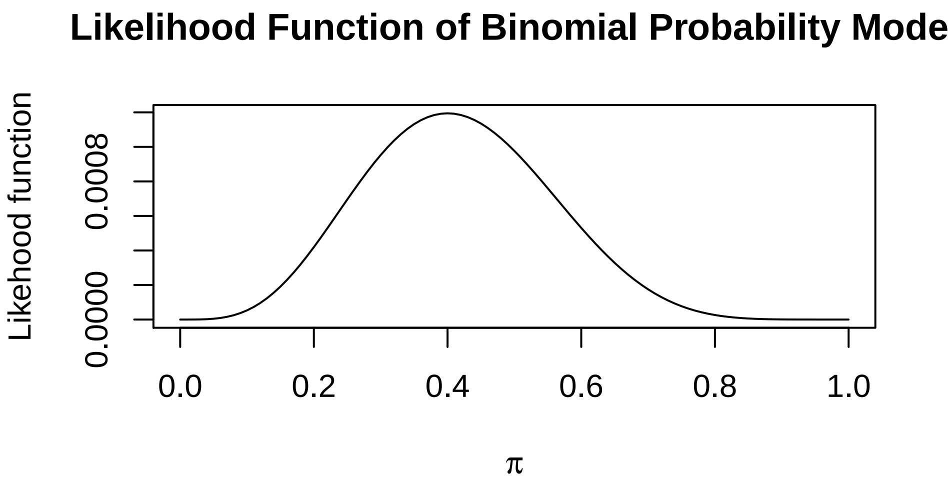
Maximum Likelihood Estimation

Suppose w = 4 and n = 10. Given this observed information, we would like to find the corresponding parameter value for pi that produces the largest probability of obtaining this particular sample.

```r
sum.y = 4
n = 10
# try different value of pi
pi = c(0.2, 0.3, 0.35, 0.39, 0.4, 0.41, 0.5)
Lik = pi^sum.y * (1-pi)^(n-sum.y)
data.frame(pi, Lik)
```

```
##     pi          Lik
## 1 0.20 0.0004194304
## 2 0.30 0.0009529569
## 3 0.35 0.0011317547
## 4 0.39 0.0011918935
## 5 0.40 0.0011943936
## 6 0.41 0.0011919211
## 7 0.50 0.0009765625
```

```r
# likelihood funciton plot
curve(expr = x^sum.y*(1-x)^(n-sum.y), xlim = c(0,1),
      xlab = expression(pi), ylab = "Likehood function",
      main = "Likelihood Function of Binomial Probability Model")
```



Note that pi = 0.4 is the most plausible value of pi for the observed data because this maximizes the likelihood function. Therefore, 0.4 is the maximum likelihood estimate.

Wald Confidence Interval

```r
qnorm(p = 1 - 0.05/2, mean = 0, sd = 1)
```

```
## [1] 1.959964
```

```r
qnorm(p = 0.05/2, mean = 0, sd = 1)
```

```
## [1] -1.959964
```

Wald Confidence Interval

```r
w = 4
n = 10
alpha = 0.05
pi.hat = w / n

var.wald = pi.hat * (1 - pi.hat) / n
lower = pi.hat - qnorm(p = 1-alpha/2) * sqrt(var.wald)
upper = pi.hat + qnorm(p = 1-alpha/2) * sqrt(var.wald)
round(data.frame(lower, upper), 4)
```

```
##    lower  upper
## 1 0.0964 0.7036
```

```r
# quicker
round(pi.hat + qnorm(p = c(alpha/2, 1- alpha/2)) * sqrt(var.wald), 4)
```

```
## [1] 0.0964 0.7036
```

Calculate the True Confidence or Coverage (textbook page 20)

suppose that n = 40, pi = 0.157, and alpha = 0.05

1. Simulate 1,000 samples using the rbinom() function with n = 40 and pi = 0.157
2. Caclualte the 95% Wald confidence interval for each sample, and
3. Calculate the proportion of intervals that contain pi = 0.157; this is the estimated true confidence interval

```r
numb.bin.samples = 1000 # try 1000 times
pi = 0.157
alpha = 0.05
n = 40
set.seed(4516)
w = rbinom(n = numb.bin.samples, size = n, prob = pi)
pi.hat = w/n
var.wald = pi.hat*(1 - pi.hat)/n
lower = pi.hat - qnorm(p = 1 - alpha/2) * sqrt(var.wald)
upper = pi.hat + qnorm(p = 1 - alpha/2) * sqrt(var.wald)
data.frame(w, pi.hat, lower, upper)[1:10, ]
```

```
##    w pi.hat        lower     upper
## 1  6  0.150  0.039344453 0.2606555
## 2  6  0.150  0.039344453 0.2606555
## 3  7  0.175  0.057249138 0.2927509
## 4  8  0.200  0.076040994 0.3239590
## 5  8  0.200  0.076040994 0.3239590
## 6  6  0.150  0.039344453 0.2606555
## 7  8  0.200  0.076040994 0.3239590
## 8  3  0.075 -0.006624323 0.1566243
## 9  5  0.125  0.022511030 0.2274890
## 10 4  0.100  0.007030745 0.1929693
```

```r
save = ifelse(test = pi > lower, yes = ifelse(test = pi < upper, yes = 1, no = 0), no = 0)
save[1:10]
```

```
##  [1] 1 1 1 1 1 1 1 0 1 1
```

```r
mean(save)
```

```
## [1] 0.878
```

In this example, an estimate of the true confidence level is only 0.878 (not 0.95)

Contingency Table and Confidence Interval of Two Binary Variables

```r
c.table = array(data = c(251, 48, 34, 5), dim = c(2,2),
                dimnames = list(First = c("made", "missed"),
```

```
                                  second = c("made", "missed")))
c.table
```

```
##         second
## First    made missed
##   made    251     34
##   missed   48      5
```

```
rowSums(c.table)
```

```
##   made missed
##    285     53
```

```
pi.hat.table = c.table / rowSums(c.table)
pi.hat.table
```

```
##         second
## First         made     missed
##   made   0.8807018 0.11929825
##   missed 0.9056604 0.09433962
```

The estimated probability that Larry Bird makes his second free throw attemp is pi one hat is 0.8808, given that he makes the first, and pi two hat is 0.9057, given he misses the first.

Formulation of Contingency Table and Confidence Interval of Two Binary Variables

```
alpha = 0.05
pi.hat1 = pi.hat.table[1,1]
pi.hat2 = pi.hat.table[2,1]

# Wald CI
var.wald = pi.hat1*(1 - pi.hat1) / sum(c.table[1,]) + pi.hat2*(1-pi.hat2) /sum(c.table[2,])
pi.hat1 - pi.hat2 + qnorm(p = c(alpha/2, 1-alpha/2)) * sqrt(var.wald)
```

```
## [1] -0.11218742  0.06227017
```

```
# Agresti-Caffo CI
pi.tilde1 = (c.table[1,1] + 1) / (sum(c.table[1,]) + 2)
pi.tilde2 = (c.table[2,1] + 1) / (sum(c.table[2,]) + 2)
var.AC = pi.tilde1*(1-pi.tilde1) / sum(c.table[1,] + 2) + pi.tilde2*(1-pi.tilde2) / (sum(c.table[2,] + 2
pi.tilde1 - pi.tilde2 + qnorm(p = c(alpha/2, 1- alpha/2)) * sqrt(var.AC)
```

```
## [1] -0.10215394  0.07643332
```

Testing the difference of two probabilities

```
alpha = 0.05
pi.hat1 = pi.hat.table[1,1]
pi.hat2 = pi.hat.table[2,1]

colSums(c.table)
```

```
##   made missed
##    299     39
```

```r
n1 = rowSums(c.table)[1]
n2 = rowSums(c.table)[2]

pi.bar = colSums(c.table)[1] / sum(colSums(c.table))
pi.bar
```

```
##      made
## 0.8846154
```

```r
z = (pi.hat1 - pi.hat2) / sqrt(pi.bar * (1 - pi.bar) * (1/n1 + 1/n2) )
abs(z)
```

```
##      made
## 0.5222416
```

```r
crit.value = qnorm(p = 1 - alpha/2)
ifelse(abs(z) > crit.value, "reject", "fail to reject")
```

```
##             made
## "fail to reject"
```

```r
prop.test(x = c.table, conf.level = 0.95, correct = FALSE)
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c.table
## X-squared = 0.27274, df = 1, p-value = 0.6015
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.11218742  0.06227017
## sample estimates:
##    prop 1    prop 2
## 0.8807018 0.9056604
```