Lab 6

Solutions

3/4/2020

Binomial Distribution

Conditions

1. Independent trials
2. Each outcome can be classified as a success or failure.
3. Fixed number of trials, n
4. Same probability of success, p, for each trial

R Code

- $P(X = k)$ =dbinom(x=k, size=n, prob=p)
- $P(X \leq k)$ =pbinom(q=k, size=n, prob=p, lower.tail=TRUE)
- $P(X > k)$ =pbinom(q=k, size=n, prob=p, lower.tail=FALSE)
- rbinom(n=number of simulations, size=n, prob=p)

Problem description

Let's suppose you want to play a card game. This game will be played with a standard deck of 52 cards: 13 hearts, 13 diamonds, 13 clubs, and 13 spades. Each time you play this game, you start with a well-shuffled deck, cards facedown, and you guess the suit of the card on top. You pay in $1 to play each time. If you guess correctly, you win $3, thereby netting $2. Otherwise, you lose the dollar you paid to play the game. Suppose you play the game 10 times.

Consider the random variable X = number of wins. Check the four conditions to confirm that this is a binomial process.

1. Independent trials: each game is independent
2. Each outcome classified as a success or failure: win=success, loss=failure
3. Fixed number of trials, n: play the game 10 times
4. Same probability of success, p, for each trial: p=18/38

Run the following code chunk to simulate one realization of this process (one set of 10 games). We will save it as s1 so we can reference it in text (see below code chunk). This will ensure that your random output matches your description when you knit!

```r
s1 <- data.frame(wins=rbinom(n=1, size=10, prob=1/4))
```

In this process, we won 2 out of 10 games.

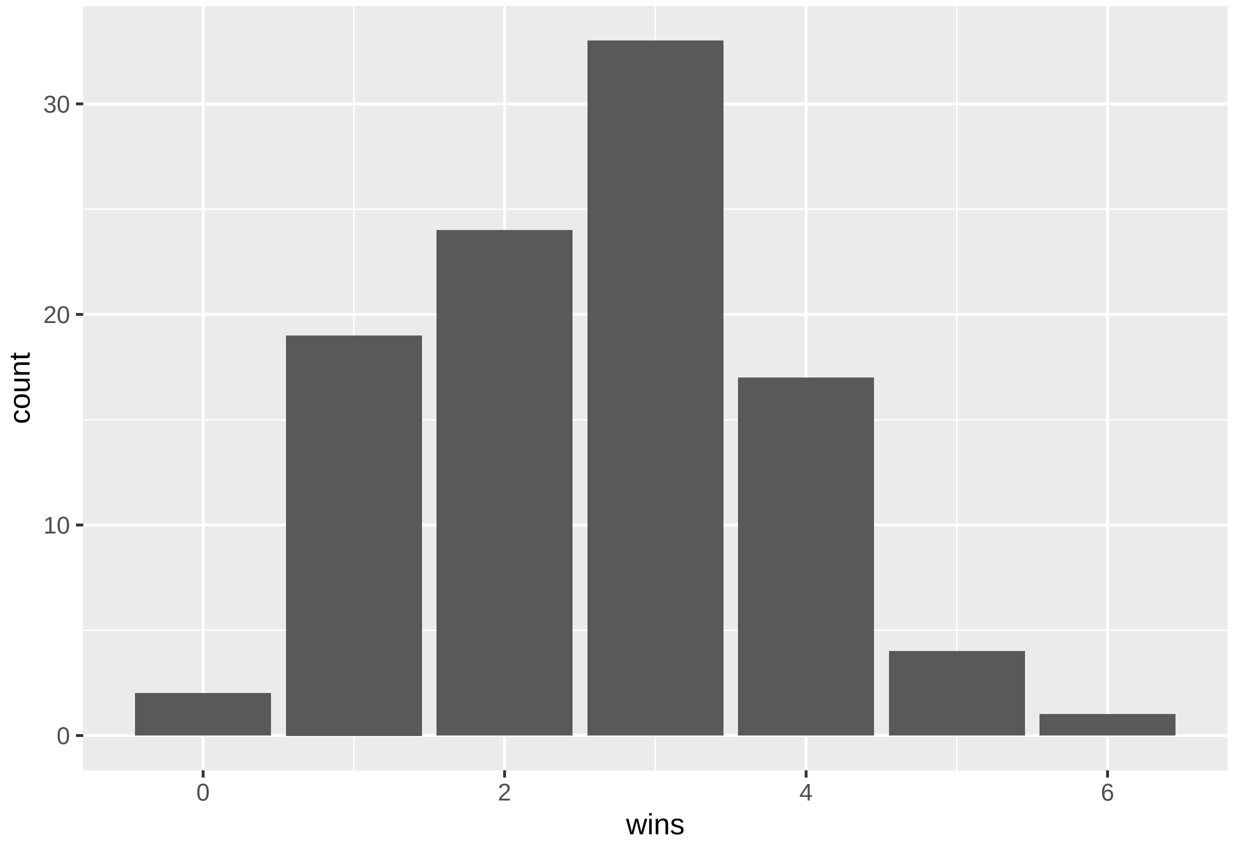
Now, amend the code from the previous chunk to simulate 100 realizations of this process. Save this as s100.

```r
s100 <- data.frame(wins=rbinom(n=100, size=10, prob=1/4))
```

Make a barplot of the wins from s100. Use the ggplot2 package.

```r
library(ggplot2)

ggplot(data=s100, aes(x=wins)) + geom_bar()
```



Summarize the counts from s100 using table(). If you don't remember how to use table(), look at the help file using ?table.

```r
table(s100)
```

```
## s100
##  0  1  2  3  4  5  6
##  2 19 24 33 17  4  1
```

Use dbinom() to find the following 11 probabilities:

```r
## P(X=0)
p0 <- dbinom(x=0, size=10, prob=1/4)
p0
```

```
## [1] 0.05631351
```

```
## P(X=1)
p1 <- dbinom(x=1, size=10, prob=1/4)
p1
```

```
## [1] 0.1877117
```

```
## P(X=2)
p2 <- dbinom(x=2, size=10, prob=1/4)
p2
```

```
## [1] 0.2815676
```

```
## P(X=3)
p3 <- dbinom(x=3, size=10, prob=1/4)
p3
```

```
## [1] 0.2502823
```

```
## P(X=4)
p4 <- dbinom(x=4, size=10, prob=1/4)
p4
```

```
## [1] 0.145998
```

```
## P(X=5)
p5 <- dbinom(x=5, size=10, prob=1/4)
p5
```

```
## [1] 0.0583992
```

```
## P(X=6)
p6 <- dbinom(x=6, size=10, prob=1/4)
p6
```

```
## [1] 0.016222
```

```
## P(X=7)
p7 <- dbinom(x=7, size=10, prob=1/4)
p7
```

```
## [1] 0.003089905
```

```
## P(X=8)
p8 <- dbinom(x=8, size=10, prob=1/4)
p8
```

```
## [1] 0.0003862381
## P(X=9)
p9 <- dbinom(x=9, size=10, prob=1/4)
p9
```

```
## [1] 2.861023e-05
```

```
## P(X=10)
p10 <- dbinom(x=10, size=10, prob=1/4)
p10
```

```
## [1] 9.536743e-07
```

How do these exact probabilities compare to the probabilities from s100?

```
comp_df <- data.frame(theoretical_prob=round(c(p0,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10),2),
                      empirical_prob=c(as.vector(table(s100))/100,rep(0,11-dim(table(s100)))))
comp_df
```

```
##    theoretical_prob empirical_prob
## 1              0.06           0.02
## 2              0.19           0.19
## 3              0.28           0.24
## 4              0.25           0.33
## 5              0.15           0.17
## 6              0.06           0.04
## 7              0.02           0.01
## 8              0.00           0.00
## 9              0.00           0.00
## 10             0.00           0.00
## 11             0.00           0.00
```

The theoretical probabilities (from the dbinom calculations) are very close to those calculated from the simulated data (empirical probabilities).

Use pbinom() to find the following probabilities:

```
## Probability that you win at least once: P(X >= 1)
## P(X >= 1) = P(X > 0)
p_ge1 <- pbinom(q=0, size=10, prob=1/4, lower.tail=FALSE)


## Probability that you win no more than twice: P(X <= 2)
p_le2 <- pbinom(q=2, size=10, prob=1/4, lower.tail=TRUE)


## Probability that you win less than two times: P(X < 2)
## P(X < 2) = P(X <= 1)
p_le1 <- pbinom(q=1, size=10, prob=1/4, lower.tail=TRUE)
```

```
## Probability that you win a majority of the games: P(X >= 5)
## P(X >= 5) = P(X > 4)
p_ge5 <- pbinom(q=4, size=10, prob=1/4, lower.tail=FALSE)
```

Note, if you are unsure of any of your pbinom() code, you can check it by taking sums of the appropriate probabilities from dbinom() that you calculated previously.

How do these exact probabilities compare to the probabilities from s100?

```
comp_df2 <- data.frame(theoretical_prob=round(c(p_ge1, p_le2, p_le1, p_ge5),2),
                       empirical_prob=c(1-comp_df$empirical_prob[1],
                                        sum(comp_df$empirical_prob[1:3]),
                                        sum(comp_df$empirical_prob[1:2]),
                                        sum(comp_df$empirical_prob[6:11])))
comp_df2
```

```
##   theoretical_prob empirical_prob
## 1             0.94           0.98
## 2             0.53           0.45
## 3             0.24           0.21
## 4             0.08           0.05
```

These are pretty close to the same.

What is the expected number of wins? Calculate this in two ways, and compare them.

```
## Mean of s100:
mean(s100$wins)
```

```
## [1] 2.6
```

```
## Using the binomial mean: n*p
10*0.25
```

```
## [1] 2.5
```

These are very close, which we expect.

If you were to increase the number of simulations, would you expect your probabilities from your simulation to get closer to the probabilities you calculated using dbinom() and pbinom()? Why or why not?

If we had a larger number of simulations (e.g. 1000), these empirical probabilities (from the table) should get even closer to the theoretical probabilities (from the formulae) because of the Law of Large Numbers.

Since there is a cost associated with playing this game, as well as a return for winning, we are perhaps more interested in the monetary aspect of this game. Run the following code to include a column in your s100 dataframe that calculates the return for each of the simulations:

```
n <- 10

s100$return <- -1*n + 3*(s100$wins)
```

Calculate the mean and standard deviation of s100$return.

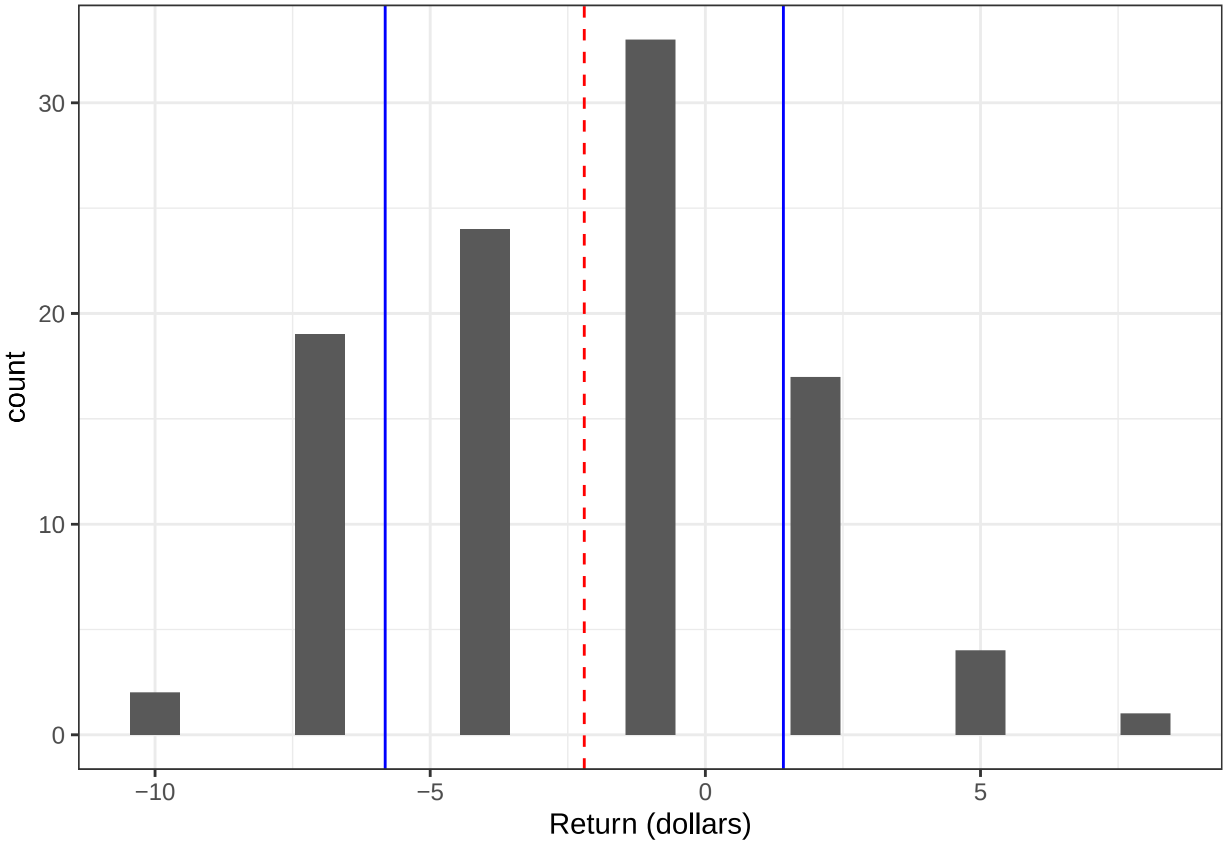
```
mean_return <- mean(s100$return)

sd_return <- sd(s100$return)
```

Based on your calculations, do you think this is a game you should play?

No, this is not a game you should play unless you are trying to lose money. The expected return is -2.2, which is negative. Also, if we examine the barplot of this variable, we see that the distribution is right skewed, and most of the observed returns are negative.

```
ggplot(data=s100, aes(x=return)) +
  geom_bar() +
  theme_bw() +
  geom_vline(xintercept = mean_return, color="red", linetype="dashed") +
  geom_vline(xintercept = mean_return - sd_return, color="blue") +
  geom_vline(xintercept = mean_return + sd_return, color="blue") +
  xlab("Return (dollars)")
```


count
Return (dollars)