Workshop 05: Binomial & Plots

MATH 1051H Instructors

24/09/2018

Some reminders

Anything you write that is outside of an **R chunk** (started and ended by the ticks+{r}) will show up in your final document as text. Try experimenting with using words to help your document flow.

Once we're inside an R chunk, we've learned to create vectors:

```
my_vec1 <- c(1, 2, 3, 4, 5)
my_vec2 <- 1:5
my_vec3 <- c("This", "is", "also", "a", "vector")
```

You can also make vectors by expanding on previously created vectors, using the names as placeholders:

```
my_vec4 <- c("This", "is", "my", "first")
my_vec5 <- c(my_vec4, "vector")
my_vec4
```

```
## [1] "This"  "is"    "my"    "first"
```

```
my_vec5
```

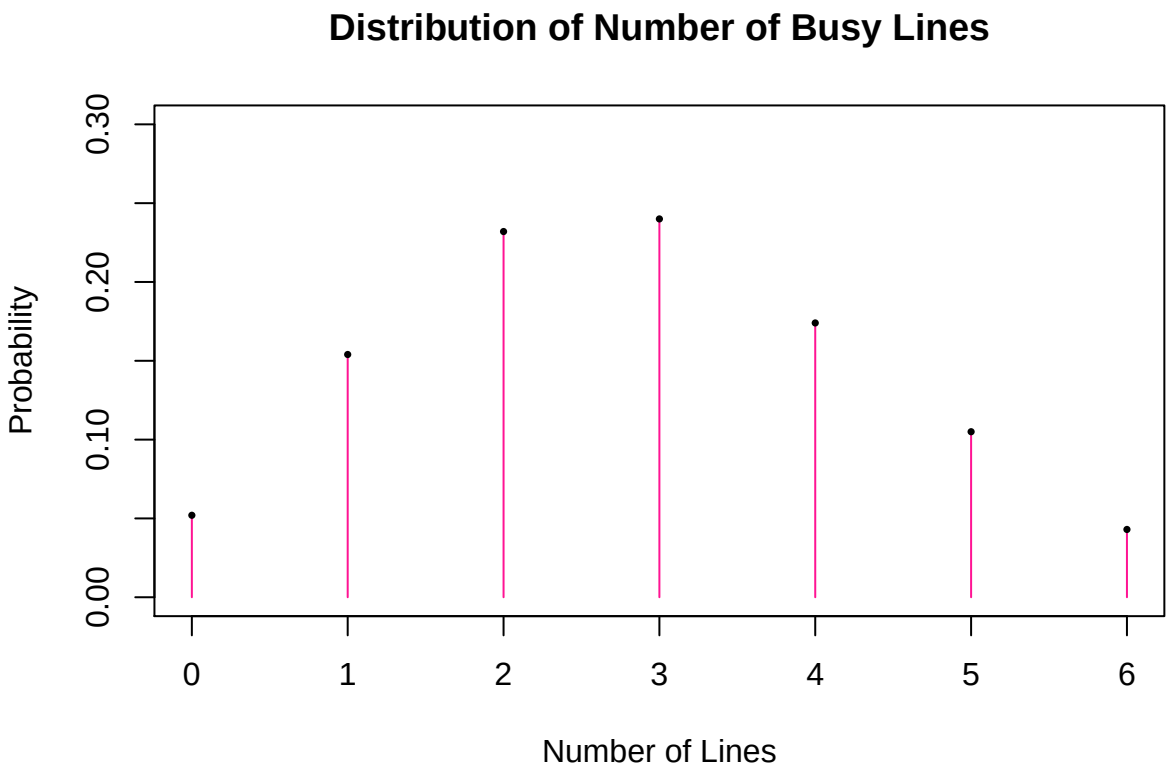
```
## [1] "This"   "is"     "my"     "first"  "vector"
```

Run the above chunk. Do you see what happened when we **concatenated** the vector **my__vec4** and the element "vector"?

Probability Distribution of the Number of Busy Lines

Now, let's consider an example of how many busy telephone lines there are at a local service center, which has six total lines. The probabilities are given, and we're interested in the **distribution**. Let's create a vertical line plot (like bars):

```
x <- c(0:6)
problines <- c(0.052, 0.154, 0.232, 0.240,0.174, 0.105, 0.043)
plot(0:6, problines, type="h", xlim=c(0, 6), ylim=c(0, 0.3),
     main = "Distribution of Number of Busy Lines",
     xlab = "Number of Lines", ylab = "Probability",
     col = "deeppink")
points(0:6, problines, pch=16, cex=.5)
```



Let's experiment with this a little. There was a new argument in this **plot()** call you may not have seen before: **type = "h"**. Try going up into the above code, and changing **h** into **l** (lower-case L). What happens to your plot? This is a **line plot**, also sometimes referred to (incorrectly) as a **scatterplot**. It might be useful for you in your lab courses!

Some Statistics

We can compute statistics on the busy lines, and find $\mathbf{E}[X]$ and $\mathbf{Var}[X]$. Recall that because we are using probabilities, these are not statistics - they're special things called the **expectation** and **variance**. Start with the expectation (mean) of the number of busy lines:

```
x <- c(0:6)
problines <- c(0.052, 0.154, 0.232, 0.240,0.174, 0.105, 0.043)
meanx <- sum(x * problines)
meanx
```

```
## [1] 2.817
```

and then compute the variance:

```
varx <- sum( (x - meanx)^2 * problines )
varx
```

```
## [1] 2.263511
```

```
sqrt(varx)
```

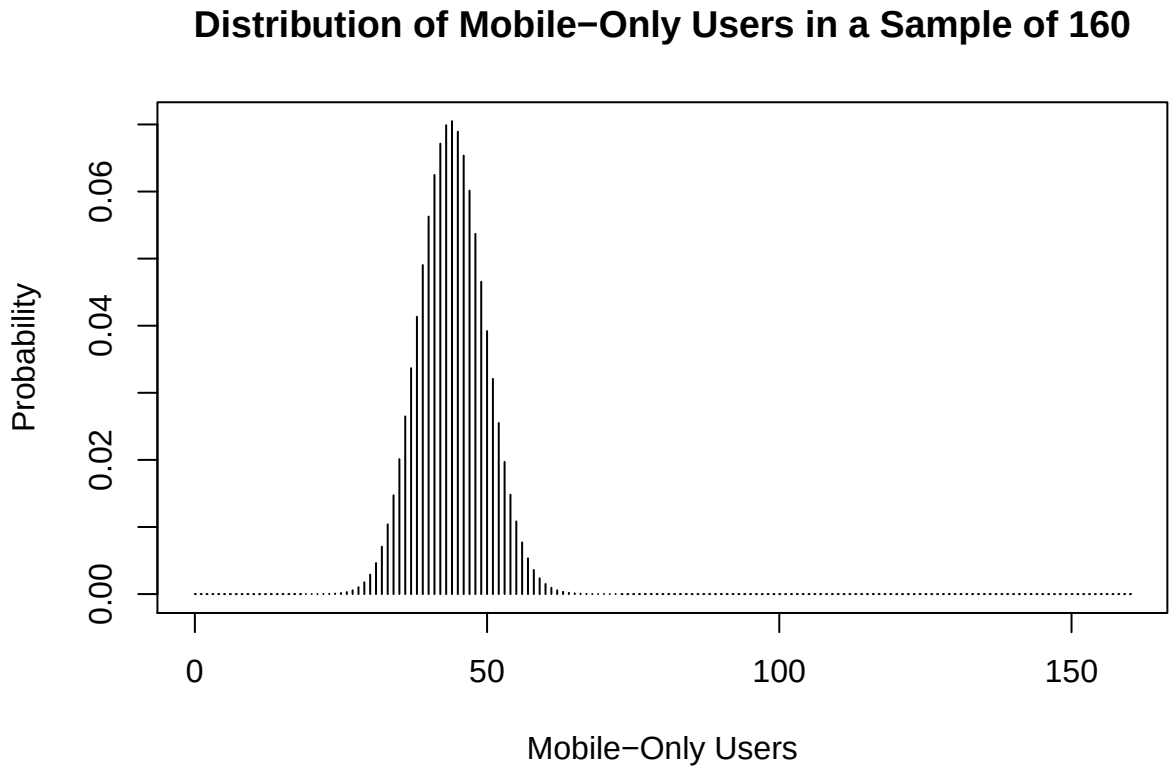
```
## [1] 1.504497
```

Binomial Distribution: Mobile-Only Canadians

Many Canadians no longer have landline telephones, preferring to only have a mobile phone. If we are told that the proportion of Canadians who fall into this category is 27.5%, what is the probability distribution for the number of Canadians who have only cell phones in a sample of size 160?

From recent lectures, we know this is a **binomial** distribution - success being "only has a mobile phone", $p = 0.275$, and $n = 160$. The **distribution** function for this is **dbinom()**, and it has three arguments.

```
x <- 0:160
dist <- dbinom(x, size = 160, prob = 0.275)
plot(x, dist, type="h",
     main = "Distribution of Mobile-Only Users in a Sample of 160",
     xlab = "Mobile-Only Users", ylab = "Probability")
```



So for every number between 0 and 160, we have a **probability** that our sample of 160 Canadians will have that exact number of people who have only a mobile phone. Do you think it is likely that we will find 140 people who only have a mobile phone?

Specific Probability: At Most 20

We can compute specific probabilities on this problem. The probability that **at most 20** will have only a cell phone is one. Note that **at most 20** is equivalent to ≤ 20, which corresponds to the set of outcomes $\{0, 1, 2, \cdots, 19, 20\}$: all of the numbers from 0 to 20. There are two ways we can compute this: find all these numbers, and add them up, or use the **pbinom()** function. We'll show both in the following.

```
our_question <- dist[1:21]
compare <- dbinom(x = 0:20, size = 160, prob = 0.275)
```

```
our_question - compare
```

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

What did we do here? We took the first 21 vector components from **dist** from the earlier chunk, and saved them as **our_question**. We then re-ran the **dbinom()** command and saved it as **compare**. When we take the difference, it turns out these are identical!

Then, what is the probability we want? The sum of all of these!

```
sum(compare)
```

```
## [1] 3.870103e-06
```

```
sum(our_question)
```

```
## [1] 3.870103e-06
```

So this is our probability: basically 0.

The **second way** we can compute this is by using the second function, **pbinom()**. It works a little differently: you specify the **max** or **min** value you want, and then it computes all of the required pieces and sums them up all in one step. Let's do the same question, P[X $\leq$ 20] using this.

```
pbinom(q = 20, size = 160, prob = 0.275, lower.tail = TRUE)
```

```
## [1] 3.870103e-06
```

Exactly the same number, and one less step. Seems handy!

Specific Probability: At Least 50

What if we wanted to ask the question: "What is the probability that **at least** 50 people have only a cell phone?". We can do this too!

Method 1:

```
sum( dbinom(x = 50:160, size = 160, prob = 0.275) )
```

```
## [1] 0.1648542
```

Method 2:

This is a little bit trickier. **pbinom()** is setup to work, by default, on P[X$\leq$x]. If we want to flip it around, we have to be careful, and use the **lower.tail** argument. But there's a catch! If we flip it around, we have to ask "is 50 included or not?".

The probability that P[X $\leq$ 50]:

```
pbinom(q = 50, size = 160, prob = 0.275, lower.tail = TRUE)
```

```
## [1] 0.8743452
```

The probability that P[X $>$ 50]:

```
pbinom(q = 50, size = 160, prob = 0.275, lower.tail = FALSE)
```

```
## [1] 0.1256548
```

But the last one is not exactly what we want, is it? We want P[X $\geq$ 50], **not** P[X $>$ 50]. So we need to subtract 1 from 50, because P[X $>$ 49] is the same thing as P[X $\geq$ 50].

```r
pbinom(q = 49, size = 160, prob = 0.275, lower.tail = FALSE)
```

```
## [1] 0.1648542
```

There we go: now the two methods give the same results!

We'll talk more next week about this "edge" problem, and how it relates to the **normal approximation**. For now, we encourage you to try a few versions of these numbers (20, 50, etc.) and see if you can internalize how to use these functions. Your next assignment will require it!