# Simulation II

W. John Braun, UBC

COSC 405, DATA 405, COSC 505 and DATA 505

## Probability and Discrete Random Variable Simulation

**Discrete Probability Distributions**

**Simulation of Discrete Random Variables**

**Applications: Control Charting, Rain Event Modelling with Poisson Processes**

## Discrete Probability Distributions

**Examples of discrete random variables:**

1. **The number of major earthquakes in a region.**
2. **The number of errors in a software program.**
3. **The number of accidents at a traffic intersection.**
4. **The proportion of mosquitoes killed at a given dose of insecticide.**
5. **The number of attempts made at passing a test until the first success.**

## Discrete Probability Distributions

**Characteristics of discrete random variables:**

1. **Numeric.**
2. **Finite or countably valued.**
3. **Each value is associated with a probability.**

## Discrete Probability Distributions

**Example: Number of heads $H$ obtained in 2 independent coin tosses:**

**Possible values of $H$: $0, 1, 2$.**

**Corresponding probabilities $p_H(h)$ :1/4, 1/2, 1/4.**

**Probability distribution table:**

| $h$ | **0** | **1** | **2** |
|---|---|---|---|
| $p_H(h)$ | **1/4** | **1/2** | **1/4** |

## Discrete Probability Distributions

**Probability distributions can be modelled from data:**

**e.g. For quality purposes, in a refrigerator manufacturing setting, the number of flaws were counted in the surfaces of a sample of 100 refrigerator doors, yielding the following information:**

| Number of Flaws | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Count | 35 | 39 | 12 | 13 | 0 | 0 | 1 |

**The sample can be viewed as an estimate for the total population of refrigerator doors (manufactured by the particular company).**

## Discrete Probability Distributions

**We can divide by the sample size to yield an estimate of the probability distribution for surface flaws $F$:**

| $f$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $p_F(f)$ | 0.35 | 0.39 | 0.12 | 0.13 | 0 | 0 | 0.01 |

**In other words, there is an approximate probability of 0.12 that a randomly selected refrigerator door has exactly 2 surface flaws.**

## Discrete Probability Distributions

**We could create a function in R to return such probabilities:**

```r
dFlaws <- function(x) {
    return(c(.35, .39, .12, .13, 0, 0, .01)[x+1])
}
```

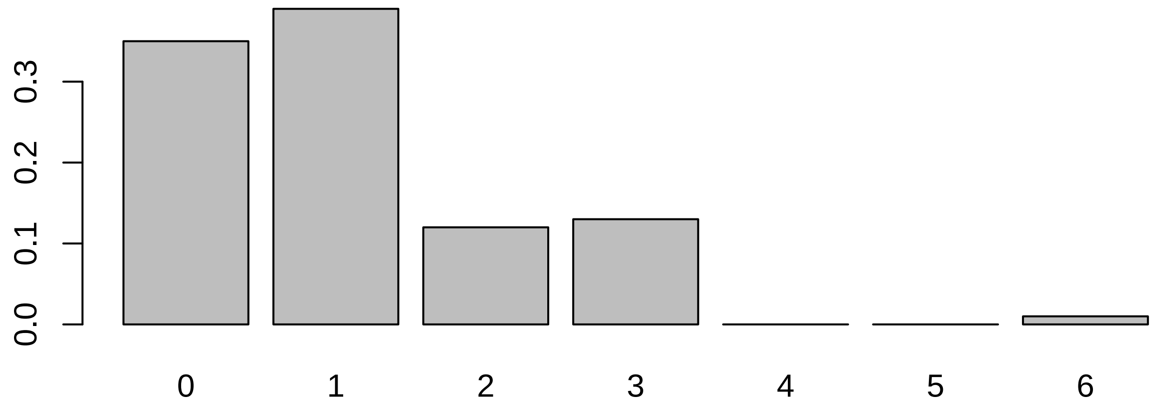
**e.g.** $P(F = 3)$**:**

```r
dFlaws(3)


## [1] 0.13
```

## Visualizing a Discrete Distribution

**The bar plot is an appropriate vehicle to view discrete distributions:**

```
f <- 0:6;  probs <- dFlaws(f); names(probs) <- f
barplot(probs)
```



## Cumulative Distributions

**For a given random variable $X$, we often need to evaluate $P(X \leq x)$ or its complement $P(X > x) = 1 - P(X \leq x)$.**

**The cumulative distribution function is $F(x) = P(X \leq x) = \sum_j^x P(X = j)$.**

## Cumulative Distributions

**For the surface flaws example, we would estimate the cumulative distribution function as:**

```r
pFlaws <- function(x)  {
    return(c(.35, .74,.86, .99, .99, .99, 1)[x+1])
}
```

**e.g.** $P(F > 4)$

```r
1 - pFlaws(4)

## [1] 0.01
```

## Pseudorandom Numbers which follow the Flaw Count Distribution

**The following function shows how one might simulate large numbers of independent variates which follow the distribution of surface flaw counts obtained above:**

```r
rFlaws <- function(n) {
    U <- runif(n)
    X <- numeric(n)
    for (x in 0:6) {
        X[U >= pFlaws(x)] <- x + 1
    }
    return(X)
}
```

## Pseudorandom Numbers which follow the Flaw Count Distribution

**Examples:**

```
rFlaws(10)

##  [1] 0 1 0 1 0 0 3 2 0 1

table(rFlaws(100))

##
##  0  1  2  3
## 39 37 13 11
```

## A Model for the Surface Flaw Distribution

**The completely data-driven model and simulator produced earlier cannot predict counts of 4 or 5. This is not realistic.**

**Models can provide more realism, at the expense of other kinds of inaccuracy.**

**A possible model for the flaw distribution is**

$p(x) = (1 - (x/7)^2)^8/2.59676876439$**, for** $x = 0, 1, 2, \ldots, 6$ **and** $0$**, otherwise.**

## A Model for the Surface Flaw Distribution

```
dFlaw <- function(x) {
    (1-(x/7)^2)^8*(x >=0)*(x <= 6)/2.59676876439
}
```

```
dFlaw(0:6)
```

```
## [1] 3.850940e-01 3.265337e-01 1.948490e-01
## [4] 7.594130e-02 1.629707e-02 1.275522e-03
## [7] 9.452461e-06
```

**The probabilities of 4 and 5 flaws are now small, but nonzero.**

**The total probability is still**

```
sum(dFlaw(0:6))
```

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

## Simulating from the Distribution Model

```r
rFlaw <- function(n) {
    F <- function(f) sum(dFlaw(0:f))
    U <- runif(n)
    X <- numeric(n)
    for (j in 0:5) {
        X[U > F(j)] <- j + 1
    }
return(X)
}
```

```r
table(rFlaw(100000))/100000
```

```
## 
##       0       1       2       3       4       5 
## 0.38596 0.32592 0.19316 0.07723 0.01635 0.00137 
##       6 
## 0.00001
```

## Expected Values

The expected value of a distribution is a single number which provides a partial summary of the distribution of a random variable $X$.

It is also known as the mean and denoted as $E[X]$.

The expected value is calculated according to

$$E[X] = \sum_{j=0}^{\infty} j d_X(j)$$

where $d_X(j) = P(X = j)$.

## Expected Values

**Example: Find the expected value of the data-driven model for the counts of flaws:**

```
j <- 0:6
sum(dFlaws(j)*j)

## [1] 1.08
```

**Note that this matches the average of the data exactly:**
$(39 + 12(2) + 13(3) + 1(6))/100 = 1.08$.

## Expected Values

**Example: Find the expected value of the alternative model for the counts of flaws:**

```
j <- 0:6
sum(dFlaw(j)*j)

## [1] 1.015678
```

**Note that this no longer matches the average of the data but is not far off.**

## Variance and Standard Deviation

**How far is far? The standard deviation can often provide a useful measure of distance.**

$$\textbf{Var}(X) = E[X^2] - (E[X])^2.$$

$$S = \sqrt{\textbf{Var}(X)}.$$

**Dividing by the square root of the sample size gives us the standard error which can be used to assess distance between an average and an expected value.**

## Variance and Standard Deviation

**Example: Find the standard deviation of the alternative model for the counts of flaws:**

```r
j <- 0:6
V <- sum(dFlaw(j)*j^2) - (sum(dFlaw(j)*j))^2
SD <- sqrt(V)
SD/sqrt(100)   # standard error

## [1] 0.1025076
```

**Note that `sum(dFlaw(j)*j^2)` is the implementation of $E[X^2]$, where $j^2$ represents a squared value taken from the distribution of $X$.**

## Special Discrete Random Variable Models

- Bernoulli
- Binomial
- Geometric
- Poisson
- Negative Binomial

## Bernoulli Random Variables

A Bernoulli trial is an experiment in which there are only 2 possible outcomes.

For example, a light bulb may work or not work; these are the only possibilities.

Each outcome (‘work’ or ‘not work’) has a probability associated with it; the sum of these two probabilities must be 1.

Other possible outcome pairs are: (living, dying), (success, failure), (true, false), (0, 1), (-1, 1), (yes, no), (black, white), (go, stop) . . ..

$\rightsquigarrow$ binary data

## Simulating a Bernoulli Random Variable

**We could also think about outcomes that come from simulating a uniform random variable $U$ on $[0, 1]$.**

**For example, the event that $U$ is less than 0.2 is a possible outcome. It occurs with probability 0.2. It does not occur with probability 0.8.**

**Outcome pair: ($U < 0.2, U \geq 0.2$)**

**We can associate the event $U < 0.2$ with an event that we want to simulate.**

```
set.seed(88832)  # use this to replicate the results below
```

## Simulating Guess Outcomes on a Multiple Choice Test

**Consider a student who guesses on a multiple choice test question which has 5 possible answers, of which exactly 1 is correct.**

**The student may guess correctly with probability 0.2 and incorrectly with probability 0.8.**

**We can simulate the correctness of the student on one question with a $U[0,1]$ random variable. If the outcome is `TRUE`, the student guessed correctly; otherwise the student is incorrect.**

```
U <- runif(1)   # generate U[0,1] number
U

## [1] 0.7125406

U < 0.2 # test U < 0.2 and simulate student's outcome

## [1] FALSE
```

## Simulating Guess Outcomes on a Multiple Choice Test

**The student guesses at another question:**

```
U <- runif(1)   # generate U[0,1] number
U

## [1] 0.7214004

U < 0.2   # student outcome

## [1] FALSE
```

## Simulating Guess Outcomes on a Multiple Choice Test

**Suppose we would like to know how well such a student would do on a multiple choice test consisting of 20 questions.**

**Again, each question corresponds to an independent Bernoulli trial with probability of success equal to 0.2.**

**R can do the simulation as follows:**

```
guesses <- runif(20)
correct <- (guesses < 0.2)
correct

##  [1]  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE
##  [8] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [15] FALSE FALSE FALSE FALSE FALSE FALSE
```

## A Quick Way to Calculate a Student’s Score

**The total number of correct guesses can be calculated.**

```
table(correct)
```

```
## correct
## FALSE  TRUE
##    14     6
```

**Our simulated student would score** $6/20$**.**

## Explanation

In the preceding example, we could associate the values ‘1’ and ‘0’ with the outcomes from a Bernoulli trial.

This defines the Bernoulli random variable: a random variable which takes the value 1 with probability $p$, and 0 with probability $1 - p$.

## Expected Value of a Bernoulli Random Variable $X$

**The expected value of a Bernoulli random variable is $p$: $E[X] = p$.**

**This follows from the fact that $X = 1$ with probability $p$ and $X = 0$ with probability $1 - p$:**

$$E[X] = 0 \times P(X = 0) + 1 \times P(X = 1) = 1p = p.$$

## Variance of a Bernoulli Random Variable $X$

$$E[X^2] = E[X] \text{ since } X^2 = X$$

**which is true for any variable which can only take on values 0 or 1.**

**Variance:**

$$\mathbf{Var}(X) = E[X^2] - (E[X])^2 = E[X] - (E[X])^2 = p - p^2.$$

**Therefore, the theoretical variance of $X$ is $p(1-p)$. (Standard deviation is $\sqrt{p(1-p)}$).**

## Simulated Student Example

**A student would expect to guess correctly on the multiple choice questions 20% of the time; our simulated student was a little bit lucky, obtaining a mark of 30%.**

## Binary Data with a Trend

Suppose there are 40 students in the class, and some study and some do not.

Let $s$ denote the number of hours that a student studies. A possible model for the probability of answering a question correctly might be

$$p(s) = .8 - .6\mathbf{e}^{-s}$$

which gives the probability of 0.2 for no studying, and a probability of .8 for an unlimited amount of studying (there are other factors besides studying that influence a student's performance).

We might model the number of hours of study for each student with a uniform distribution on $[0, 4]$.

## Binary Data with a Trend

**We can simulate such a class using**

```
n <- 40
S <- runif(n, min = 0, max = 4)
S # number of study hours for each student
```

```
##  [1] 3.42819 2.77051 0.00868 3.48061 2.52366
##  [6] 3.59945 3.19887 3.29401 1.09803 1.70981
## [11] 1.80608 1.21620 2.16889 0.15092 1.81191
## [16] 3.94547 2.75876 1.96593 0.30341 1.51880
## [21] 3.88865 0.37969 2.75121 0.60315 1.79566
## [26] 2.35321 3.16678 0.26153 3.67785 1.83443
## [31] 3.35034 0.19594 2.81032 1.88879 2.56764
## [36] 2.49493 1.15094 2.63753 3.36818 1.30130
```

```
U <- runif(n)
U < .8 - .6*exp(-S) #  results for the first question

##  [1] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
##  [8]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
## [15]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
## [22]  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE
## [29]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
## [36]  TRUE  TRUE  TRUE  TRUE  TRUE
```

## Binary Data with a Trend

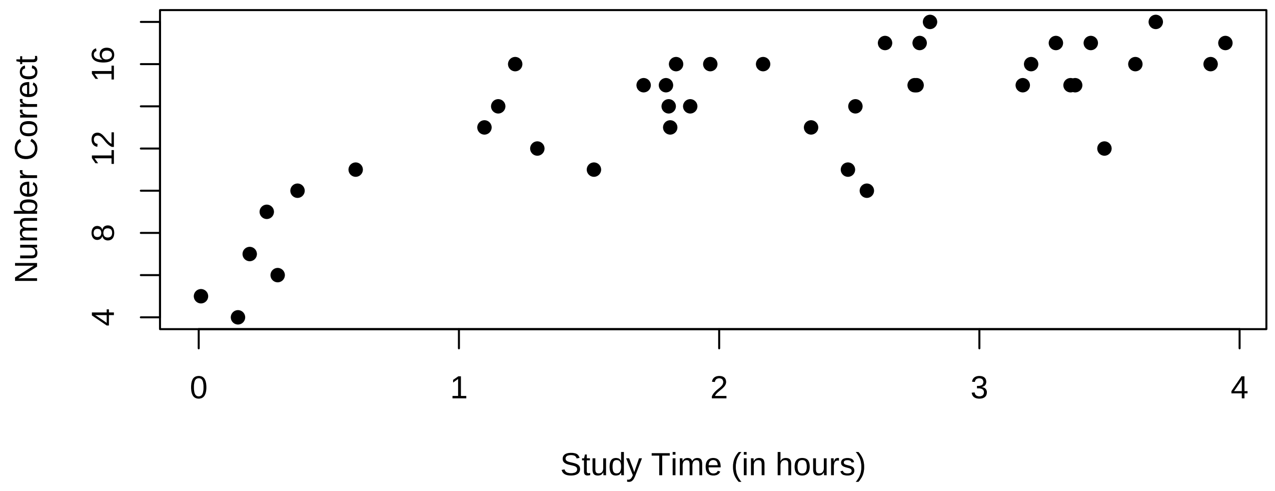
**Simulating the class performance on a 20 question test:**

```
Ncorrect <- (U < .8 - .6*exp(-S)) # 1st question
for (i in 2:20) {
U <- runif(n)
Ncorrect <- Ncorrect + (U < .8 - .6*exp(-S))
}
table(Ncorrect)
```

```
## Ncorrect
##  4  5  6  7  9 10 11 12 13 14 15 16 17 18
##  1  1  1  1  1  2  3  2  3  4  7  7  5  2
```

## Visualizing the Trend

```
plot(Ncorrect ~ S, pch = 16, xlab = "Study Time (in hours)",
    ylab="Number Correct")
```



## Binomial Random Variables

**Let $X$ denote the sum of $m$ independent Bernoulli random variables, each having probability $p$.**

**$X$ is called a binomial random variable; it represents the number of 'successes' in $m$ Bernoulli trials.**

**A binomial random variable can take values in the set $\{0, 1, 2, \ldots, m\}$.**

**Example: When the student guessed at 20 multiple choice questions, the number of correct guesses was a binomial random variable $X$ with $m = 20$ and $p = 0.2$.**

$X \sim$ **bin**$(20, 0.2)$**.**

## Binomial Random Variables

**The probability of a binomial random variable $X$ taking on any one of these values is governed by the binomial distribution:**

$$P(X = x) = \binom{m}{x} p^x (1-p)^{m-x}, \quad x = 0, 1, 2, \ldots, m.$$

**These probabilities can be computed using the dbinom() function.**

## Calculating Binomial Probabilities in R

```
dbinom(x, size, prob)
```

**Here, `size` and `prob` are the binomial parameters $m$ and $p$, respectively, while `x` denotes the number of 'successes'. The output from this function is the value of $P(X = x)$.**

**Example - Guessing on Multiple Choice:**

```
dbinom(6, 20, 0.2)  # probability of exactly 6 correct

## [1] 0.1091
```

**Our simulated student did something that had an 11% chance of occurring.**

## Example

**Compute the probability of getting exactly 4 heads in 6 tosses of a fair coin.**

```
dbinom(x = 4, size = 6, prob = 0.5)

## [1] 0.234375
```

**Thus, $P(X = 4) = 0.234$, when $X$ is a binomial random variable with $m = 6$ and $p = 0.5$.**

## Binomial Probabilities

**Recall the cdf:** $F(x) = P(X \leq x)$**.**

**Cumulative binomial probabilities can be computed using pbinom().**

**This function takes the same arguments as** `dbinom()`**.**

**Example: The probability of a student scoring 6 or less by guessing on a multiple choice test is**

```
pbinom(6, 20, .2)

## [1] 0.913307
```

## Binomial Probabilities

**The probability of a student scoring 6 or more by guessing on a multiple choice test is**

```
1 - pbinom(5, 20, .2)

## [1] 0.195792
```

**This means that our simulated student is not highly unusual.**

**Example: The probability of a student scoring 10 or more by guessing on a multiple choice test is**

```
1 - pbinom(9, 20, .2)

## [1] 0.00259483
```

**A student who passes the test purely by guessing would be unusually lucky. This is an example of a p-value for a test of the hypothesis that the student is guessing. In this case, we might infer that a student who passes the test is not just guessing.**

## Binomial Pseudorandom Numbers

**The rbinom() function can be used to generate binomial pseudorandom numbers.**

```
rbinom(n, size, prob)
```

**Here, `size` and `prob` are the binomial parameters $m$ and $p$, while `n` is the number of variates generated.**

**Simulating 12 other students' perfomances after guessing on a multiple choice test with 20 questions:**

```
rbinom(12, 20, 0.2)

##  [1] 8 2 4 4 2 5 1 3 5 5 2 6
```

## Expected Value of a Binomial Random Variable $X$

A binomial random variable is the sum of $m$ independent Bernoulli random variables, each with success probability $p$:

$$X = \sum_{j=1}^{m} X_j$$

where $X_j$ is Bernoulli with probability $p$. Because expected values are based on summation, it is possible to show that

$$E[X] = \sum_{j=1}^{m} E[X_j].$$

We can conclude that

$$E[X] = mp.$$

## Expected Values of Products of Random Variables

**If $X_1$ and $X_2$ are independent Bernoulli random variables, then**

$$P(X_1 = x_1, X_2 = x_2) = P(X_1 = x_1)P(X_2 = x_2).$$

**Therefore,** $E[X_1X_2] = \sum_{x_1=0}^{1}\sum_{x_2=0}^{1} x_1x_2P(X_1 = x_1, X_2 = x_2)$.
**Independence allows us to write**

$$E[X_1X_2] = \sum_{x_1=0}^{1}\sum_{x_2=0}^{1} x_1x_2P(X_1 = x_1)P(X_2 = x_2).$$

**Reordering the sums:**

$$E[X_1X_2] = \sum_{x_1=0}^{1} x_1P(X_1 = x_1)\sum_{x_2=0}^{1} x_2P(X_2 = x_2) = E[X_1]E[X_2].$$

**This result holds for all pairs of independent random variables.**

## Expected Values of Products of Sums of Random Variables

**If $X = \sum_{j=1}^{m} X_j$, then $X^2 = \sum_{j=1}^{m} X_j^2 + \sum_{j=1}^{m} \sum_{i=1, i\neq j}^{m} X_i X_j$.**

**Therefore, $E[X^2] = \sum_{j=1}^{m} E[X_j^2] + \sum_{j=1}^{m} \sum_{i=1, i\neq j}^{m} E[X_i]E[X_j]$.**

**If the $X_j$'s all have the same distribution, then $E[X_j]$ will be constant (say, $a$, and so will $E[X_j^2$ (say, $b$) giving us**

$$E[X^2] = ma + m(m-1)b.$$

**In the case of the independent Bernoulli $p$ random variables, $E[X_j] = E[X_j^2] = p$, so**

$$E[X^2] = mp + m(m-1)p^2.$$

## Variances of Sums of Random Variables

The argument on the previous slide can be complete to show that the variance of a sum of independent random variables is the same as the sum of the variances of those individual random variables.

In the case where $X$ is a sum of $m$ independent Bernoulli random variables, we have

**Var**$(X) = E[X^2] - (E[X])^2 = mp + m^2p^2 - mp^2 - m^2p^2 = mp(1-p).$

The standard deviation is $\sqrt{mp(1-p)}$.

## A Slightly More Realistic Simulation

**A student that guesses would represent a kind of worst-case scenario while a student that gets correct answers every time would represent the best-case scenario.**

**We could model a class of 12 different students using a uniform random variable to represent their probability of answering correctly.**

```
U <- runif(12, min=0.2, max=0.9)
U

##  [1] 0.343702 0.744337 0.284354 0.537338 0.484097
##  [6] 0.276168 0.351801 0.420542 0.871380 0.218155
## [11] 0.623122 0.275150
```

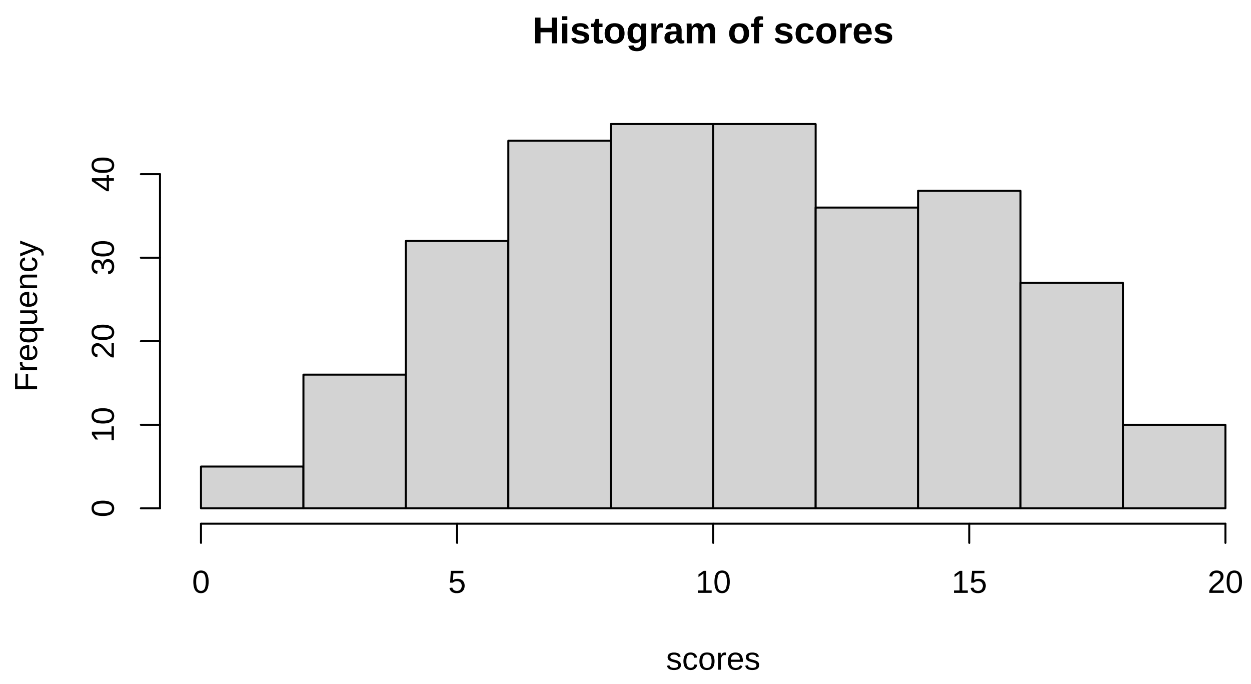
**Simulating 12 different students' perfomances after writing a multiple choice test with 20 questions:**

```
rbinom(12, 20, U)

##  [1]  6 13  5  6 10  7  6  9 19  3 10  7
```

## Simulating a Larger Class

**We could model a larger class, say of 300 students:**

```
U <- runif(300, min=0.2, max=0.9)
scores <- rbinom(300, 20, U)
hist(scores)
```



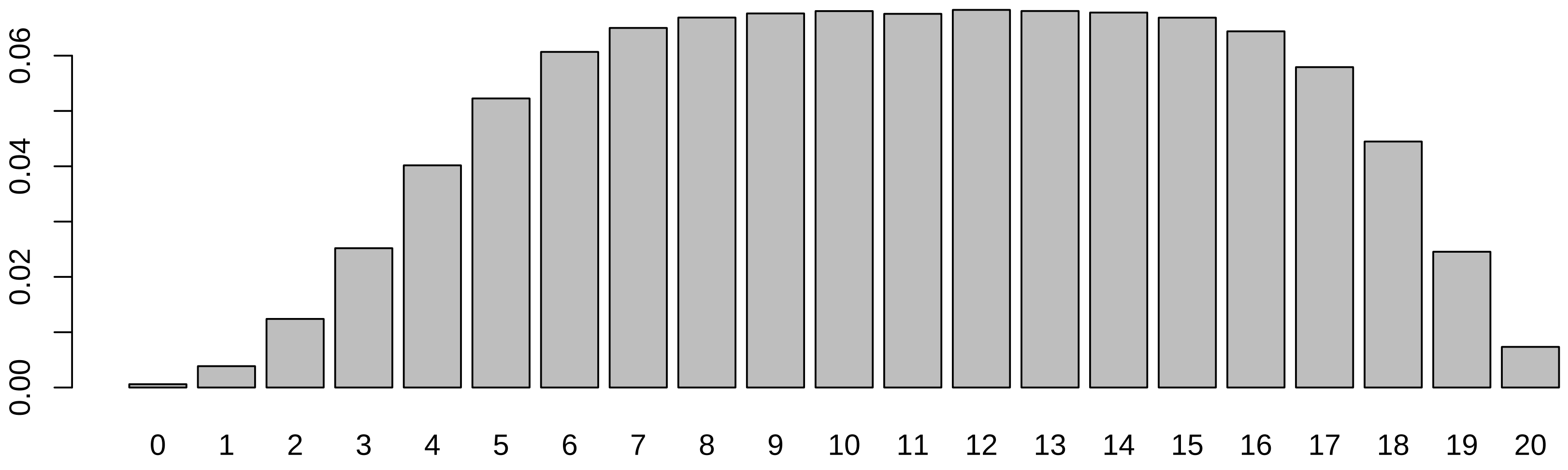
## Using Simulation to Visualize a New Distribution

**According to the model we have developed, a randomly selected student's score $S$ is a binomial random variable, conditional on the amount of studying and aptitude, summarized by a uniform random variable on $[0.2, 1.0]$.**

**We can visualize the distribution of the random variable $S$ by simulating a large number of such variables. The code and plot are on the next slide.**

## Using Simulation to Visualize a New Distribution

```r
Nsims <- 1000000
U <- runif(Nsims, min=0.2, max=0.9)
scores <- rbinom(Nsims, 20, U)
barplot(table(scores)/Nsims)
```



## Example - Control Charting

Suppose 10% of the windshields produced on an assembly line are defective, and suppose 15 windshields are produced each hour.

Each windshield is independent of all other windshields.

This process is judged to be out of control when more than 4 defective windshields are produced in any single hour.

Simulate the number of defective windshields produced for each hour over a 24-hour period, and determine if any process should have been judged out of control at any point in that simulation run.

## Control Charting

**One such simulation run is:**

```
defectives <- rbinom(24, 15, 0.1)
defectives


##  [1] 2 3 1 3 0 3 0 1 2 2 3 1 2 0 1 1 1 1 1 1 1 1 1
## [24] 0
```

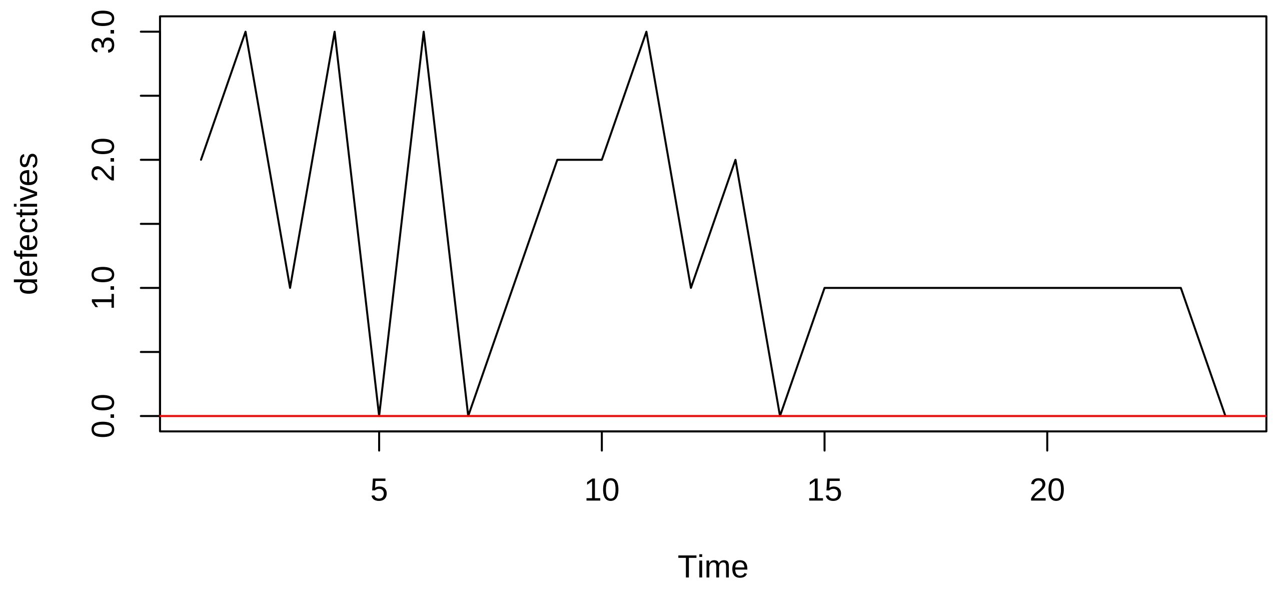
```
any(defectives > 4)  # any() asks if any of its arguments are TRUE

## [1] FALSE
```

**None of the defective counts exceed 4. The process is in control and the simulated data is in control.**

## Control Charting

**Usually, a control chart is drawn:**

```
ts.plot(defectives)
abline(h = c(0, 4), col="red")
```



**Nothing plots outside of the control limits (drawn in red).**

## Control Charting

**Another simulation. This time the true proportion defective is larger than 0.1 occasionally. Is this out of control condition detected by the control chart?**

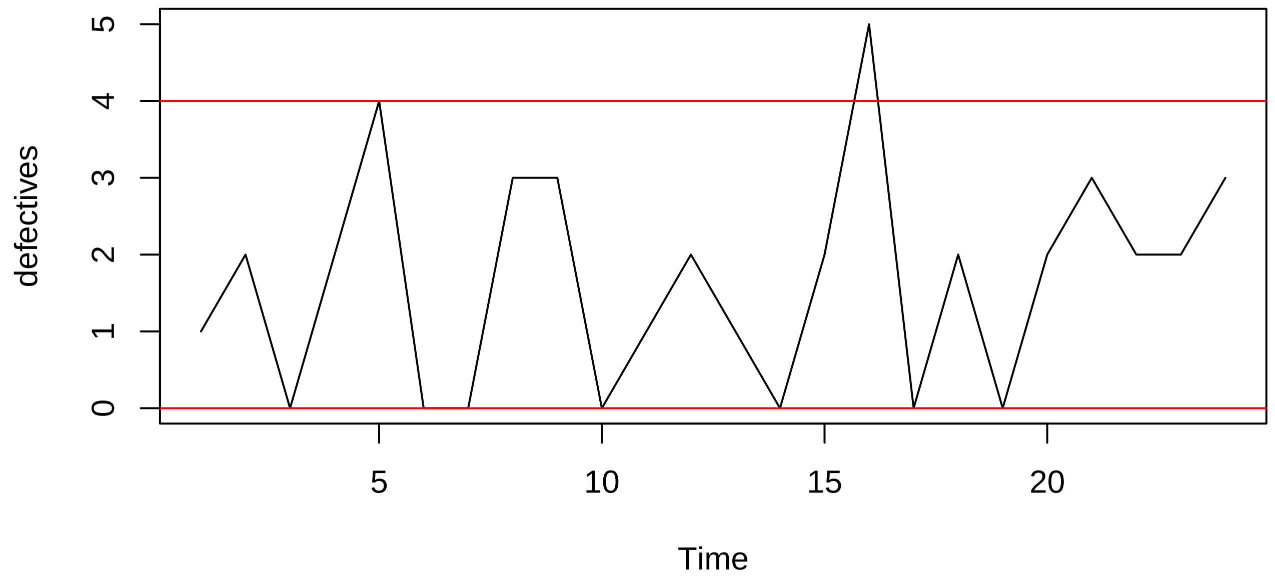
```
defectives <- rbinom(24, 15, 0.1+0.1*rbinom(24, 1, .3))
defectives


##  [1] 1 2 0 2 4 0 0 3 3 0 1 2 1 0 2 5 0 2 0 2 3 2 2
## [24] 3
```

```
any(defectives > 4)  # any() asks if any of its arguments are TRUE

## [1] TRUE
```

**The out of control condition is detected.**

## Visualizing the Result

```
ts.plot(defectives)
abline(h = c(0, 4), col="red")
```



## Simulating geometric random variables

**We saw that a binomial random variable is defined as the sum of $m$ independent Bernoulli random variables.**

**Now, we will define a geometric random variable as the *number* of Bernoulli random variables that must be generated before the first 1 appears.**

## Simulating geometric random variables

**Example. Suppose we toss a coin repeatedly and want to count the number of heads (0) until we toss the first tail.**

**If we simulate 5 coin tosses, we obtain**

```
p <- 0.5; # probability of a tail
runif(5) < p

## [1]  TRUE  TRUE FALSE  TRUE FALSE
```

**The first 2 tosses are heads and the third toss is a tail, so in this example, the geometric random number would be 2.**

**The third, fourth and fifth tosses weren't needed, so we would be better off using a `while()` loop here.**

## Simulating geometric random variables

**Example.**

```
set.seed(123488) # use this seed to get our result
p <- 0.5; # probability of a tossing a tail
G <- 0 # eventual value of geometric random variable
U <- runif(1) # first coin toss
IsItaTail <- (U <= p) # this will be TRUE when we toss a tai
while (IsItaTail == FALSE) {
    G <- G + 1 # add one to G until IsItaTail is TRUE
    U <- runif(1)
    IsItaTail <- (U <= p)
}
G


## [1] 1
```

## Simulating geometric random variables

**Example. This time simulate the number of independent die rolls until rolling a 6.**

```
p <- 1/6; # probability of a rolling a 6
G <- 0 # eventual value of geometric random variable
U <- runif(1) # first die roll
IsIt6 <- (U < p) # TRUE if we roll a 6
while (IsIt6 == FALSE) {
    G <- G + 1 # add one to G until IsIt6 is TRUE
    U <- runif(1)
    IsIt6 <- (U < p)
}
G


## [1] 12
```

## Simulating geometric random variables

**Example. This time simulate the number of independent 2-dice rolls until rolling a 12.**

```r
p <- 1/6; # probability of a rolling a 6
U <- runif(2) # first 2 dice rolls
G <- 0 # eventual value of geometric random variable
IsIt12 <- (U[1]<1/6)&(U[2]<1/6) # TRUE if both dice are 6's
while (IsIt12 == FALSE) {
    G <- G + 1 # add one to G until both dice are 6's
    U <- runif(2)
    IsIt12 <- (U[1]<1/6)&(U[2]<1/6) #
}
G
```

```
## [1] 4
```

## Simulating geometric random variables

**We can also use the `rgeom()` function to simulate these numbers. For example, to simulate the number of die rolls until the first 6, use**

```
G <- rgeom(1, p = 1/6)
G

## [1] 3
```

**To simulate the number of 2-dice rolls until the first 12 (an event with probability 1/36), use**

```
G <- rgeom(1, p = 1/36)
G

## [1] 65
```

## Simulating geometric random variables

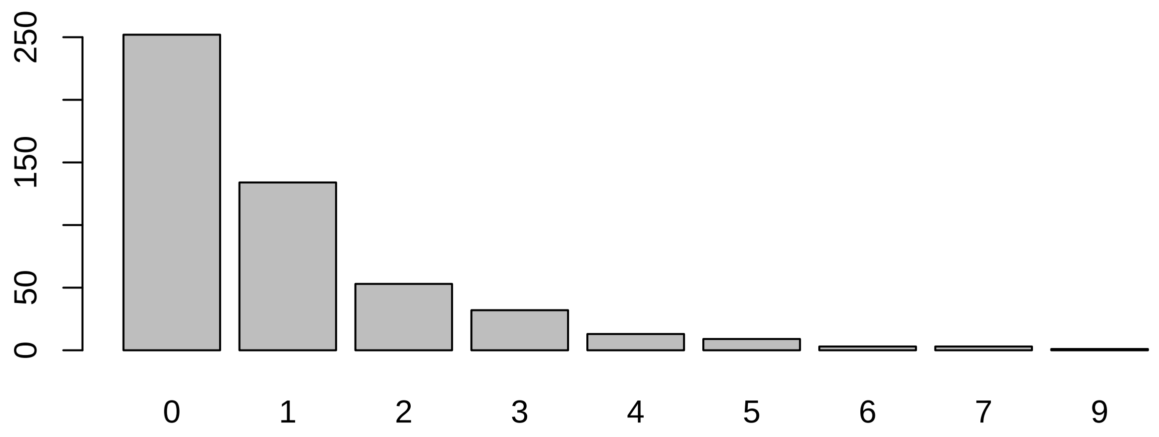
**Let's look at the distribution of a geometric random variable where $p = 1/2$ by simulating 500 values and tabulating the result:**

```
N <- 500;
G <- rgeom(N, p = 1/2)
table(G)

## G
##   0   1   2   3   4   5   6   7   9
## 252 134  53  32  13   9   3   3   1
```

## Simulating geometric random variables

**The bar plot is**

```
barplot(table(G))
```



**From the plot, we see that the probability of a 0 is higher than for a 1 which is a higher than for a 2 and so on.**

## Calculating geometric probabilities

**The probability of a geometric random variable $X$ taking on any value can be computed using the dgeom() function.**

**dgeom(x, prob)  Here, `prob` is the parameter $p$, while `x` denotes the number of trials before the first 'success'. The output from this function is the value of $P(X = x)$.**

## Example

**Compute the probability that it will take 5 die rolls before obtaining the first 6.**

```
dgeom(x = 5, prob = 1/6)

## [1] 0.0669796
```

**Thus, $P(X = 5) = 0.067$, when $X$ is a geometric random variable with $p = \frac{1}{6}$.**

## Example

**Compute the probability that it will take 5 *or fewer* die rolls before obtaining the first 6. (Use the `pgeom()` function for this.)**

```
pgeom(5, prob = 1/6)

## [1] 0.665102
```

**Thus, $P(X \leq 5) = 0.6651$, when $X$ is a geometric random variable with $p = \frac{1}{6}$.**

## The probability of a 100-year disaster

**The geometric distribution is the simplest of all models that can be used to predict the occurrence of a disaster, such as a flood.**

**If the probability of a disaster in a given year is .01, how long would we expect to wait for the event?**

**If we simulate a large number of geometric random variables with $p = .01$, we can visual the distribution of the waiting time:**

```
W <- rgeom(500, p = .01)
mean(W) # this gives us the average waiting time
```

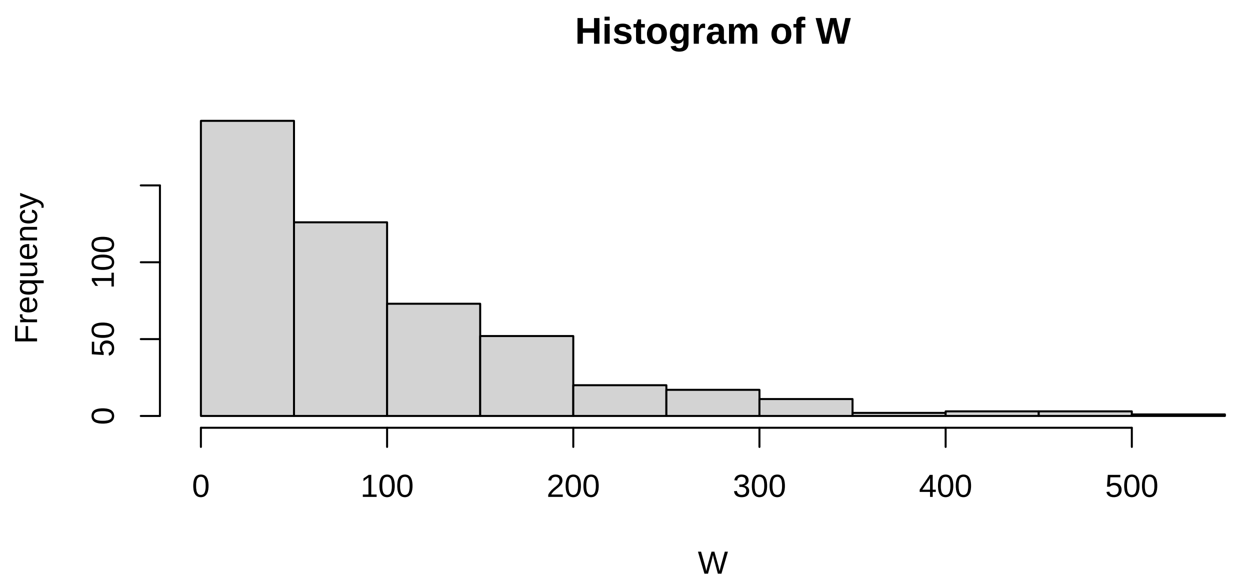
```
## [1] 97
```

**This is what is meant by a 100-year event.**

## The probability of a 100-year disaster

**But note that the 100-year event could happen pretty soon:**

```
hist(W)
```



**The probability of the event occurring within the next 25 years is**

```
pgeom(25, p = .01)

## [1] 0.229957
```

## The probability of a 100-year disaster

**Example.**

**Using the binomial distribution, calculate the probability that 2 such disasters could occur in the same 100 year period.**

**The probability of 1 or fewer disasters in 100 years is**

```
pbinom(1, 100, p=.01)

## [1] 0.735762
```

**so we can subtract this from 1 to get the required probability:**

```
1-pbinom(1, 100, p=.01)

## [1] 0.264238
```

## The Expected Value of a Geometric Random Variable

**For the geometric random variable $X$ with $P(X = j) = p(1-p)^j$, for $j = 1, 2, \ldots$, it can be shown using arguments involving geometric series, that**

$$E[X] = \frac{1-p}{p}.$$

**Note that if the variable is defined so that $P(X = j) = p(1-p)^{j-1}$, then**

$$E[X] = \frac{1}{p}.$$

## Poisson Random Variables

**The Poisson distribution is the limit of a sequence of binomial distributions with parameters $n$ and $p_n$, where $n$ is increasing to infinity, and $p_n$ is decreasing to 0, but where the expected value (or mean) $np_n$ converges to a constant $\lambda$.**

**The variance $np_n(1 - p_n)$ converges to this same constant.**

**Thus, the mean and variance of a Poisson random variable are both equal to $\lambda$.**

**This parameter is sometimes referred to as a *rate*.**

## Applications of Poisson Random Variables

Poisson random variables arise in a number of different ways.

They are often used as a crude model for count data.

Examples of count data are the numbers of earthquakes in a region in a given year, or the number of individuals who arrive at a bank teller in a given hour.

The limit comes from dividing the time period into $n$ independent intervals, on which the count is either 0 or 1.

The Poisson random variable is the total count.

## Distribution of Poisson Random Variables

**The possible values that a Poisson random variable $X$ could take are the non-negative integers $\{0, 1, 2, \ldots\}$.**

**The probability of taking on any of these values is**

$$P(X = x) = \frac{e^{-\lambda}\lambda^x}{x!}, \quad x = 0, 1, 2, \ldots.$$

## Calculation of Poisson Probabilities

**The Poisson probabilities can be evaluated using the dpois() function.**

```
dpois(x, lambda)
```

**Here,** `lambda` **is the Poisson rate parameter, while** `x` **is the number of Poisson events. The output from the function is the value of** $P(X = x)$**.**

## Example

The average number of arrivals per minute at an automatic bank teller is 0.5. Arrivals follow a Poisson process. (Described later.)

The probability of 3 arrivals in the next minute is

```
dpois(x = 3, lambda = 0.5)

## [1] 0.0126361
```

Therefore, $P(X = 3) = 0.0126$, if $X$ is Poisson random variable with mean $0.5$.

## Poisson Probabilities

**Cumulative probabilities of the form $P(X \leq x)$ can be calculated using ppois().**

**Example: Find the probability that a Poisson random $X$ with mean 3 (or rate 3) is less than or equal to 2:**

```
ppois(2, 3)

## [1] 0.42319
```

**Find the probability that such a random variable is larger than 6:**

```
1 - ppois(6, 3)

## [1] 0.0335085
```

## Poisson Pseudorandom Numbers

**The following function shows how one might simulate large numbers of independent Poisson variates:**

```r
rPois <- function(n, lambda) {
    U <- runif(n)
    X <- numeric(n)
    x <- 0
    while (max(U) > ppois(x, lambda)) {
        X[U >= ppois(x, lambda)] <- x + 1
        x <- x+1
    }
    return(X)
}
```

## Poisson Pseudorandom Numbers

**Example:**

```
n <- 100000; rate <- 3
X <- rPois(n, rate)
X[1:5] # the first 5 variates

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

## Poisson Pseudorandom Numbers

**Tabular summary of the Poisson counts:**

```
table(X)

## X
##     0     1     2     3     4     5     6     7
##  5007 14946 22474 22335 16755 10140  4962  2135
##     8     9    10    11    12    13
##   887   243    82    26     7     1
```

## Poisson Pseudorandom Numbers

**Comparison of the simulated probabilities with theoretical probabilities:**

```
table(X)/n - dpois(0:max(X), rate)
```

```
## X
##             0             1             2 
##  2.82932e-04  9.87949e-05  6.98192e-04 
##             3             4             5 
## -6.91808e-04 -4.81356e-04  5.81187e-04 
##             6             7             8 
## -7.89407e-04 -2.54031e-04  7.68488e-04 
##             9            10            11 
## -2.70504e-04  9.84882e-06  3.90497e-05 
##            12            13 
##  1.47624e-05 -2.74713e-06
```

## Poisson Pseudorandom Numbers

**We can also generate Poisson random numbers using the rpois() function.**

```
rpois(n, lambda)
```

**The parameter `n` is the number of variates produced, and `lambda` is as above.**

## Rain Events - Poisson Distribution Example

**Suppose rain events occur in a particular area, daily, between May 1 and Sept 15, according to a Poisson distribution with rate 0.6 per day.**

**Simulate the numbers $N$ of daily rain events for this 138 period, assuming independence from day to day.**

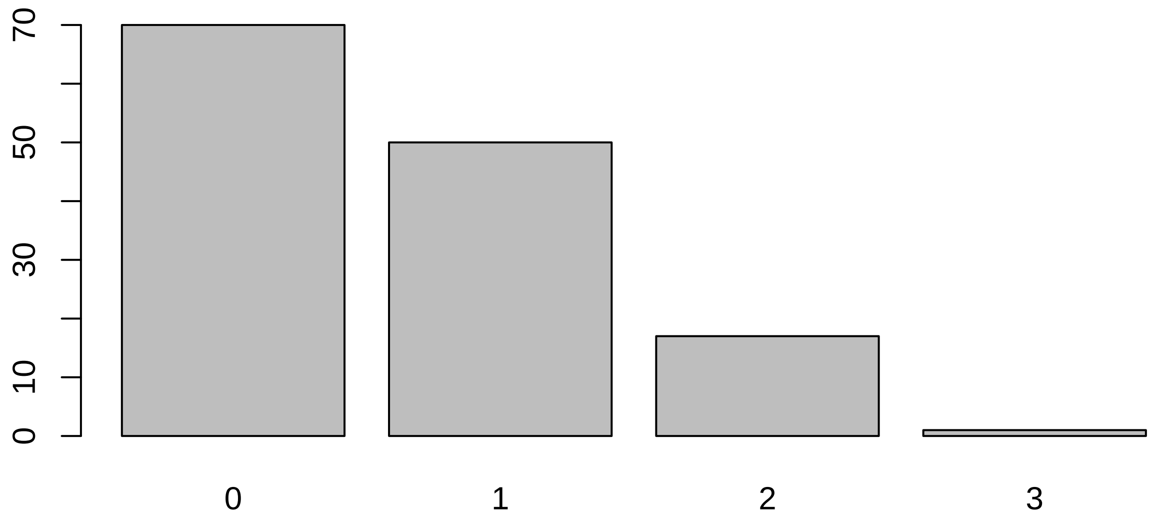
```
N <- rpois(138,  0.6)
```

**Calculate the mean and variance of $N$.**

```
mean(N)

## [1] 0.630435

var(N)

## [1] 0.526658
```

## Rain Events - Poisson Distribution Example

```r
barplot(table(N))
```



## Negative Binomial Model

**In fact, the observed average number of daily rain events in the region 0.45 and the variance is 0.73.**

**This means the Poisson distribution is not really appropriate as a model for this data. The mean and variance should match.**

**The data are over-dispersed. The variance is larger than the mean.**

**One model for over-dispersed data is the negative binomial model.**

## Negative Binomial Model

A negative binomial random variable counts up the number of Bernoulli ($p$) trials until the $r$th success occurs.

If $X$ is a negative binomial random variable, then

$$E[X] = r(1-p)/p$$

and

$$\text{Var}(X) = r(1-p)/p^2$$

The first result can be obtained by noting that $X$ is the sum of $r$ geometric random variables. The second follows, except that the independence of the geometric random variables is also needed.

## Negative Binomial Probabilities

```
dnbinom(x, size, prob)
```

**Here, `size` and `prob` are the parameters $r$ and $p$, respectively, while `x` denotes the number of observed trials until the $r$th 'success'. The output from this function is the value of $P(X = x)$.**

**Example - The probability that it takes 6 trials before a student guesses 2 multiple choice questions correctly is**

```
dnbinom(6, 2, 0.2) # probability that 6 guesses are required

## [1] 0.0734003
```

## Negative Binomial Pseudorandom Numbers

**We can generate negative binomial random numbers using the rnbinom() function.**

```
rnbinom(n, size, prob)
```

**The parameter `n` is the number of variates produced, and `size` is $r$ and `prob` is $p$ as above.**

**Note that $r$ does not have to be an integer. The model is more general than the interpretation given on the previous slide suggests.**

## Rain Events - Negative Binomial Example

**Suppose rain events occur in a particular area, daily, between May 1 and Sept 15, according to a Negative binomial distribution with $r = 0.52$ and $p = 0.57$ (these parameters would be estimated from real data).**

**Simulate the numbers $N$ of daily rain events for this 138 period, assuming independence from day to day.**

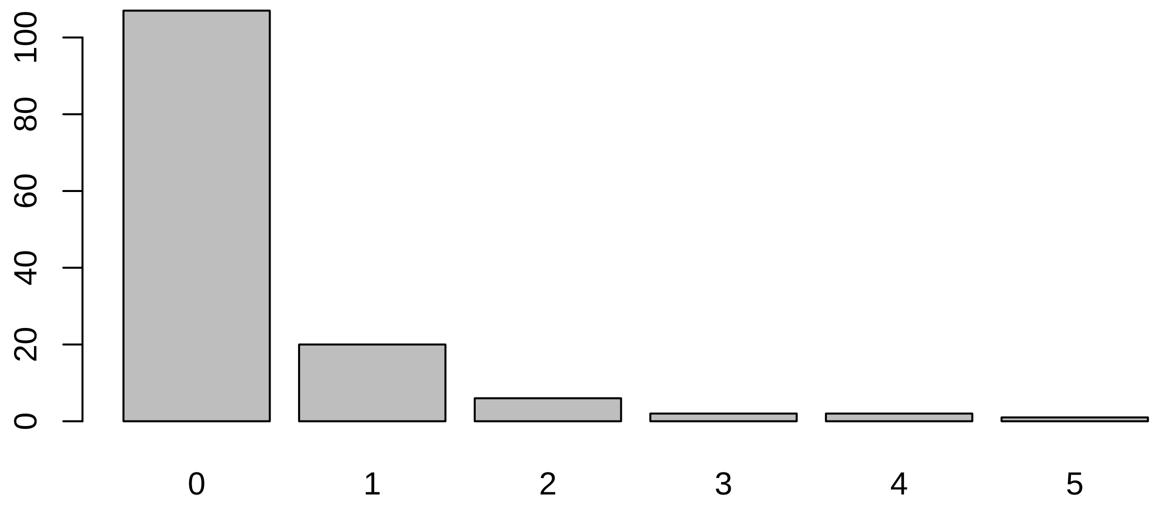
```
N <- rnbinom(138,  0.52, 0.57)
```

**Calculate the mean and variance of $N$.**

```
mean(N)

## [1] 0.369565

var(N)

## [1] 0.731038
```

## Rain Events - Negative Binomial Example

```r
barplot(table(N))
```



## Summary

We can generate the building blocks for discrete simulation using uniform random numbers.

Bernoulli random variables can be generated from uniforms.

Binomial random variables are sums of independent Bernoullis and are a basic model for counting defectives.

Poisson random variables are a basic model for counting defects.

Negative binomial variables are sometimes useful as a more accurate model than the Poisson. (Geometric random variables are a special case.)

## Exercises

1. Crates containing a large number of batteries are monitored by randomly selecting 20 from each crate and running an accelerated life test. If 2.5 percent of the batteries are defective under normal conditions, identify a model for the number of defective batteries in a given sample.

2. Identify an appropriate alternative model for the refrigerator surface flaw data.