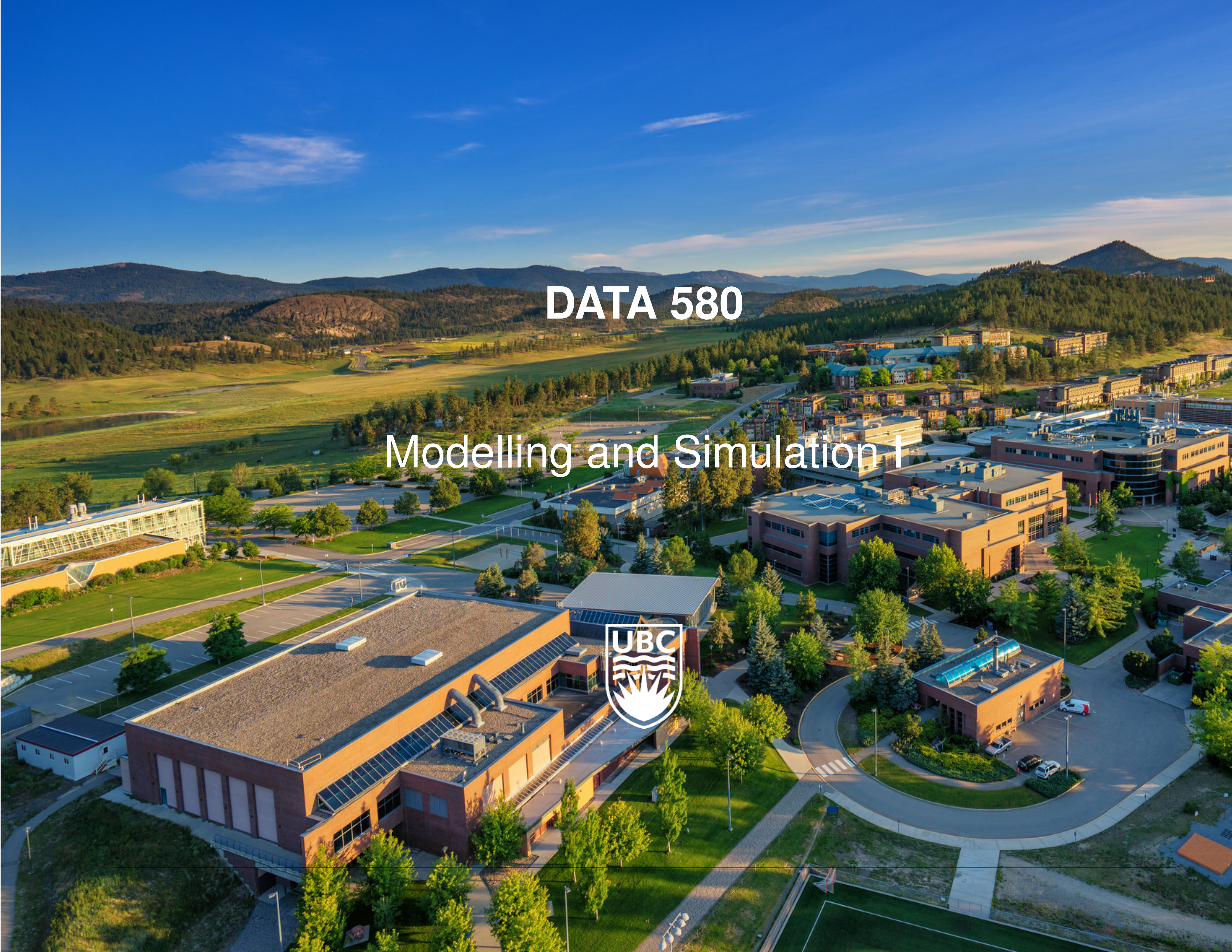


DATA 580

Modelling and Simulation I

Probability, and Discrete Random Variable Simulation

Probability Density Function (pdf)

Calculation of Probabilities: Cumulative distribution function (cdf)

Expected Value (E)

Variance (Var)

Simulation of Discrete Random Variables (Bernoulli, Binomial, Poisson, Negative Binomial)

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

Probability Density Function (pdf)

A continuous function $f(x)$ is a probability density function if it is always nonnegative, and the area under its graph is exactly 1.0. That is,

$$f(x) \geq 0, \text{ for all } x$$

and

$$\int_{-\infty}^{\infty} f(x)dx = 1.$$

All probability density functions have these two properties.

The pdf completely characterizes the probability model.

The pdf is highest at values of x that are most probable.

Uniform Random Variables

The function

$$f_U(x) = \begin{cases} \frac{1}{b-a}, & \textbf{for } x \in [a,b] \\ 0, & \textbf{otherwise} \end{cases}$$

is an example of a pdf since

$$f_U(x) \geq 0$$

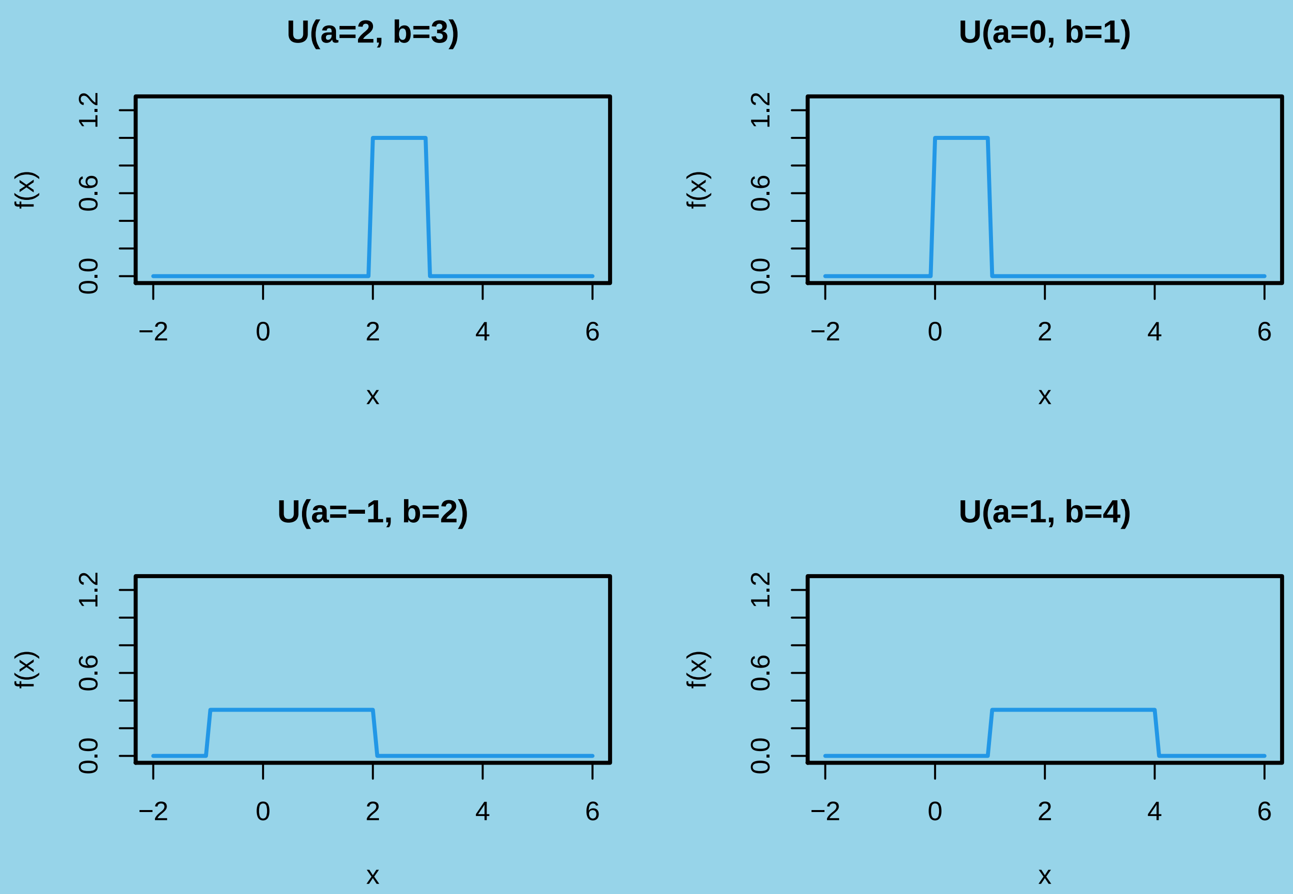
and

$$\int_{-\infty}^{\infty} f_U(x)dx = \int_a^b f_U(x)dx = 1.$$

$f_U(x)$ **is the uniform density function.**

The uniform distribution is a possible model for measurement error but its most important function is as a building block for almost all other distributions.

Picturing Some Examples of the Uniform pdf



The area under the blue curve is 1 in all cases. This represents the probability that the random variable takes a value in the interval $[a, b]$.

Calculation of Probabilities

The probability that a random variable X with density function $f(x)$ takes a value in an interval $[a_1, b_1]$ is calculated as

$$P(a_1 \leq X \leq b_1) = \int_{a_1}^{b_1} f(x)dx.$$

Such probabilities are also expressed in terms of the *cumulative distribution function* (cdf):

$$F(y) = P(X \leq y) = \int_{-\infty}^{y} f(x)dx.$$

$$P(a_1 \leq X \leq b_1) = F(b_1) - F(a_1).$$

Note also that the probability density function can be recovered from the cumulative distribution function by differentiation:

$$f(x) = F'(x).$$

Evaluation of Probabilities in R

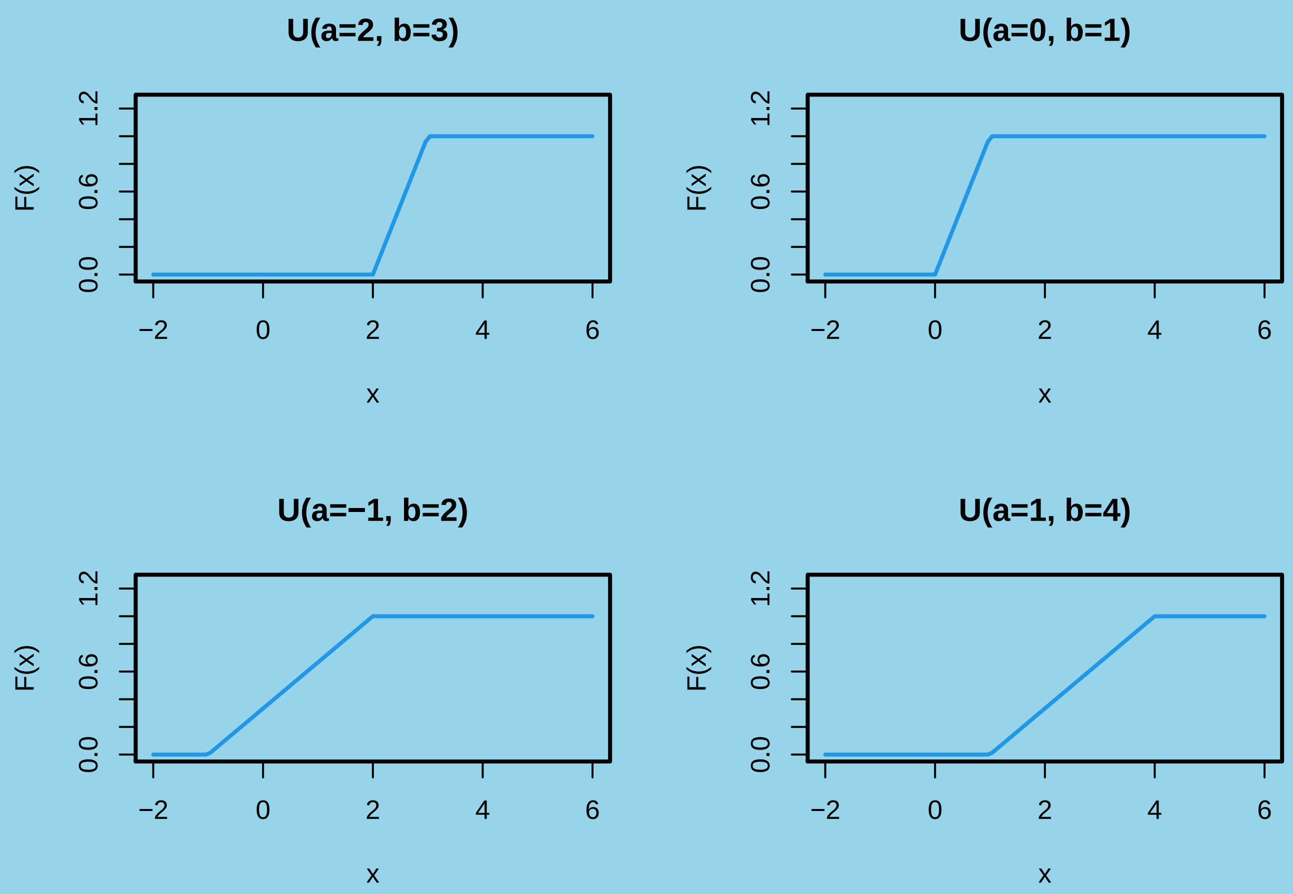
The `punif()` function can be used to calculate the probability that a uniform random variable is less than a given value, i.e. the cumulative distribution function at the given value.

To calculate $F(x) = P(X \leq x)$ the use `punif(x, a, b)`. This explicitly evaluates the uniform cumulative distribution function $F(x) = \frac{x-a}{b-a}$, when x lies in $[a, b]$.

For example,

```
punif(.6, 0, 1)

## [1] 0.6

punif(2.5, 2, 3)

## [1] 0.5
```

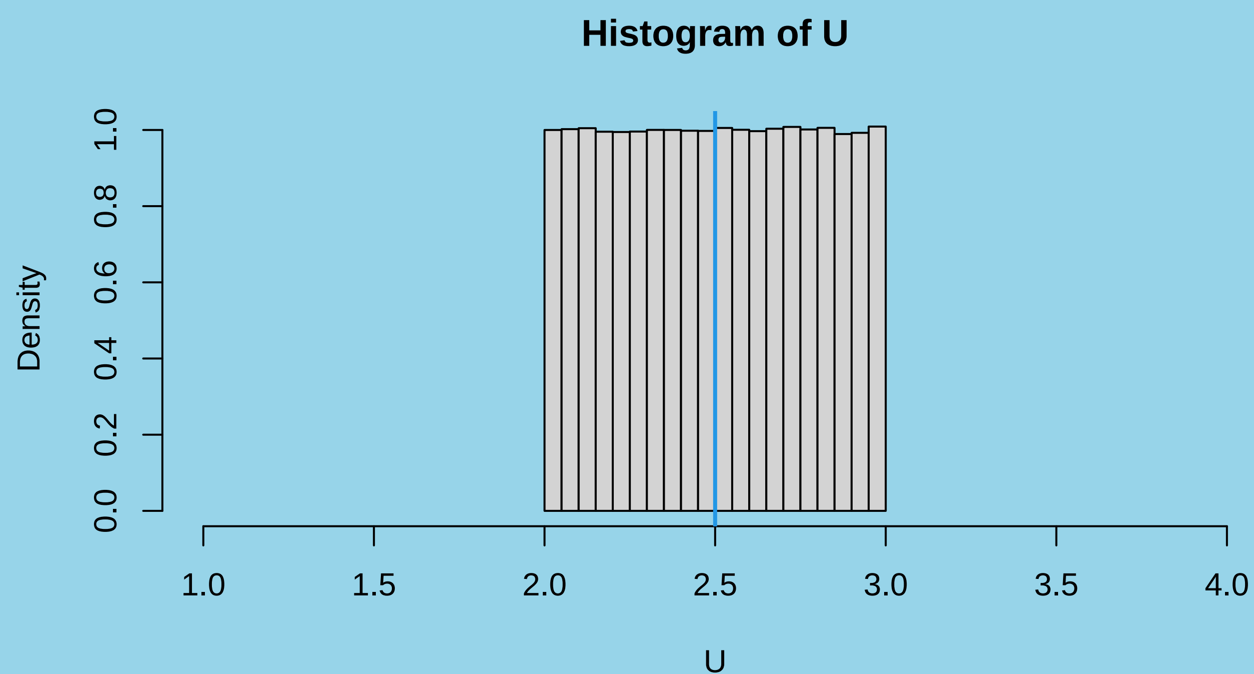
Picturing Some Examples of the Uniform cdf



From these graphs, you can read of the probability that a uniform random variable is less than the given value on the horizontal axis. e.g. in the upper left panel, $F(2.5) = 0.5$ so the $P(U \leq 2.5) = 0.5$.

Estimation of Probabilities by Simulation

```
U <- runif(1000000, 2, 3)
hist(U, freq = FALSE)
abline(v = 2.5, col="blue")
```



The proportion of the area to the left of the blue line estimates the probability that U is less than 2.5.

Estimation of Probabilities by Simulation

Observe:

First 10 simulated uniforms:

```
U[1:10]

##  [1] 2.003 2.484 2.512 2.922 2.301 2.544 2.522 2.963 2.884 2.310
```

Which ones are less than 2.5?

```
U[1:10] < 2.5

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

How many are less than 2.5?

```
sum(U[1:10] < 2.5)    # FALSE is equivalent to 0 in R;  TRUE <--> 1.

## [1] 4
```

What proportion are less than 2.5?

```
sum(U[1:10] < 2.5)/10  # divide by sample size

## [1] 0.4
```

Estimation of Probabilities by Simulation

Fact: The sample mean is equal to the sum of the sample values (0's and 1's here) divided by the sample size.

$\rightsquigarrow$ **Equivalent Calculation:**

```
mean(U[1:10] < 2.5) # proportion less than 2.5

## [1] 0.4
```

More accurate calculation would use the larger sample size:

```
mean(U < 2.5)

## [1] 0.4994
```

Additional Examples: Assume U is Uniform on $[2, 3]$.

Estimate $P(U \leq 2.1)$ and compare with true value.

```
mean(U <= 2.1)

## [1] 0.1001

punif(2.1, 2, 3)

## [1] 0.1
```

Additional Examples: Assume U is Uniform on $[2, 3]$.

Estimate $P(U \leq 2.9)$ and compare with the true value.

```
mean(U <= 2.9)

## [1] 0.8999

punif(2.9, 2, 3)

## [1] 0.9
```

Additional Examples: Assume U is Uniform on $[2, 3]$.

Estimate $P(U > 2.9)$ and compare with the true value.

```r
mean(U > 2.9)

## [1] 0.1001

1 - punif(2.9, 2, 3)

## [1] 0.1
```

Additional Examples: Assume U is Uniform on $[2, 3]$.

Estimate $P(2.1 \leq U < 2.9)$ and compare with the true value.

```
mean(U < 2.9 & U >= 2.1 )

## [1] 0.7998

punif(2.9, 2, 3) - punif(2.1, 2, 3)

## [1] 0.8
```

Expected Value

The expected value of a single (continuous) random variable X can be written as

$$E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

where $f(x)$ is the probability density function of X.

We say $E[X]$ is the *mean* of X.

The expected value gives us a single number that, at least in a rough sense, conveys a typical value for the random variable.

It is sometimes called a measure of *location*, since it specifies the location of the distribution along the real axis.

Expected Value

For the density function $f_U(x)$, we have

$$E[X] = \int_a^b x/(b-a)dx = \frac{b+a}{2}. \tag{1}$$

In other words, the expected value of a uniform random variable is at the midpoint of the interval.

A commonly used alternate notation for the mean of a distribution is μ, the Greek letter which roughly translates to the letter “m”.

Expected Value

Other types of expected value can be calculated by the appropriate integration. For continuous functions $g(x)$, we have

$$E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx.$$

When a is a nonrandom constant, and $g(x) = ax$, we have

$$E[aX] = \int_{-\infty}^{\infty} axf(x)dx = a\int_{-\infty}^{\infty} xf(x)dx = aE[X].$$

Expected Value

It can also be shown that

$$E[X + a] = E[X] + a.$$

(Add something to a random variable, and the expected value of the variable will change by that amount.)

For example, if T is the boiling point of a liquid which is subject to random fluctuations in air pressure and with mean $E[T] = 100°C$, the expected boiling point of the temperature measurements if measured in Kelvin units is $E[T + 273] = E[T] + 273 = 373K$.

Expected Value

When $g(x) = x^2$, and the probability density function is as above, we have

$$E[X^2] = \int_a^b \frac{x^2}{(b-a)} dx = \frac{b^3 - a^3}{3(b-a)}$$

Variance

A feature of a distribution which is every bit as important as its location is its *scale*, or a measure of the degree of variability of the distribution.

The variance (or its square root, the standard deviation) is one way to measure the variability of a random variable.

Denoting the mean of X by μ, we have

$$V(X) = E[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 f(x) dx.$$

An algebraically equivalent expression is

$$V(X) = E[X^2] - \mu^2.$$

Variance

For the uniform distribution $f_U(x)$, the variance is

$$V(X) = \frac{(b-a)^2}{12}.$$

A small value of $V(X)$ implies that there is more certainty about the value of X; it will tend to take values close to μ when $V(X)$ is very small.

The distribution will be more spread out when $V(X)$ is large. (i.e. when a and b are farther apart)

Variance

The standard deviation is the square root of the variance. Both quantities summarize the spread or variability in a probability distribution. Note also that

$$\text{Var}(aX) = a^2\text{Var}(X) \tag{2}$$

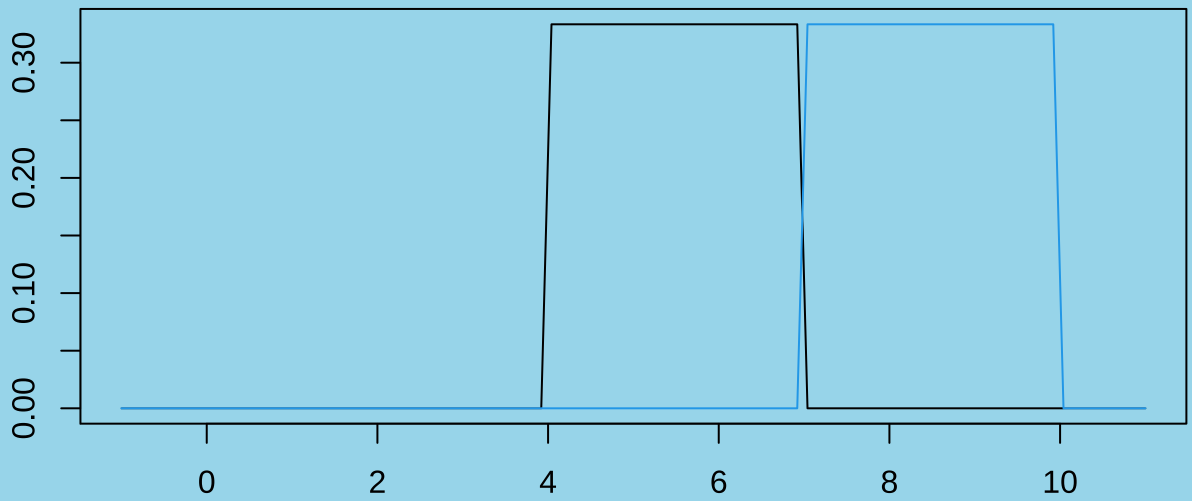
for any nonrandom constant a, and

$$\text{Var}(X + a) = \text{Var}(X). \tag{3}$$

In other words, the standard deviation of X is multiplied by a when X is. And the spread of the distribution doesn't change if it is only shifted by an amount a.

Example

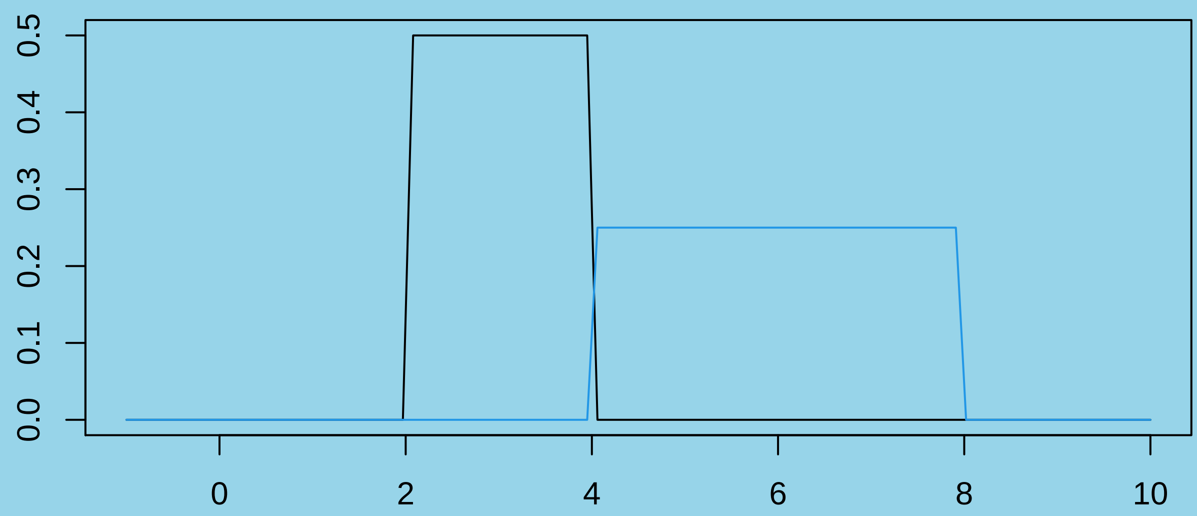
X and $X + 3$



No change in range of probable values in the distribution after adding 3.

Example

X and $2X$



The distribution becomes much more spread out after multiplying by 2.

Calculating the Mean and Variance from a Sample

When confronted with a sample of measurements $x_1, x_2, \ldots, x_n$, we can calculate the *sample mean* by taking the average of the sample values:

$$\bar{x} = \frac{1}{n} \sum_{j=1}^{n} x_j.$$

The *sample variance* is calculated as

$$s^2 = \frac{1}{n-1} \sum_{j=1}^{n} (x_j - \bar{x})^2.$$

The *sample standard deviation* is the square root of this: s.

Calculating the Mean and Variance from a Sample

```r
unifSample <- runif(50, 3,  7)
```

For the sample contained in `unifSample`, the sample mean, sample variance, and sample standard deviation are, respectively,

```r
mean(unifSample)

## [1] 5.109

var(unifSample);  sd(unifSample)

## [1] 1.428
## [1] 1.195
```

where we have also demonstrated how the calculations could be carried out in R.

Simulation of Other Random Variables

Discrete Random Variables

- **Bernoulli**
- **Binomial**
- **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] 0.7125

U < 0.2   # test the truth of U < 0.2 --> student's outcome

## [1] FALSE
```

Too bad, so sad. The student guessed wrong.

Simulating Guess Outcomes on a Multiple Choice Test

The student guesses at another question:

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

## [1] 0.7214

U < 0.2   # test the truth of U < 0.2 --> student's outcome

## [1] FALSE
```

Too bad, so sad. The student guessed wrong again.

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 FALSE FALSE
## [10]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [19] 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 $1 - p$.

The expected value of a Bernoulli random variable is p.

Its theoretical variance is $p(1 - p)$. (Standard deviation is $\sqrt{p(1 - p)}$).

In the above example, a student would expect to guess correctly 20% of the time; our simulated student was a little bit lucky, obtaining a mark of 30%.

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 $p = 0.2$.

$X \sim \textbf{bin}(20, 0.2)$.

Binomial Random Variables

The mean or expected of a binomial random variable is mp and the variance is $mp(1-p)$. (The standard deviation is $\sqrt{mp(1-p)}$.)

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 a 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] 0.2344
```

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] 0.9133
```

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] 0.1958
```

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] 0.002595
```

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

We can simulate a bin(m, p) random variate, by simulating m Bernoulli (p) variates and and adding them up.

In R, the rbinom() function can be used to generate n 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] 6 5 0 6 5 6 5 6 3 4 4 3
```

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] 0.5796 0.2264 0.5171 0.8905 0.6828 0.5440 0.2531 0.4658
##  [9] 0.8805 0.2664 0.6815 0.3056
```

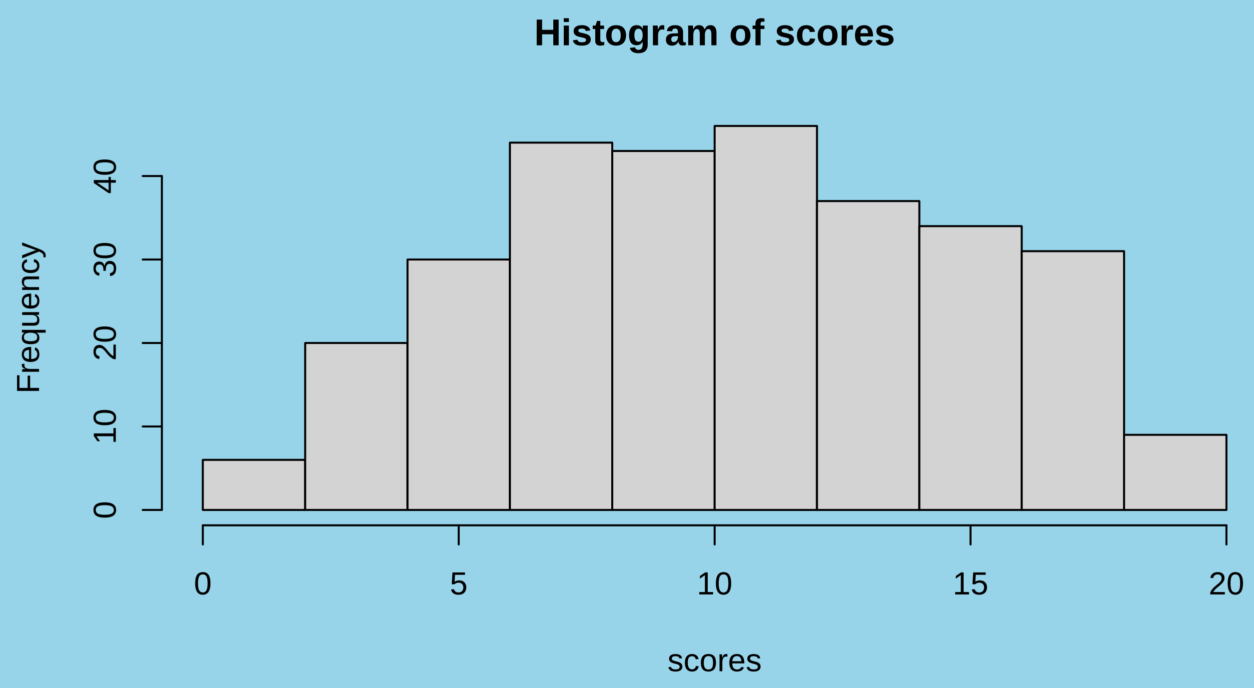
Simulating 12 different students’ perfomances after writing a multiple choice test with 20 questions:

```
rbinom(12, 20, U)

##  [1] 12  5  9 20 11 11  7  6 17  5 13  7
```

Simulating a Larger Class

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

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



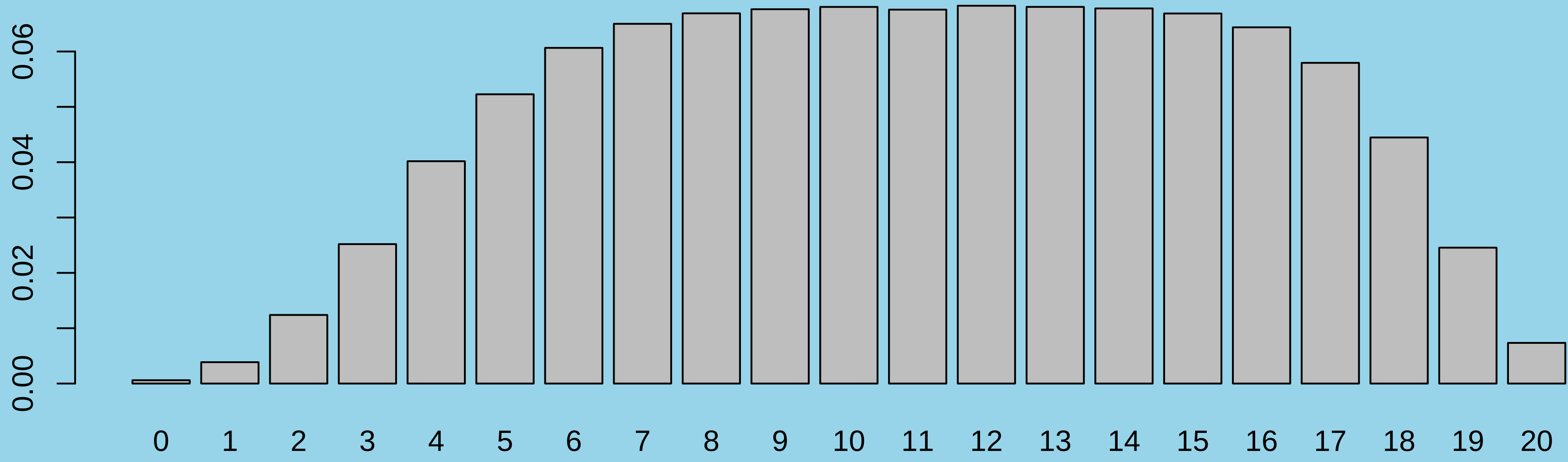
Using Simulation to Visualize a New Distribution

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:

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

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

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

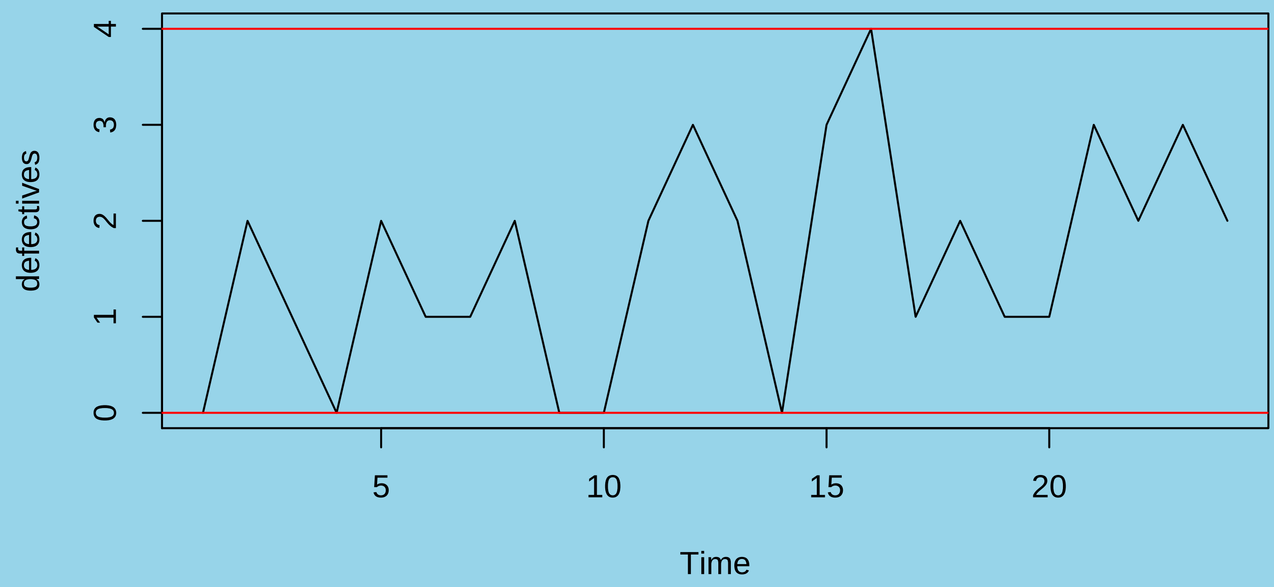
```
## [1] FALSE
```

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 1 0 1 1 0 1 3 3 0 0 1 4 0 5 2 3 1 2 1 2 2 2 1
```

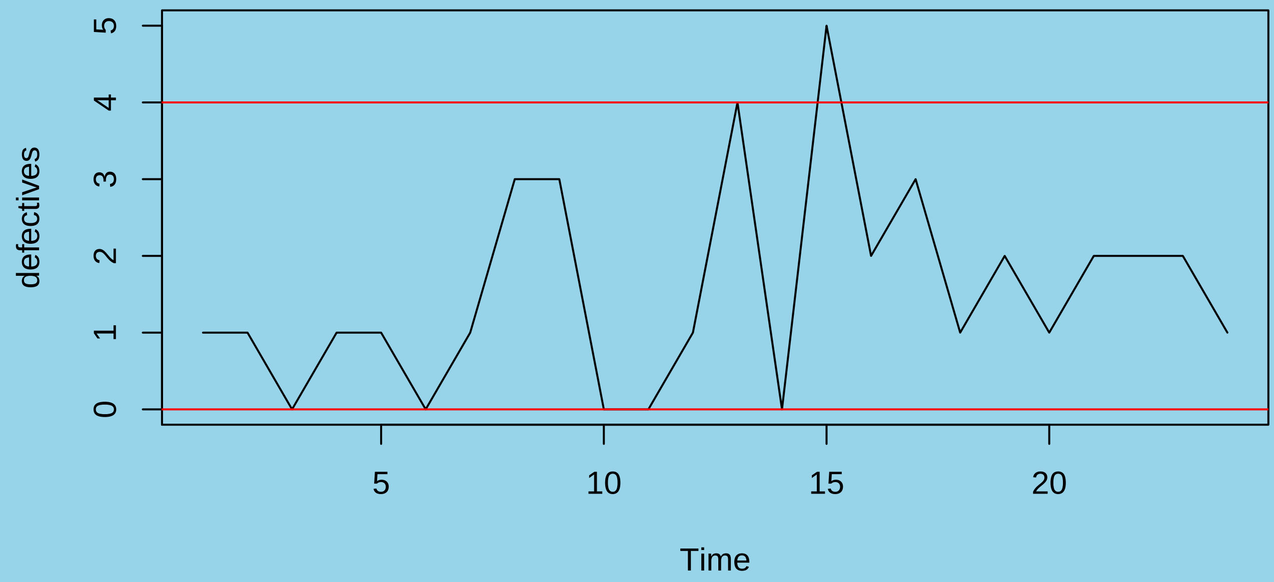
```
any(defectives > 4)  # any() asks if any of its arguments are TRUE
```

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

The out of control condition is detected.

Visualizing the Result

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



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 λ.

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 λ.

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] 0.01264
```

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

Poisson Probabilities and Pseudorandom Numbers

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

```
ppois(x, lambda)
```

We can 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.

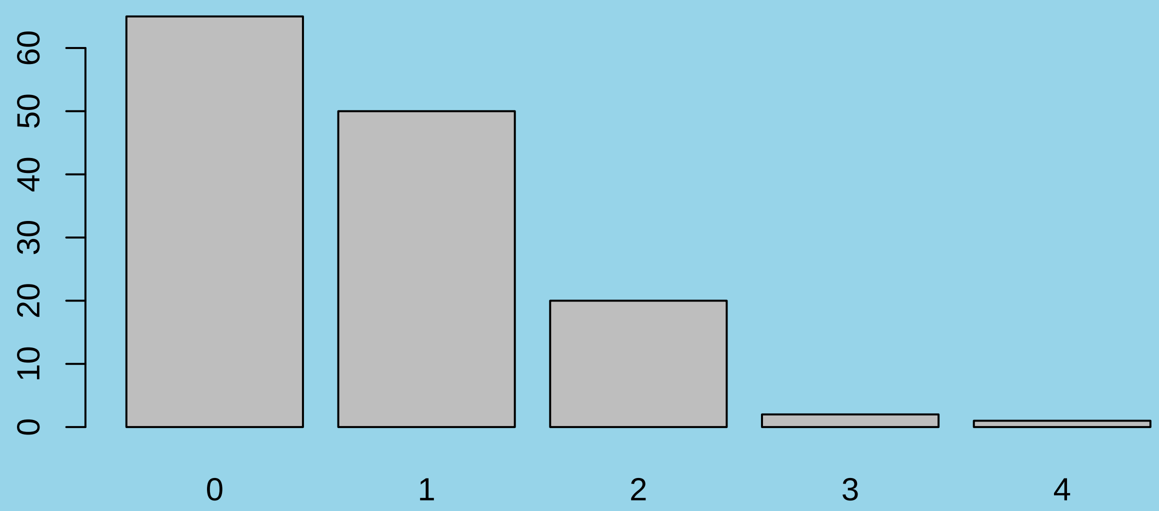
```r
N <- rpois(138,  0.6)
```

Calculate the mean and variance of N.

```r
mean(N)

## [1] 0.7246

var(N)

## [1] 0.6681
```

Rain Events - Poisson Distribution Example

```
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 rth success occurs.

If X is a negative binomial random variable, then

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

and

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

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 rth '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 needed

## [1] 0.0734
```

We can find the probability that it takes 6 or more trials using `pnbinom` as in

```
1 - pnbinom(5, 2, 0.2)

## [1] 0.5767
```

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.

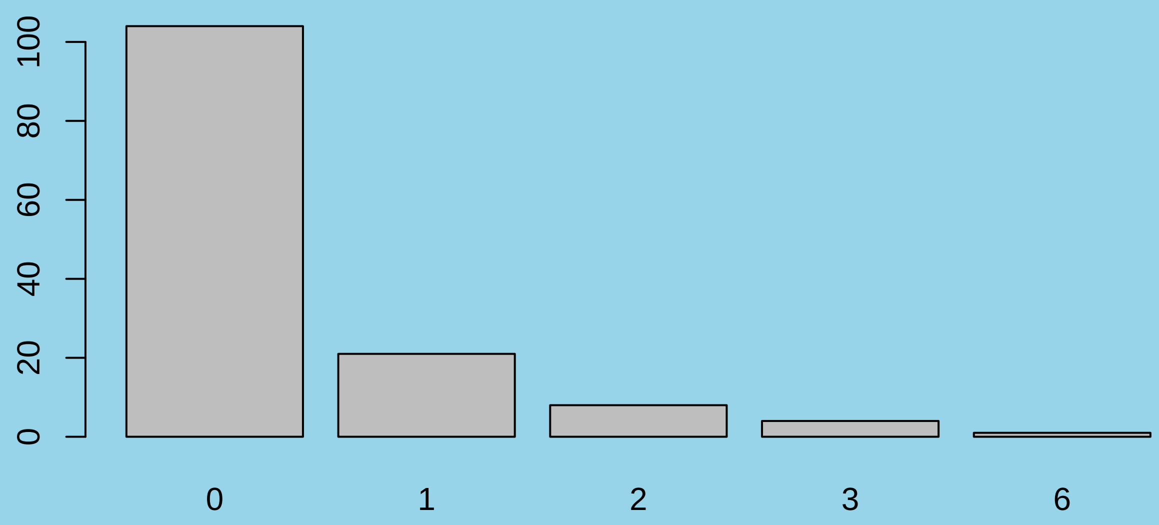
```
N <- rnbinom(138,  0.52, 0.57)
```

Calculate the mean and variance of N.

```
mean(N)

## [1] 0.3986

var(N)

## [1] 0.7524
```

Rain Events - Negative Binomial Example

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



What to Take Away from this Lecture

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, since they incorporate a mechanism for taking clustering into account.

What to Take Away from this Lecture

Be able to calculate:

- **theoretical means, variances and standard deviations for uniform, Bernoulli, binomial, and Poisson random variables.**
- **means, variances, and standard deviations for any kind of random variable using simulation, and understand that these are (good) estimates, as long as the simulation sample size is large.**
- **theoretical probabilities for uniform, Bernoulli, binomial, Poisson and negative binomial random variables, using `d*()` and `p*()`.**
- **estimated probabilities for any random variable using `r*()` appropriately, together with `mean()` and appropriate relational operator(s). e.g. `mean(runif(100000) < 0.25)`.**

Today's Lecture was Brought to You By the Letter R

Random Variables:

```
dunif()     # pdf for uniform
punif()     # cdf for uniform
runif()     # rng for uniform

dbinom      # pdf for binomial
pbinom      # cdf for binomial
rbinom      # rng for binomial

dpois       # pdf for Poisson
ppois       # cdf for Poisson
rpois       # rng for Poisson

dnbinom     # pdf for negative binomial
pnbinom     # cdf for negative binomial
rnbinom     # rng for negative binomial
```

Today's Lecture was Brought to You By the Letter R

Graphics:

```
hist()
abline()  # draws lines
    abline(h = a)  # draws horizontal line at y = a
    abline(v = b)  # draws vertical line at x = b
    abline(a, b) # draws line of slope b and intercept a
    abline(h = a, col="red") # draws red horizontal line
barplot(x) # bargraph with bar heights equal to x values
ts.plot(x) # plot of x against corresponding indices
  # (time series plot)
```

Today’s Lecture was Brought to You By the Letter R

MIscellanenous:

```
U[a:b]  # extracts elements from array U at indices a to b
U < a   # tests which elements of U are less than a

TRUE + TRUE #  = 2  (logical is coerced to numeric)
FALSE + FALSE # = 0

mean(x)  # average of elements in x
var(x)   # variance of elements in x
sd(x)    # standard deviation of x

set.seed(13323) # sets RNG seed for remainder of R session
any(x)   # tests whether elements of x (logical) are TRUE
```