STAT340 Lecture 01: Random Variables

Brian Powers

9/4/2024

Overview

The most fundamental object in statistics: random variables

Use random variables to model how our data came to be.

We will first introduce the idea of a random variable (and its associated distribution) and review probability theory.

Then, we will walk through a number of different basic distributions and discuss the kinds of data for which these different models are appropriate.

Learning objectives

After this lesson, you will be able to

- Explain what a random variable is
- Identify appropriate random variables for modeling different real-world events and explain why one choice might be better or worse than another
- Combine random variables to build simple models of real-world phenomena
- Compute the probabilities of simple events under different probability distributions using R.

What is a random variable?

$\boxed{X}$

Consider the following quantities/events:

- Whether or not a coin flip comes up heads or tails.
- How many people in the treatment group of a vaccine trial are hospitalized.
- The water level measured in Lake Mendota on a given day.
- How many customers arrive at a store between 2pm and 3pm today.
- How many days between installing a lightbulb and when it burns out.

All of these are examples of events that we might reasonably model according to different *random variables*.

Informal definition of a random variable: a (random) number X about which we can compute quantities of the form $\Pr[X \in S]$, where S is a set.

Aside: probability refresher I

set is unordered
no repeats in the {...}

We have a set of possible *outcomes*, usually denoted Ω.

finite

- when we flip a coin, it can land either heads (H) or tails (T), so the outcomes are $\Omega = \{H, T\}$.
- When we roll a six-sided die, there are six possible outcomes, $\Omega = \{1, 2, 3, 4, 5, 6\}$.
- In other settings, the outcomes might be an infinite set.
  - Ex: if we measure the depth of Lake Mendota, the outcome may be any positive real number (theoretically)

denoted individually

Usually Ω will is discrete (e.g., $\{1, 2, \dots\}$) or continuous (e.g., $[0, 1]$). We call the associated random variable discrete or continuous, respectively.

A subset $E \subseteq \Omega$ of the outcome space is called an *event*.

outcomes are numerable / countable

roll a 6 sided die

event:
"roll an odd #"
$E = \{1, 3, 5\}$

Aside: probability refresher II

A *probability* is a function that maps events to numbers, with the properties that

no outcomes in common = disjoint
= Mutually exclusive

- $\Pr[E] \in [0,1]$ for all events E
- $\Pr[\Omega] = 1$
- For $E_1, E_2 \in \Omega$ with $\boxed{E_1 \cap E_2 = \emptyset}$, $\Pr[E_1 \overset{\text{or}}{\cup} E_2] = \Pr[E_1] + \Pr[E_2]$

Two events E_1 and E_2 are *independent* if $\Pr[E_1 \cap E_2] = \Pr[E_1] \Pr[E_2]$.

Two random variables X and Y are independent if for all sets S_1, S_2, we have $\Pr[X \in S_1 \ \& \ Y \in S_2] = \Pr[X \in S_1] \Pr[Y \in S_2]$.

Roughly speaking, two random variables are independent if learning information about one of them doesn't tell you anything about the other.

- For example, if each of us flips a coin, it is reasonable to model them as being independent.
- Learning whether my coin landed heads or tails doesn't tell us anything about your coin.

Example: Coin flipping

Consider a coin toss, in which the possible outcomes are $\Omega = \{H, T\}$.

This is a discrete random experiment, but beacuse the outcomes are not numeric it is not a random variable. If, however we define X to be 1 when we flip an H and 0 when we flip T, then we have defined a *discrete random variable*, because the outcome set $\{0, 1\}$ is discrete (and numeric).

If we have a fair coin, then it is sensible that $\Pr[X = 1] = \Pr[X = 0] = 1/2$.

Exercise (optional): verify that this probability satisfies the above properties!

We will see in a moment that this is a special case of a Bernoulli random variable, which you are probably already familiar with.

Example: Six-sided die

If we roll a die, the outcome space is $\Omega = \{1, 2, 3, 4, 5, 6\}$, and the events are all the subsets of this six-element set.

So, for example, we can talk about the event that we roll an odd number $E_{\text{odd}} = \{1, 3, 5\}$ or the event that we roll a number larger than 4, $E_{>4} = \{5, 6\}$.

Example: Human heights

We pick a random person and measure their height in, say, centimeters. **What is the outcome space?**

- One option: the outcome space is the set of positive reals, in which case this is a *continuous random variable*.
- Alternatively, we could assume that the outcome space is the set of all real numbers.

Highlight: the importance of specifying our assumptions and the outcome space we are working with in a particular problem.

A note on models, assumptions and approximations

Note that we are already making an approximation– our outcome sets aren't really exhaustive, here.

- When you toss a coin, there are possible outcomes other than heads and tails: it is technically possible to land on the edge.
- Similarly, perhaps the die lands on its edge.

Human heights:

- We can only measure a height to some finite precision (say, two decimal places), so it is a bit silly to take the outcome space to be the real numbers.
- After all, if we can only measure a height to two decimal places, then there is no way to ever obtain the event, "height is 160.3333333... centimeters".

Good to be aware of these approximations - but they usually won't bother us.

Random variables

A random variable is specified by a probability:

A random variable X is specified by an outcome set Ω and a function that specifies probabilities of the form $\Pr[X \in E]$ where $E \subseteq \Omega$ is an event.

Let's look at some commonly-used random variables and real-world phenomena to which they are well-suited.

Bernoulli

A Bernoulli random variable has outcome set $\Omega = \{\overset{\text{"failure"}}{0}, \overset{\text{"success"}}{1}\}$.

To specify a probability on this set it is enough for us to specify $\Pr[\{0\}]$ and $\Pr[\{1\}]$.

Typically, we do this by specifying the *success probability* $p = \Pr[\{1\}] \in [0, 1]$. Once we have done this, it is immediate that (check!) $\underline{\Pr[\{0\}]} = 1 - p$.

Note that we can check that this gives us a probability by verifying that it sums to 1:

$$\Pr[\Omega] = \Pr[\{0\} \cup \{1\}] = \Pr[\{0\}] + \Pr[\{1\}] = 1 - p + p = 1.$$

sometimes

π instead of

"p"

Bernoulli examples

Bernoulli random variables are commonly used to model “yes or no” events. That is, events of the form “whether or not event A happens”. Common examples:

- Coin flips
- whether or not a person gets sick with a disease
- whether or not a team wins a game.

If X is a Bernoulli random variable with probability of success p, then we write $\boxed{X \sim \text{Bernoulli}(p).}$

We read this as something like “X is distributed as Bernoulli p”.

Binomial

What if we flip many coins, all with the same probability of coming up heads?

Then the total number of heads is distributed as a *binomial* random variable.

In particular, we describe a binomial distribution by specifying two *parameters*:

1. the number of trials (i.e., coins flipped) n, often called the *size* parameter and
2. the success probability p (i.e., the probability that an individual coin lands heads).

Often we will write Binomial(n, p) to denote this distribution.

So if X is a Binomial random variable with n trials and success probability p, we write $X \sim \text{Binomial}(n, p)$.

If $X \sim \text{Binomial}(1, p)$ then $X \sim \text{Bernoulli}(p)$

Example: modeling COVID-19

In a population of 250,000 people (approximately the population of Madison), we may imagine that each person has some probability p of becoming seriously ill with COVID-19.

Then, in a sense, the total number of people in Madison who become seriously ill with COVID-19 is like the total number of probability-p coin flips that land heads when we flip $250,000$ coins.

We might then model the number of COVID-19 patients by a binomial random variable with $n = 250,000$ and $p = 0.01$ (just to be clear, we are completely making up this choice of p here, just for the sake of example!).

Generating random binomial RVs I

We can generate binomial random variables using the `rbinom` function. Think “`r` for random”.

`rbinom` takes three arguments. - The first is the number of random variables we want to generate (confusingly, this is called n in the R docs). - The size argument specifies the number of coins to flip, i.e., n in our notation above (I know! Confusing!) - The prob argument specifies the probability that one coin lands heads, i.e., p in our notation above.

The code below produces a random number from $\{0,1,2,\ldots,10\}$, with 2,3,4 being most common (because $np = 3$ is the expected value; we'll come back to this!)

```
rbinom(1, size=10, prob=0.3)
## [1] 5
```

Generating random binomial RVs II

If we repeat the experiment a few times, we get different random values.

```
rbinom(1, size=10, prob=0.3);
## [1] 0
rbinom(1, size=10, prob=0.3);
## [1] 1
rbinom(1, size=10, prob=0.3);
## [1] 3
```

We can also use the binomial to generate Bernoulli random variables, by setting the `size` argument to 1 (i.e., flip 1 coin):

```
rbinom(1, size=1, prob=0.5); ## 1 is "heads", 0 is "tails"
## [1] 0
```

Generating random binomial RVs III

Important note: if you read the R documentation, there is a possible notational confusion waiting for you, alluded to in the comments of the code above. The signature of the `rbinom` function is given as `rbinom(n, size, prob)`. Based on the Binomial(n, p) notation we used above, you might expect that `n` in the `rbinom` function is the number of coins and `prob` is the success probability. Unfortunately, that isn't quite right. `n` is the number of Binomial random variables to generate. `size` specifies the size parameter (n in our math notation above).

Generating random binomial RVs IV

Compare the following:

```
rbinom(n=3, size=10, prob=0.5); ## 3 draws from a Binomial(10,0.5)
## [1] 7 5 4
```

```
rbinom(n=10, size=3, prob=0.5); ## 10 draws from a Binomial(3,0.5)
##  [1] 2 1 3 3 1 2 2 2 2 2
```

All of the R functions for generating random variables take `n` as the number of draws from the distribution. This is in keeping with the convention in most of probability and statistics that *n* is a sample size. Unfortunately, this is just one of those places where two different notational conventions collide. It's unfortunate that it arises in such a common and early-stage part of R!

Geometric

failures before first success

Let's consider a different coin-flipping experiment. We flip a coin repeatedly and we count how many flips it takes before it lands heads.

So perhaps we flip the coin and it comes up heads immediately, in which case we would count zero (because there were no flips before the one where the coin landed heads). If we flipped the coin and it came up heads for the first time on the fourth toss, then we would count three, and so on.

This game describes the geometric distribution.

Its behavior is controlled by a single parameter, the probability p of landing heads.

The geometric distribution is a natural model for "time to failure" experiments.

Geometric Example and generating values

For example, suppose we install a light bulb, and measure how many days until the lightbulb burns out (one such experiment has been ongoing for a very long time!.

We can generate random geometric random variables using the `rgeom` function:

```
# Generate one geometric random variable with p=0.5.
# Most likely outcomes: 0,1,2
rgeom(1, prob=0.5);
## [1] 0
```

Geometric PMF

The probability that a Geom(p) random variable X takes a particular value k ($k = 0, 1, 2, \ldots$) is given by $\boxed{\Pr[X = k] = (1-p)^k p.}$

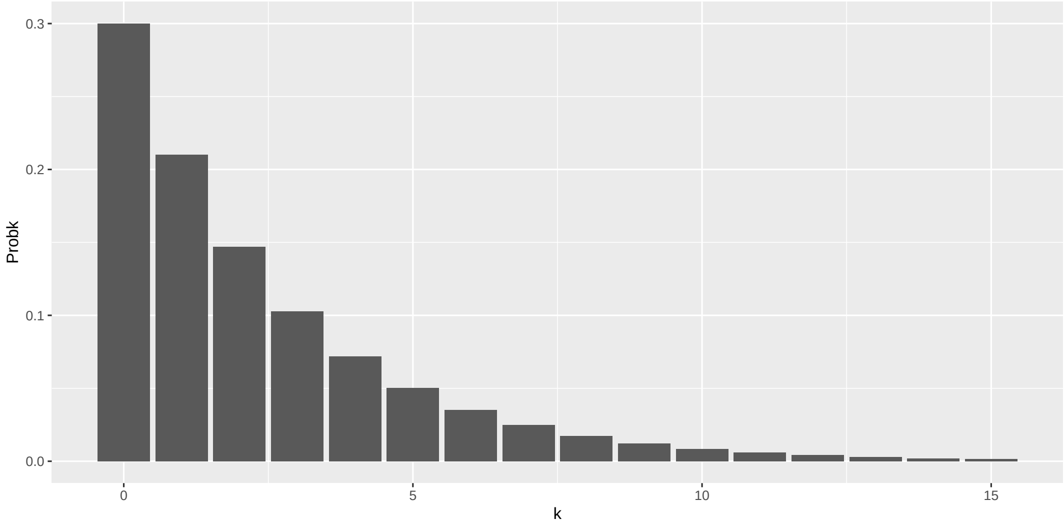
This is the *probability mass function* of the geometric distribution. It assigns to each value in the outcome space a nonegative probability. These nonnegative numbers sum to 1. (Some texts will refer to this function as a *distribution function*)

Let's plot this as a function of k:

$k = 3$: $\underbrace{F, F, F,}_{k}\ S$

```
library(ggplot2)
k <- seq(0,15); p <- 0.3;
df <- data.frame('k'=k,'Probk'=p*(1-p)^k );
(pp <- ggplot(df, aes(x=k, y=Probk) ) + geom_col())
```

Geometric PMF II



Geometric PMF III

Looking at the plot, we see that the geometric distribution puts most of its probability close to zero– the most likely outcomes are 0, then 1, then 2, and so on.

We plotted the distribution only up to $k = 15$, but a geometric random variable can, technically, take any non-negative integer as a value.

For any value of k, $Pr[X = k] = p(1 - p)^k$ is non-zero (as long as $0 < p < 1$).

So for any non-negative integer, there is a small but non-zero probability that a geometric random variable takes that integer as a value.

We say that the geometric random variable has *infinite support*.

The support of a discrete random variable is the set of values that have non-zero probability mass. A random variable has *infinite* support if this set is infinite.

Geometric PMF IV

Exercise: verify that this is a bona fide probability by checking that $\sum_{k=0}^{\infty} p(1-p)^k = 1$.

Note: some resources follow a slightly different convention, whereby a geometric random variable counts the total number of attempts (i.e., coinflips) before success, so the support is $\{1, 2, \dots\}$. Our discussion above follows the convention of most textbooks and research papers (and the convention followed by R– see `?rgeom`), but this is an important thing to be aware of!

Refresher: expectation

Before we continue with more random variables, let's take a pause to discuss one more important probability concept: expectation. You will hopefully recall from previous courses in probability and/or statistics the notion of expectation of a random variable.

Expectation: long-run averages

The expectation of a random variable X, which we write $\mathbb{E}X$, is the "long-run average" of the random variable.

What we would see on average if we observed many realizations of X.

That is, we observe $\underline{X_1, X_2, \ldots, X_n,}$ and consider their average, $\bar{X} = n^{-1} \sum_{i=1}^{n} X_i$.

Expected value and the Law of Large Numbers (first peek)

The *law of large numbers* (LLN) states that in a certain sense, as n gets large, $\bar{X}$ gets very close to $\mathbb{E}X$.

We would *like* to say something like

$$\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} X_i = \mathbb{E}X.$$

But $n^{-1} \sum_i X_i$ is a random sum, so how can we take a limit?

Roughly speaking, for n large, with **high probability**, $\bar{X}$ is **close** to $\mathbb{E}$.

Expectation: formal definition

More formally, if X is a discrete random variable, we define its expectation to be

$$\mathbb{E}X = \sum_{\underline{k}} k \cdot \underline{\Pr[X = k]},$$

weighted average

where the sum is over all k such that $\Pr[X = k] > 0$.

- Note that this set could be finite or infinite.
- If the set is infinite, the sum might not converge, in which case we say that the expectation is either infinite or doesn't exist. But that won't be an issue this semester.

Question: can you see how this definition is indeed like the "average behavior" of X?

Exercise: Expectation of Bernoulli

Compute the expectation of a Bernoulli random variable with success probability p. What about a Binomial(n, p) random variable? **Hint:** the expectation of a sum of RVs is the sum of their expectations. Write the Binomial RV as a sum of Bernoullis.

$X \sim \text{Bern}(p)$

x	$\Pr(X = x)$
0	$1-p$
1	p

$$E(X) = 0(1-p) + 1(p) = p$$

LLN Important take-away

The law of large numbers says that if we take the average of a bunch of independent RVs, the average will be close to the expected value.

- Sometimes it's hard to compute the expected value exactly (e.g., because the math is hard– not all sums are nice!)
- This is where Monte Carlo methods come in– instead of trying to compute the expectation exactly, we just generate lots of RVs and take their average!
- If we generate enough RVs, the LLN says we can get as close as we want.
- We'll have lots to say about this in our lectures on Monte Carlo methods next week.

Poisson I

Suppose we are going fishing on lake Mendota, and we want to model how many fish we catch in an hour.

A common choice for this situation is the *Poisson* distribution.

The Poisson distribution is a common choice for modeling "arrivals" or other events that happen per unit time. Common examples include

- customers arriving to a store on a given day
- calls to a phone line between 2pm and 3pm
- photons or other particles hitting a detector during an experiment
- cars arriving at an intersection

Poisson II

$0! = 1$
$1! = 1$

The Poisson distribution has probability mass function

$$\Pr[X = k] = \frac{\boxed{\lambda}^k e^{-\boxed{\lambda}}}{k!}, \qquad \underline{k = 0, 1, 2, \ldots}$$

The parameter $\lambda > 0$, often called the "rate parameter", controls the average behavior of the random variable– larger choices of λ mean that the resulting random variable is larger, on average (we will make this statement more precise in a few lectures). That is, the larger λ is, the more arrivals happen per unit time– the larger λ is, the higher the rate!

Poisson III

We can generate Poisson random variables using `rpois`:

```
# Generate Poisson RV with lambda=10.5; most likely value is 10.
rpois(1, lambda=10.5);
## [1] 10
```

What if I want several random Poissons, instead of just one?

The `n` argument to `rpois` (and all the other random variable generation functions) specifies a number of variables to generate.

So, for example, to get ten random Poissons, we can write

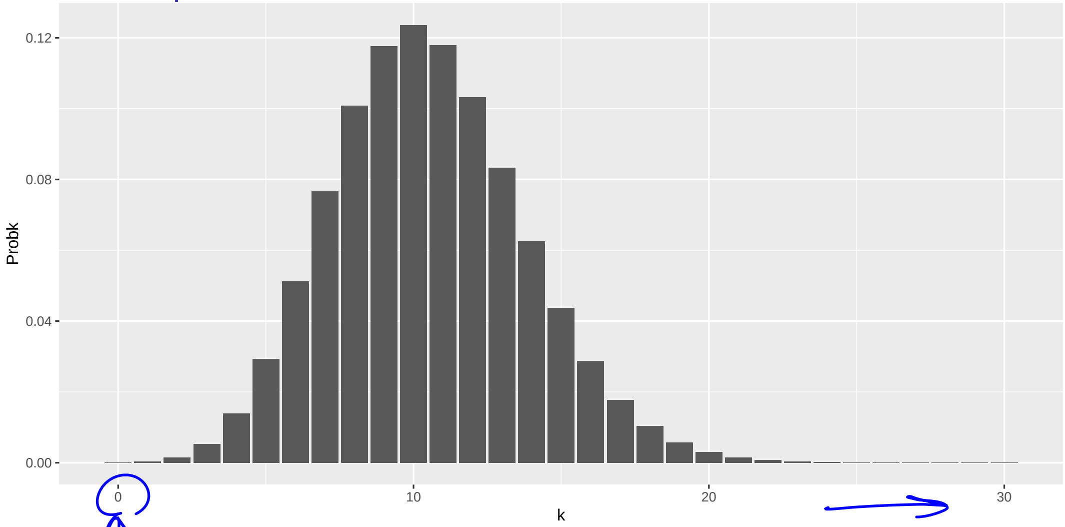
```
# Generate 10 Poisson RVs with the same parameter lambda=10.5
rpois(10, lambda=10.5);
##  [1]  6 14  9 15  8  8 12  3 10  8
```

Poisson PMF plot I

What does the probability mass function of the Poisson look like? Once again, the Poisson distribution has infinite support, since $\Pr[X = k] > 0$ for all $k = 0, 1, 2, \dots$ (check this!), but let's plot its first few values.

```r
k <- seq(0,30);
lambda <- 10.5; ## On average, we should get back the value 10.5,
df <- data.frame('k'=k,'Probk'=dpois(k, lambda) );
(pp <- ggplot(df, aes(x=k, y=Probk) ) + geom_col());
```

Poisson PMF plot II



Lower bound

continues forever.

Poisson PMF plot III

The function `dpois` above evaluates the Poisson probability mass function.

The R documentation calls this a density, which is technically correct.

Remember "`r` for random", "`d` for density". `rpois` generates random Poisson variables, `dpois` evaluates its probability mass function.

The Discrete uniform

The roll of a fair 6 sided die (or any *n*-sided die) can be modeled using a discrete uniform random variable. We will more often refer to the continuous uniform random variable, (which we'll talk about shortly), but it's worth mentioning that the roll of a die can be modeled easily.

In `R`, however, the discrete uniform is not a named distribution. Instead we would use the `sample` function. For example, suppose we wanted to simulate rolling a 6-sided die 100 times.

The sample function takes 3 important parameters.

- `x` : this can either be a vector of values to sample from or in this case simply the upper limit; R will automatically create a vector from 1 to this value to sample from
- `size` : how many samples to generate
- `replace`: this indicates whether we are able to re-sample the same value more than once. Careful, the default value is FALSE

Example: simulating die rolls

means we sample from $\{1,2,3,4,5,6\}$

```
(rolls <- sample(x=6, size=20, replace=TRUE))
##  [1] 5 2 5 6 3 5 2 2 1 2 3 2 3 3 5 2 1 1 2 3
```

$X \sim \text{Uniform}_D(a,b)$

a= lower bound

b= upper bound

$$\Pr(X=k) = \frac{1}{b-a\underbrace{+1}}$$

$k=a,a+1,\ldots,b$

$$E(X) = ? \quad \sum_k k \cdot P(X=k) = \frac{a+b}{2}$$

Aside: approximating one random variable with another

We can obtain the Poisson distribution from the binomial distribution.

Let's make two assumptions about our fish population:

- There are many fish in the lake. Let's call the number of fish N, which is a large number.
- For each fish, there is a *small* probability p that we catch it (the same probability for each fish, for the sake of simplicity)

If we let N get arbitrarily large ("infinite"; a limit like you remember from calculus) while p stays "small", the Binomial distribution comes to be equal to the Poisson distribution with rate Np.

For this reason, the Poisson is often a good approximation to the Bernoulli when N is large and p is small.

Illustration I

Just to illustrate, let's plot the density of the binomial with N really large and p really small, but chosen so that $Np = 10.5$ to match $\lambda = 10.5$ above.

```r
k <- seq(0,30); lambda <- 10.5;
N <- 1e6;
p <- lambda/N; # On average, we should get back the value lambda
poisprob <- dpois(k, lambda); # Vector of Poisson probabilities
binomprob <- dbinom( k, size=N, prob=p ); # Binomial probs.
```

We need a column in our data frame encoding which of the two distributions a number comes from. This isn't the only way to do this, but it is the easiest way to get things to play nice with the ggplot2 `facet_wrap`, which displays separate plots for different values in a particular column.

Illustration II

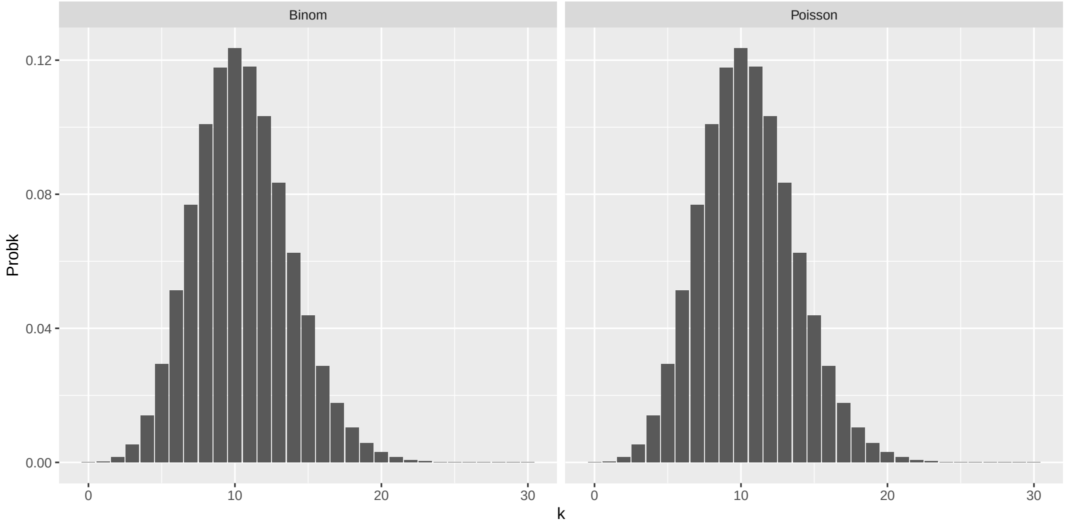
```r
dist <- c( rep('Poisson', length(k)), rep('Binom', length(k)) );

# Construct our data frame. Note that we have to repeat the k column,
# because our data frame is going to look like
# Distribution  k  Probk
# Poisson       0  dpois( 0, lambda )
# Poisson       1  dpois( 1, lambda )
# ...           ...
# Poisson       30 dpois( 30, lambda )
# Binomial      0  dbinom( 0, N, p )
# ...
# Binomial      30  dbinom( 30, N, p )
df <- data.frame('dist'=dist, 'k'=rep(k,2),
                 'Probk'=c(poisprob,binomprob) );
```

Illustration III

facet_wrap tells ggplot to create a separate plot for each group (i.e., value) in the dist column.

```
(pp <- ggplot(df, aes(x=k, y=Probk) ) + geom_col() + facet_wrap(~dist));
```

Illustration IV



Continuous random variables

So far we've seen a few different discrete random variables. Their set of possible values are discrete sets like $\{0, 1\}$ or $\{0, 1, 2, \dots\}$.

This is in contrast to *continuous* random variables, which take values in "continuous" sets like the interval $[0, 1]$ or the real like $\mathbb{R}$.

Discrete random variables have probability mass functions, like $\Pr[X = k] = p(1 - p)^k$, $k = 0, 1, 2, \dots$.

In contrast, continuous random variables have *probability density functions*, which we will usually write as $f(x)$ or $f(t)$.

These random variables are a little trickier to think about at first, because it doesn't make sense to ask about the probability that a continuous random variable takes a specific value. That is, $\Pr[X = k]$ doesn't really make sense when X is continuous (actually– in a precise sense this does make sense, but the probability is always zero; you'll see why below).

Let's see some examples.

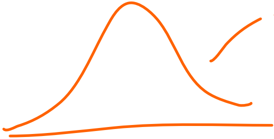
Normal

The normal or *Gaussian* (after Carl Friedrich Gauss) random variable is undoubtedly the most fundamental in all of statistics.

You have likely heard of it before both in your previous courses and just. . . well, everywhere, in the form of the famous *bell curve*.

The normal distribution really is everywhere, and there are good reasons for this, which we will return to in a few lectures (see here for a preview).

The normal distribution is controlled by two parameters: a mean $\mu \in \mathbb{R}$ and a variance $\sigma^2 > 0$.



this is the probability density function

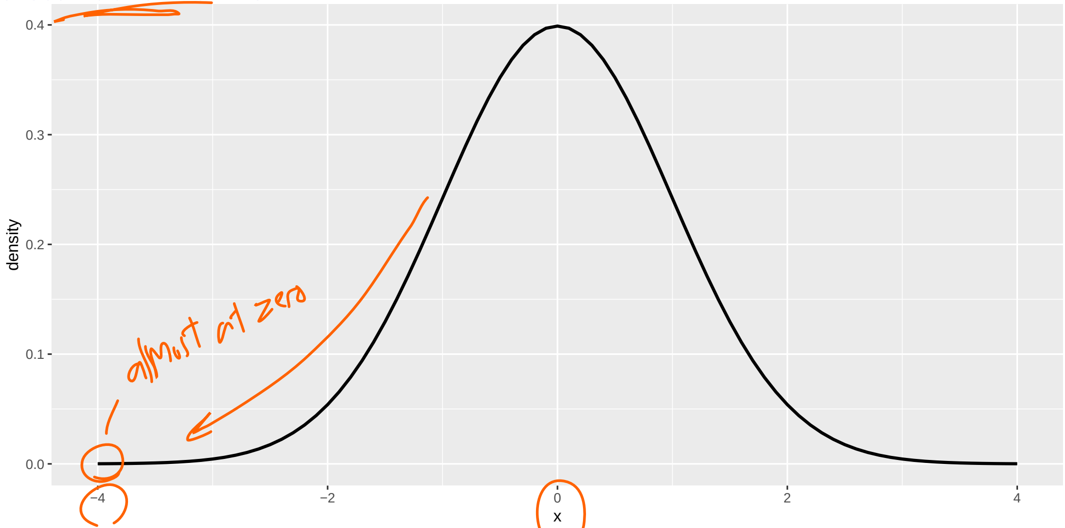
$\sigma \equiv$ standard deviation

The standard normal I

The *standard* normal has $\mu = 0$ and $\sigma^2 = 1$, and it “looks like” this:

```r
x <- seq(-4,4,0.1);
# eval density at x values, mean mu=0, standard deviation sigma=1.
f <- dnorm(x, 0,1);
df <- data.frame('x'=x, 'density'=f)
(pp <- ggplot( df, aes(x=x, y=density) ) + geom_line( linewidth=1))
```

The standard normal II



-4σ

$\sigma^2 = 1$

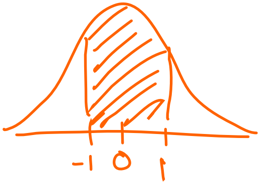
The standard normal III

This is a probability *density*, not a mass function, because to evaluate something like the probability that a normal random variable X falls between, say, -1 and 1, we have to integrate the area under this curve between these two endpoints.

This is why we refer to this as a density– recall from physics that integrating a density over a region (i.e., a volume) gives us a mass (compare to the discrete case, where we did call it a probability mass function).

Said another way, integrating the density over a region gives us the probability of that region.

Nonstandard normal distribution I



So if X is a normal random variable with mean μ and variance σ^2,

$$\underline{\Pr[-1 \leq X \leq 1]} = \underline{\int_{-1}^{1} f(t;\mu,\sigma^2)dt},$$

where $f(t;\mu,\sigma^2)$ is the density function of the normal distribution,

$$f(\underline{t};\mu,\sigma^2) = \underbrace{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(t-\mu)^2}{2\sigma^2}\right\}}.$$

The semicolon notation is to emphasize that the density is a function of t, but its shape depends on parameters μ and σ^2.

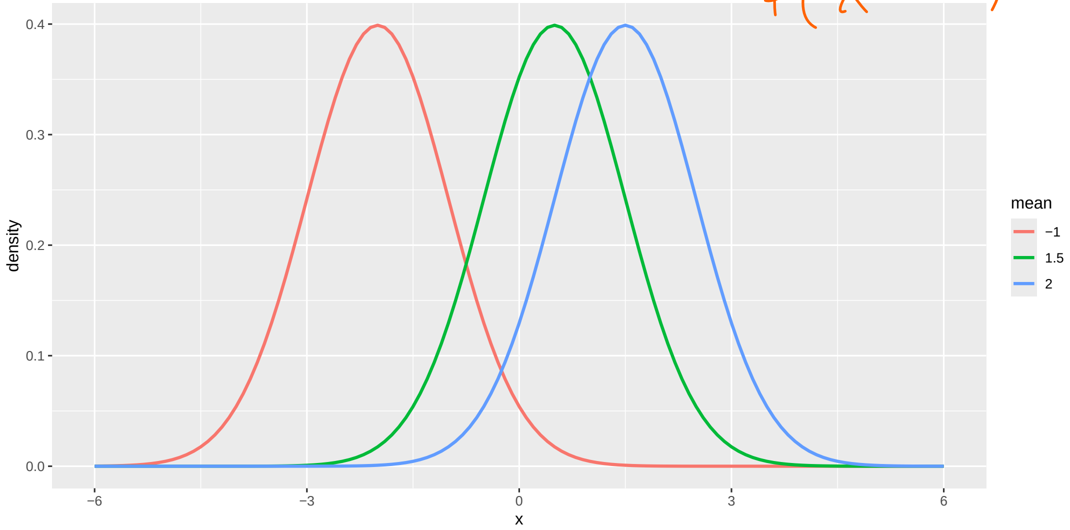
Changing $\underline{\mu}$ while keeping $\underline{\underline{\sigma^2}}$ fixed just shifts the distribution:

Nonstandard normal distribution II

```
x <- seq(-6,6,0.1); ## Some x values to evaluate at
f1 <- dnorm(x, -2,1); ## mean mu=-1, standard deviation sigma=1.
f2 <- dnorm(x, 0.5,1); ## mean mu=0.5, standard deviation sigma=1.
f3 <- dnorm(x, 1.5,1); ## mean mu=1.5, standard deviation sigma=1.
mu = c( rep(-1, length(x)), rep(1.5, length(x)), rep(2.0, length(x)));
df <- data.frame('x'=rep(x,3), 'density'=c(f1,f2,f3),
                 'mean'=as.factor(mu) )
pp <- ggplot( df, aes(x=x, y=density, color=mean) ) +
  geom_line( linewidth=1)
pp
```

Nonstandard normal distribution III

$P(X \in \Omega) =$



Nonstandard normal distribution IV

Changing σ changes the standard deviation (and hence also the variance, σ^2). Larger σ means higher variance, which means a "wider" distribution:

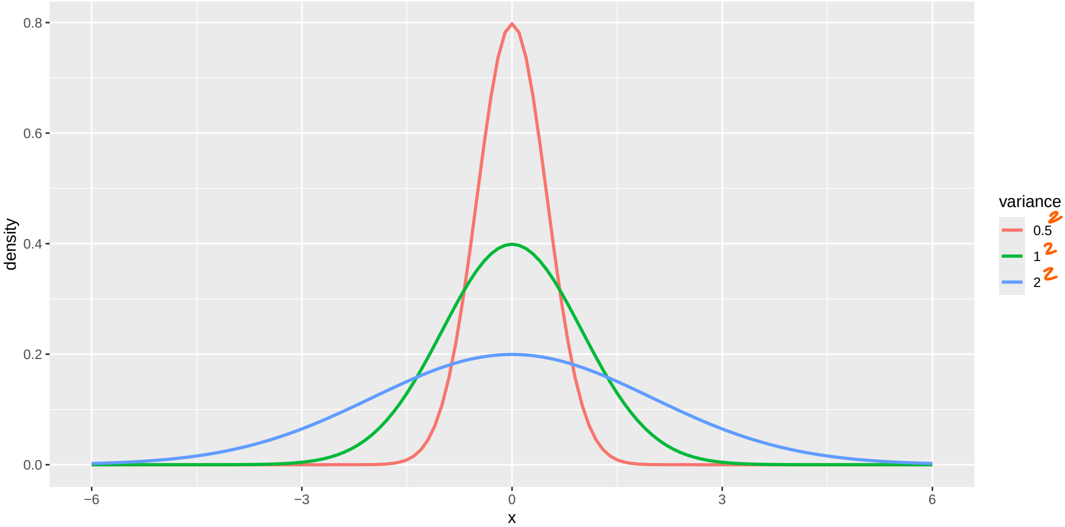
```
x <- seq(-6,6,0.1); ## Some x values to evaluate at
f1 <- dnorm(x, 0,0.5); ## mean 0, standard deviation sigma=1.
f2 <- dnorm(x, 0,1); ## mean 0, standard deviation sigma=2.
f3 <- dnorm(x, 0,2); ## mean 0, standard deviation sigma=3.
sigma2 = c(rep(0.5, length(x)), rep(1, length(x)), rep(2, length(x)));
df <- data.frame('x'=rep(x,3), 'density'=c(f1,f2,f3),
                 'variance'=as.factor(sigma2) )
pp <- ggplot( df, aes(x=x, y=density, color=variance) ) +
  geom_line( linewidth=1)
pp
```

.5

1

2

Nonstandard normal distribution V



Generating Normal random data I

We can generate normal random variables in R using the `rnorm` function. Note that we pass the standard deviation (sd), not the variance (sigma^2).

not variance

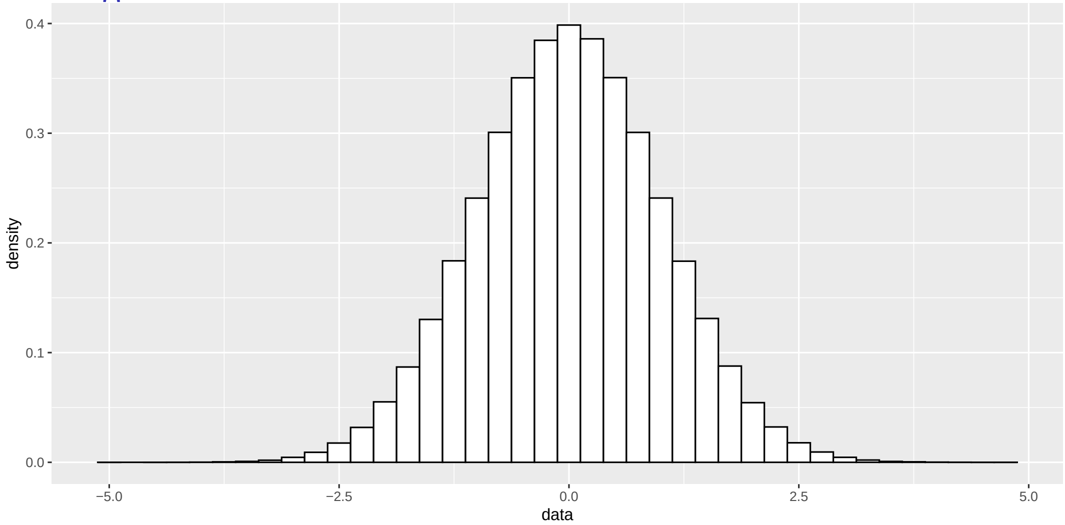
```
rnorm(1, mean=1, sd=2);
## [1] 0.7637703
# If we don't specify a mean and sd they default to 0 and 1
rnorm(1);
## [1] -0.1087678
```

Generating Normal random data II

Let's generate a bunch of normal RVs and plot their (normalized) histogram:

```
# Reminder: if mu, sigma unspecified, they default to mu=0, sigma=1
data <- rnorm(1e6); # 1e6 = 1.0x10^6 = one million standard normal RVs
pp <- ggplot( data.frame('x'=data ), aes(x=data) );
pp <- pp + geom_histogram( aes(y=after_stat(density)), binwidth=0.25,
                           color="black", fill="white");
pp
```

Generating Normal random data III



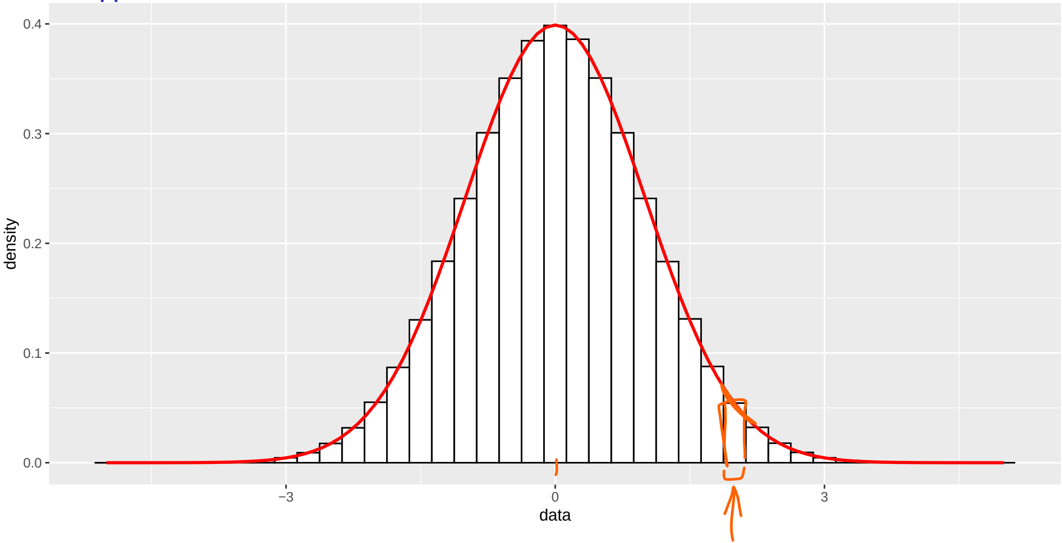
Generating Normal random data IV

Note that this is a normalized histogram– it is scaled so that the areas of the rectangles sums to 1, like a probability distribution.

Now, let's overlay the normal density function on this.

```
x <- seq(-5,5,0.1); f <- dnorm( x, mean=0, sd=1 );
df_dnorm <- data.frame('x'=x, 'f'=f);
pp <- pp + geom_line(data=df_dnorm, aes(x=x, y=f), linewidth=1,
                     color='red');
pp
```

Generating Normal random data V



2 to 2.25

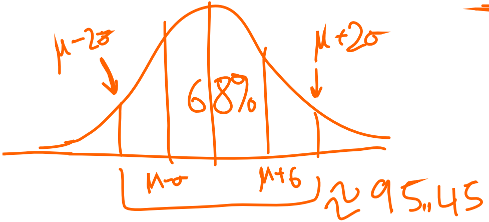
The Empirical Rule I

68 - 95 - 99.7

You will hopefully recall from any previous statistics courses that a normal random variable falls

- within one standard deviation of the mean (i.e., between -1 and 1 in the plot above) with probability about 0.68,
- within two standard deviations of the mean (i.e., between -2 and 2 in the plot above) with probability about 0.9545
- within three standard deviations of the mean (i.e., between -3 and 3 in the plot above) with probability about 0.997.

(Note - The empirical rule is a much stronger version of Chebychev's inequality. Look that up!)



$$P(\mu - k\sigma < X < \mu + k\sigma) \geq 1 - \frac{1}{k^2}$$ ☆

The Empirical Rule II

We can approximately check this by counting how many of our simulated values fall in these ranges. What fraction of these data points fell within one standard deviation of the mean?

```
sum( data > -1 & data < 1)/length(data); ## Should be about 0.68
## [1] 0.682094
```

```
sum( data > -2 & data < 2)/length(data); ## should be about 0.9545
## [1] 0.954485
```

```
sum( data > -3 & data < 3)/length(data); ## should be about 0.997
## [1] 0.99736
```

Of course, because the data is random, the proportions are not exactly equal to their predicted values, but they are quite close.

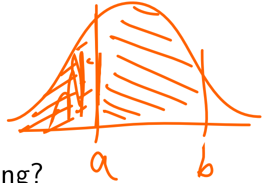
This is a nice illustration of the law of large numbers!

Computing probabilities I

Now, let's check this fact by using R to compute probabilities of the form $\Pr[a \leq X \leq b]$ for a normal RV X, where $a \geq b$. How can we compute this probability?

Let's start by writing something slightly different:

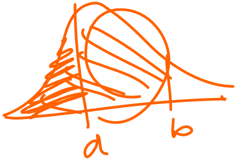
$$\underline{\Pr[X \leq b]} = \Pr[X \underline{\leq a]} + \underline{\Pr[a \leq X \leq b]}.$$



Question: why is this true? What "rule" of probability are we using?

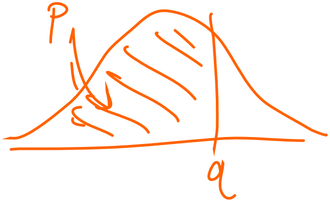
Now, let's rearrange that expression:

$$\Pr[a \leq X \leq b] = \underline{\Pr[X \leq b]} - \underline{\Pr[X \leq a]}.$$



How exactly does this help us? R makes it very easy to compute probabilities of the form $\Pr[X \leq a]$.

The Cumulative Distribution Function



The function

$$F_X(t) = \Pr[X \leq t]$$

is called the *cumulative distribution function* (CDF) of X.

We compute the CDF of the normal in R with `pnorm`. So, for example, to compute the probability that a standard normal random variable is ≤ 0, we can write

```
pnorm(0) ## Once again, mu, sigma default to 0,1.
## [1] 0.5
```

pnorm(q) = p

Aside/exercise:

You might enjoy spending some time thinking on why it is “obvious” that a normal random variable is less than or equal to its mean with probability one half. Remember (look at the density function above, or just look at our plots) that the normal density is symmetric about the mean μ. If you’re feeling up to it (warning: calculus ahead!), you can try computing the appropriate integrals to show that under the normal, $\Pr[X \leq \mu] = \Pr[X > \mu] = 1/2$.

Normal probability examples I

So suppose that we want to compute the probability that a standard normal random variable is less or equal to 1. This would be

```r
pnorm(1) ## again, recall the default behavior.
## [1] 0.8413447
```

And to compute the probability that we are within one standard deviation of the mean, we need to compute

$$\Pr[-1 \leq X \leq 1] = \Pr[X \leq 1] - \Pr[X \leq -1] = \texttt{ pnorm(1) - pnorm(-1)}$$

That is,

```r
pnorm(1)-pnorm(-1)
## [1] 0.6826895
```

Normal probability examples II

which matches our 0.68-.9545-0.997 rule so far. Let's check the other two.

```
pnorm(2)-pnorm(-2)
## [1] 0.9544997
```

```
pnorm(3)-pnorm(-3)
## [1] 0.9973002
```

Recap: RVs in R (so far)

So far, we have seen three functions for working with different distributions.

For the normal, we have `rnorm`, `dnorm` and `pnorm`, corresponding to random variable generation, the density function and cumulative distribution function (CDF), respectively.

Other RVs have similar patterns. For example, with the geometric random variable, we have `rgeom`, `dgeom` and `pgeom`, which work analogously to the normal case.

There is one more of these functions yet to come, but we aren't quite ready to introduce it.

Aside: Expectation for continuous random variables

Previously, we defined the expectation of a discrete random variable X to be

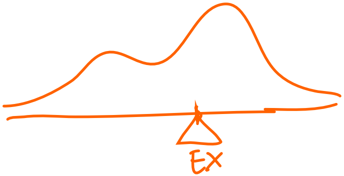
$$\boxed{\mathbb{E}X = \sum_k \boxed{k}\, \underline{\Pr[X = k]},}$$

with the summand k ranging over all allowable values of X.

When X is continuous how should we define the expectation?

Change the sum to an integral!

$$\mathbb{E}X = \int_{\underline{\Omega}} \underline{tf(t)dt},$$



where $f(t)$ is the density of X and Ω is the support.

Exercise: find the mean of the normal distribution

Check that the mean of a normal with mean μ and standard deviation σ is indeed μ. That is, check that

$$\int_{-\infty}^{\infty} \frac{t}{\sqrt{2\pi\sigma^2}} \exp\left\{ \frac{-(t-\mu)^2}{2\sigma^2} \right\} dt = \mu.$$

Hint: Make the substitution $u = t - \mu$ and use the facts that

1. $f(t) = e^{-(t-\mu)^2/2\sigma^2}/\sqrt{2\pi\sigma^2}$ integrates to 1 (because it is a probability density), and
2. $g(u) = u \exp\{-u^2/2\sigma^2\}$ is symmetric about zero (and thus integrates to zero).

Uniform

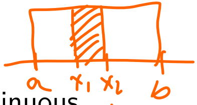
The uniform distribution is a continuous random variable whose density is constant on its outcome space.

That is, for continuous set $\Omega \subseteq \mathbb{R}$, the density is identically $f(t) = c$.

Remember that our probability has to sum to 1, but since we have a continuous support, this sum becomes an integral:

$$\int_\Omega f(t)dt = \int_\Omega c dt = c \int_\Omega 1 dt.$$

So to make the probability integrate to 1, we need $c = \int_\Omega 1 dt$.



$$P(x_1 \leq X \leq x_2)$$

$$\frac{x_2 - x_1}{b - a}$$

Uniform(0,1)

Most commonly, we take $\Omega = [0, 1]$, and call the resulting random variable "uniform 0-1", written $\mathrm{Unif}(0, 1)$.

The density function of $\mathrm{Unif}(0, 1)$ is then given by

$$f(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Exercise: check that this indeed integrates to 1, i.e., that it is a valid probability density. The support is $[0, 1]$, so we need to check that $\int_0^1 1 dt = 1$.

Applications of the Uniform RV

The most common application of uniform random variables is in simulations when we need to make a random decision.

For example, suppose we didn't have the Bernoulli distribution available to us for some reason, but we still wanted to generate random coin flips.

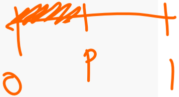
To generate a Bernoulli(p) random variable, we could first draw a uniform random variable $U \sim \text{Unif}(0, 1)$ and then output 1 (or "heads" or "true") if $U \le p$ and out put 0 (or "tails" or "false") otherwise.

default values

$$P(U \le p) = p$$

$$\frac{p-0}{1-0} = p$$

```r
p=.5
U <- runif(n=1, min=0, max=1)
U
## [1] 0.4477426
output <- ifelse(U <= p, "heads", "tails")
output
## [1] "heads"
```



Exponential

The exponential distribution is most commonly used to model "waiting times", like how long until the bus arrives.

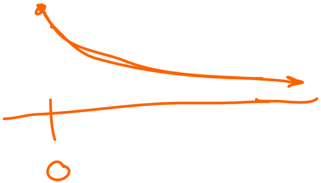
The exponential distribution is like the continuous version of the geometric distribution.

The exponential distribution is non-negative and is controlled by a single parameter $\lambda > 0$, called the *rate* (because larger λ means less time before the event, hence more events per unit time, i.e., a higher rate of events).

The density is given by

$\lambda e^{-\lambda t}$

$$f(t;\lambda) = \begin{cases} \lambda \exp\{-\lambda t\} & \text{if } t \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$



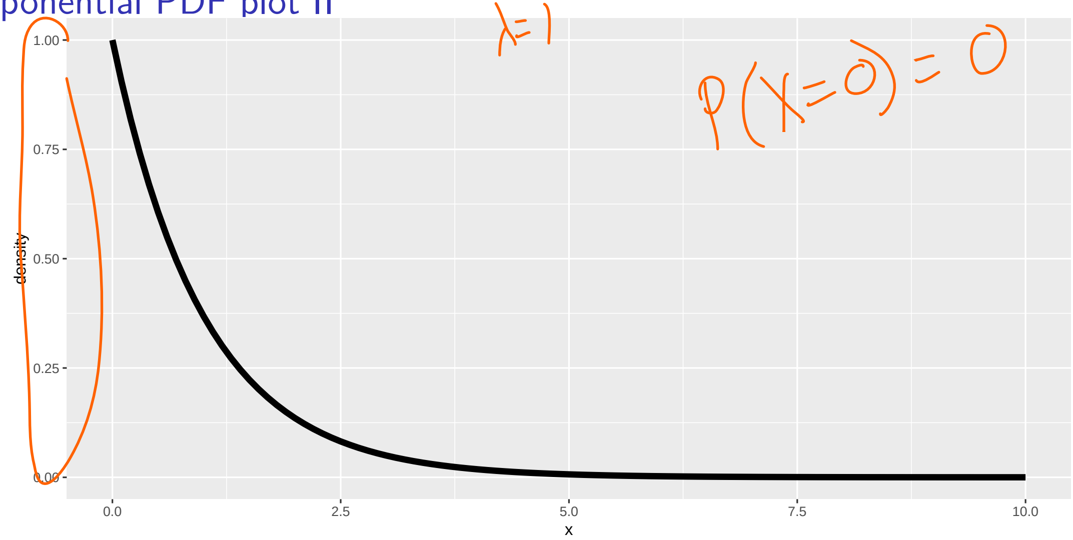
Exercise: check that this defines a probability distribution by checking that for any $\lambda > 0$, (1) $f(t;\lambda) \geq 0$ and (2) $\int_0^\infty f(t;\lambda)dt = 1$.

Exponential PDF plot I

Let's plot this density as a function of t for $\lambda = 1$.

```r
x <- seq(0, 10, 0.1); f = dexp(x, rate=1);
df_exp <- data.frame('x'=x, 'density'=f);
pp <- ggplot(df_exp, aes(x=x, y=density)) + geom_line( linewidth=2);
pp
```

Exponential PDF plot II



$\lambda = 1$

$P(X=0) = 0$

Looking at the density, we see that most of the probability is near zero.

Exponential PDF plot III

Roughly, it looks like the vast majority of the time, $X \sim \mathrm{Exp}(1)$ should be less than 5. Let's check!

```
data <- rexp(n=1e6, rate=1); # rexp to generate random exponentials.
# something like 99% of the generated points should be below 5.
sum( data < 5)/length(data);
## [1] 0.993421
```

Building Bigger Models

There are plenty more named distributions out there. See here.

Indeed, we can make all sorts of distributions– ultimately we just need to specify a support and a density or mass function, depending on whether we want a discrete or continuous variable.

More often, though, the processes out in the world that we want to model require that we build more complicated models from simple ones.

Social network models I

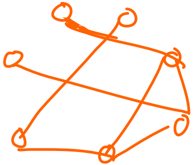
Several professors in the statistics department study networks. A common example of these are social networks (e.g., Facebook and twitter).

These data usually take the form of a list specifying who is friends with whom.

In most models for social networks, each pair of people i and j become friends, independently, with some probability p_{ij}.

Often, these probabilities depend on other random variables.

- For example, we might have a (random) number for every person that describes how much that person likes sports, and pairs of people who both have a high "sports score" are more likely to be friends (i.e., p_{ij} is higher).



Social network models II

In this way, we combine different elementary random variables (e.g., Bernoullis and normals) to yield more complicated distributions (say, a probability distribution over "possible social networks").

Having obtained a network from Twitter or Facebook data, we apply all sorts of different functions to it that describe the network's structure (e.g., compute shortest paths or eigenvectors, if you know what those are).

It is really difficult to do the math out exactly to determine analytically how these different functions behave.

Instead, we often appeal to Monte Carlo methods, which we will discuss in the next lecture.

Binomial asset pricing model I

Another way to create new random variables is to take a function of another random variable. This is what happens in the pricing of “options” in finance.

Suppose that X is the price of stock XYZ one month from today.

An “option” pays you $f(X)$ on that day for some function f.

For example, suppose stock XYZ costs \$120 today and the function is

$$f(X) = \begin{cases} 120 - X & \text{if } X < 120 \\ 0 & \text{otherwise.} \end{cases}$$

This is often referred to as a put option. It is essentially giving you the option to purchase the stock in one month and sell it at today’s price.

Binomial asset pricing model II

- If the price goes up, you would not use that option and you would make zero (but not lose any additional money, such as the cost of buying the option in the first place).
- Otherwise, you would make $120 - X$.
- In effect, you are betting that the price will go down.

Suppose you are a bank and someone wants to purchase this put option. You need to determine the price.

What would be the fair price to charge them?

To make a good guess, we need a model for the asset price X.

Once we have such a model, we can derive analytical expressions for $f(X)$ or use Monte Carlo methods, which we will discuss soon.

Binomial asset pricing model III

One of the simplest models for X is the Binomial Asset Pricing Model, which says that at every time step (e.g., every minute), the price goes up by one penny with probability p, or down by one penny with probability $1 - p$.

In this example, both X and $f(X)$ are random variables.

Question: The Binomial Asset Pricing Model is a very simple model, especially given how complicated the stock market is. Can you think of any possible problems with the model?

Election models I

Will the Democratic (D) or Republican (R) presidential candidate win Wisconsin in 2024?

What is the distribution of this random variable W, where $W = -1$ if D and $W = 1$ if R?

What about neighboring Michigan $M \in \{-1, 1\}$?

Wisconsin and Michigan are not so different, so if we find out that the Republican candidate won Michigan (i.e., we learn about M), then that certainly tells us something about W.

That is to say, W and M are *not* independent.

What if we wanted to model both Wisconsin and Michigan together?

We usually write this as a pair (W, M).

Election models II

One thing you could do is model the proportion of votes for D vs R in Wisconsin (ignoring third parties) as normally distributed.

- Perhaps you consider this to be $W_p \sim \text{Normal}(1/2, .05)$.
- Then, W is 1 if $W_p > .5$ and \$W=-1 if $W_p < .5$.

This helps because we can do the same thing for M, based on and M_p.

We can model (W_p, M_p) as being correlated via the multivariate normal.

To simulate these, you need to specify both the mean, which is now a two-dimensional "vector" and a covariance matrix $\Sigma \in \mathbb{R}^{2\times 2}$.

The diagonal of the matrix Σ specifies the variances of W_p and M_p respectively, and the off-diagonal $\Sigma_{1,2}$ specifies the covariance between W_p and M_p.

Covariance is similar to correlation, a measure of how closely two variables track one another.

Election models III

- When the covariance is large and positive, the two variables tend to track one another.
- If the covariance is large and negative, the two variables are inversely related to one another.

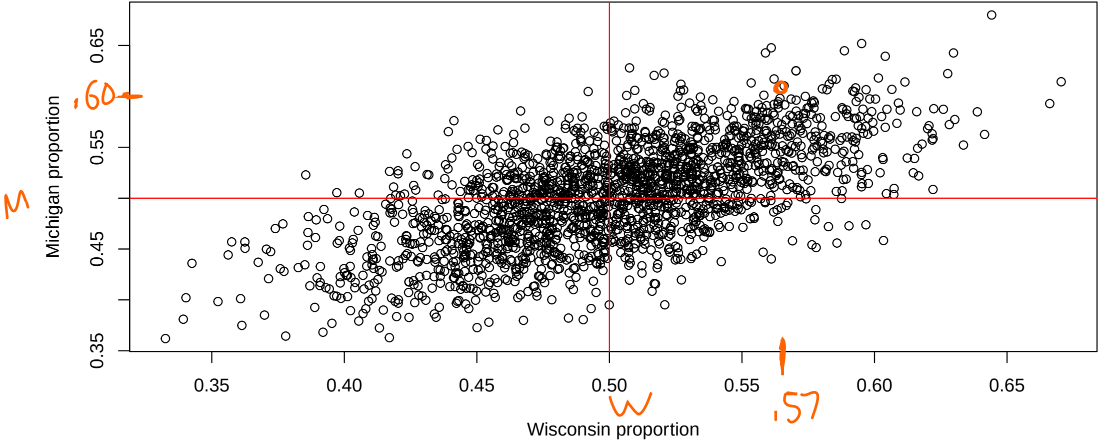
Let's try simulating some election results.

```
mu <- c(.5,.5); ## Vector of means; both W_p and M_p are mean 1/2.
# Make a two-by-two symmetric matrix.
CovMx <- matrix( c(.05^2,.04^2,.04^2,.05^2), nrow = 2);
CovMx;
##        [,1]   [,2]
## [1,] 0.0025 0.0016
## [2,] 0.0016 0.0025
```

Election models IV

```
library(MASS); ## This library includes a multivariate normal
#mvrnorm is the multivariate version of rnorm.
WpMp = mvrnorm(n=2000, mu=mu, Sigma=CovMx);
plot(WpMp, xlab = "Wisconsin proportion", ylab = "Michigan proportion");
abline(h=.5, col="red"); abline(v=.5, col="red")
```

Election models V



Each point in this plot corresponds to one simulated election.

Election models VI

Questions:

- What region of this plot corresponds to $W = -1$ and $M = +1$?
- Does it make sense that there are fewer points in the top left compared to the top right?
- Are Michigan and Wisconsin positively or negatively correlated (based on this plot, anyway)?

Review:

In these notes we covered:

- The basic rules of probability: outcome spaces, events
- The concept of a random variable
- Families of discrete random variables: Bernoulli, binomial, geometric, Poisson and uniform
- Families of continuous random variables: Gaussian (normal), exponential and uniform
- The concept of expected value
- PMF, PDF and CDF
- Computing probabilities
- Some applications of random variables