

Physics 2660

Lecture 7: C – Part 6

Today

- More probability distributions: Binomial, Poisson, ...
but the world is mostly Normal
- Histograms with weights
- Default parameters and constant specifiers in functions
- More on Structures

Part 1a: The Bernoulli Distribution



2

Today we'll talk about three new probability distributions, in addition to the Gaussian (Normal) distribution and the uniform distribution that we've already been using.

The Bernoulli Distribution:

- Only two possible outcomes (true or false, success or failure).
- The probability, p , of one possible outcome is known.

$$\begin{aligned} P(\text{heads}) &= p \\ P(\text{tails}) &= (1-p) \end{aligned}$$



The Bernoulli distribution gives the probability of observing a true result in a single TRUE/FALSE test – it describes a simple flip of a coin.

p is the probability of success on each test. (the coin may be lopsided)

It describes single true/false experiments.

Consider a coin toss: TRUE = HEADS FALSE = TAILS

What is the probability of getting one HEADs-up if you flip the coin 1 time?

Bernoulli Example: Coin Toss

Outcome 1: Heads



$$\begin{aligned} P(\text{heads}) &= p \\ &= 0.5 \end{aligned}$$

Outcome 2: Tails



$$\begin{aligned} P(\text{tails}) &= (1-p) \\ &= 0.5 \end{aligned}$$

Bernoulli Example: Die Roll

Outcome 1: Roll a 6.



$$\begin{aligned} P(6) &= p \\ &= 1/6 \\ &\text{(about 0.17)} \end{aligned}$$

Outcome 2: Roll something else.



$$\begin{aligned} P(\text{tails}) &= (1-p) \\ &= 5/6 \\ &\text{(about 0.83)} \end{aligned}$$

5

The outcomes don't both need to have the same probability, as we see here.

Part 1b: The Binomial Distribution



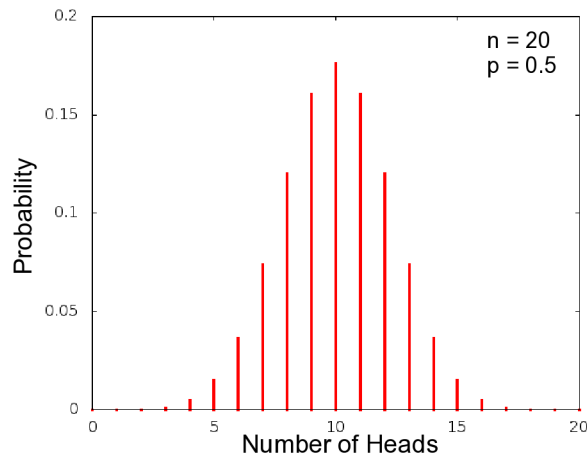
What if we flip a coin many times?

The Binomial Distribution:

What if we flip a coin many times, or if we flip many coins? How many instances of a given outcome (say, heads) should we expect?

The **Binomial distribution** gives the probability of observing a certain number of true results, x , after doing n tests.

p is the probability of success on each test.



It's used when we are interested in a number of TRUE/FALSE experiments.

Consider n coins tosses:
TRUE = HEADS,
FALSE = TAILS

What is the probability of getting x HEADS-up if you flip the coin n times?

7

The binomial distribution gives us a way to calculate the probability of getting “ x ” successes in “ n ” trials. This is a useful thing in the real world.

Understanding the Binomial Distribution:

- x = number of successes (e.g., how many heads?)
- n = number of trials (e.g., how many tosses?)
- p = probability of success in a single trial.

- $P(x;n,p)$ = Probability of seeing x successes after n trials, given probability p of success.

$$P(x; n, p) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

Number of possible ways to arrange the x successes and $(n-x)$ failures.

Probability of getting x successes and $(n-x)$ failures.

You can think of the formula as being made up of two parts: The second part just calculates the probability of getting “ x ” successes and “ $n-x$ ” failures in “ n ” trials. For example, if we were rolling a 6-sided die, the probability of getting a 1 twice would be $(1/6)*(1/6)$, and the probability of getting another number three times would be $(5/6)*(5/6)*(5/6)$. So, the probability of rolling 1,1,2,3,4 would be $(1/6)^2 * (5/6)^3$.

But there are lots of ways we could rearrange these numbers and still have two ones: 1,2,3,4,1; 2,3,4,1,1; 2,1,3,4,1;... etc. The first term above calculates the number of permutations that would produce “ x ” successes and “ $n-x$ ” failures.

Binomial Distribution for One Fair Coin Flip:

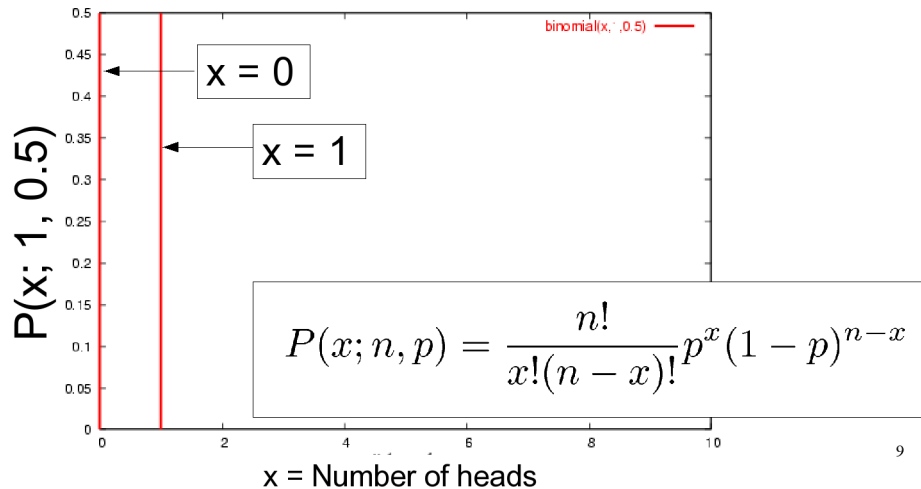
Here's the Binomial distribution with

$n = 1$ (one coin flip)

$p = 0.5$, 50% chance of heads on a flip

$1 - p = 0.5$, 50% chance of tails on a flip

One toss, 50% heads/tails



So, if you flip one fair coin the probability of getting one head is 50%, and the probability of getting zero heads is 50%. The probability is zero for any other number of heads, as you'd expect. (The formula for the Binomial distribution takes care of this, because any other number ends up producing a factorial of a negative integer, which isn't defined.)

Binomial Distribution for One Biased Coin Flip:

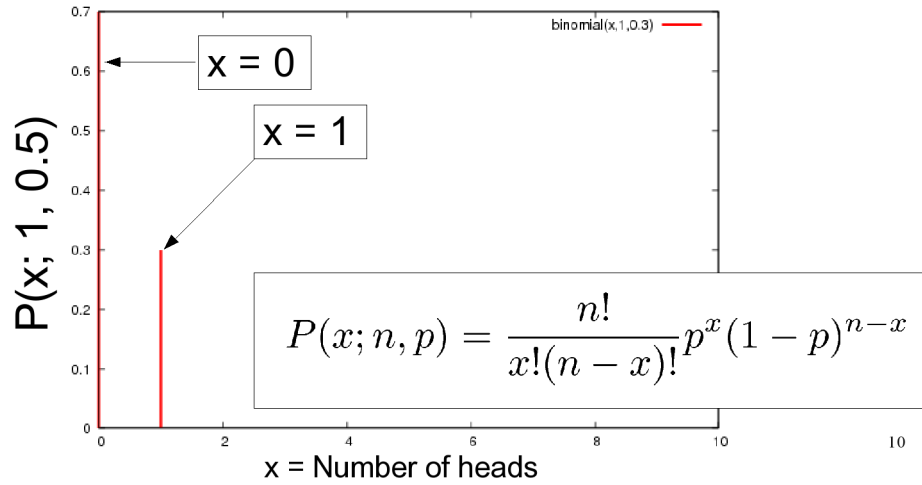
Here's the Binomial distribution with

$n = 1$ (one coin flip)

$p = 0.3$, 30% chance of heads on a flip

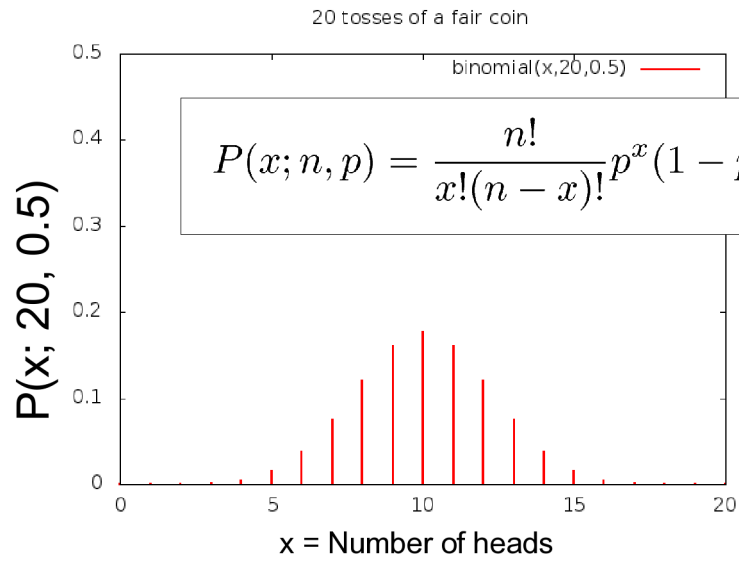
$1 - p = 0.7$, 70% chance of tails on a flip

One toss, 30% heads/ 70% tails



Again, this is just what we'd expect, intuitively.

Binomial Distribution for 20 Fair Coin Flips:



Notice that the mean value for x is 10. Could we have predicted this?

Now let's look at this more interesting case. Here we flip the coin 20 times, and look at the probabilities of getting various numbers of heads.

Mean and Variance for the Binomial Distribution:

Given the binomial distribution:

$$P(x; n, p) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

We can calculate the mean value (μ) of x :

$$\mu = np$$

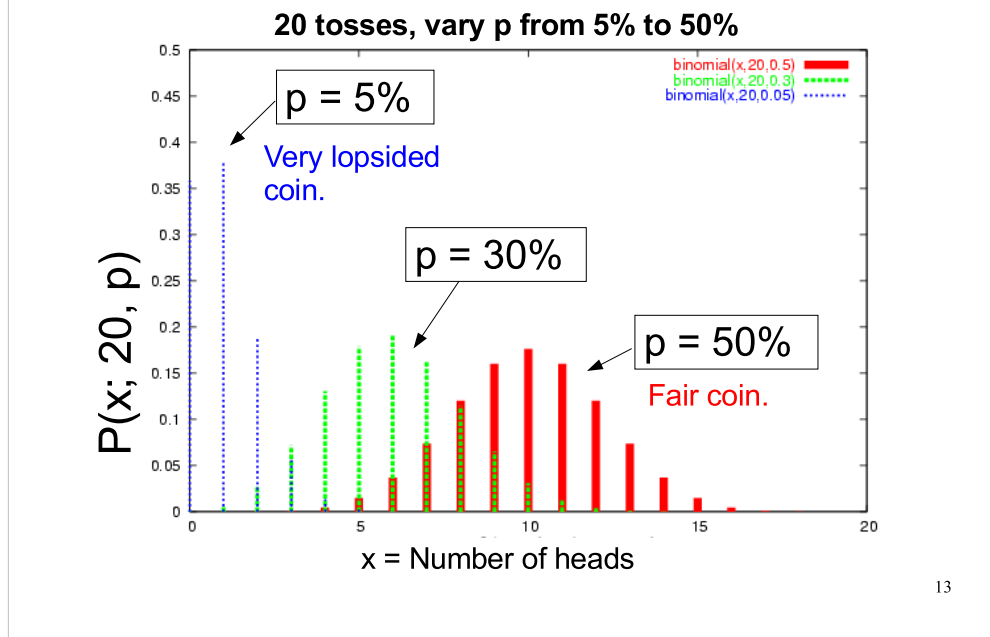
and the variance (σ^2) of x :

$$\sigma^2 = np(1-p)$$

12

Note that, although the Binomial distribution formula is complicated, the mean and average are very simple expressions.

The Effect of Varying p :



Notice that for small values of p , the distribution gets very asymmetrical, and squashed up against the y axis. For $p=50\%$, though, the distribution looks almost like the Gaussian (Normal) distributions we're used to seeing.

Part 1c: The Poisson Distribution



Probabilities from the Binomial distribution are difficult to calculate when large numbers are involved, because of the factorials in the equation. This was much more of a problem before computers were available.

As we'll see, there are a couple of useful approximations to the Binomial distribution for which calculations are much easier.

The first of these is the Poisson distribution.

The Poisson Limit:

An interesting special case of the binomial distribution is the one in which:

- The number of trials, **n**, approaches infinity,
- The probability of success, **p**, approaches zero,
- The mean number of successes, **$\mu = np$** , remains fixed.

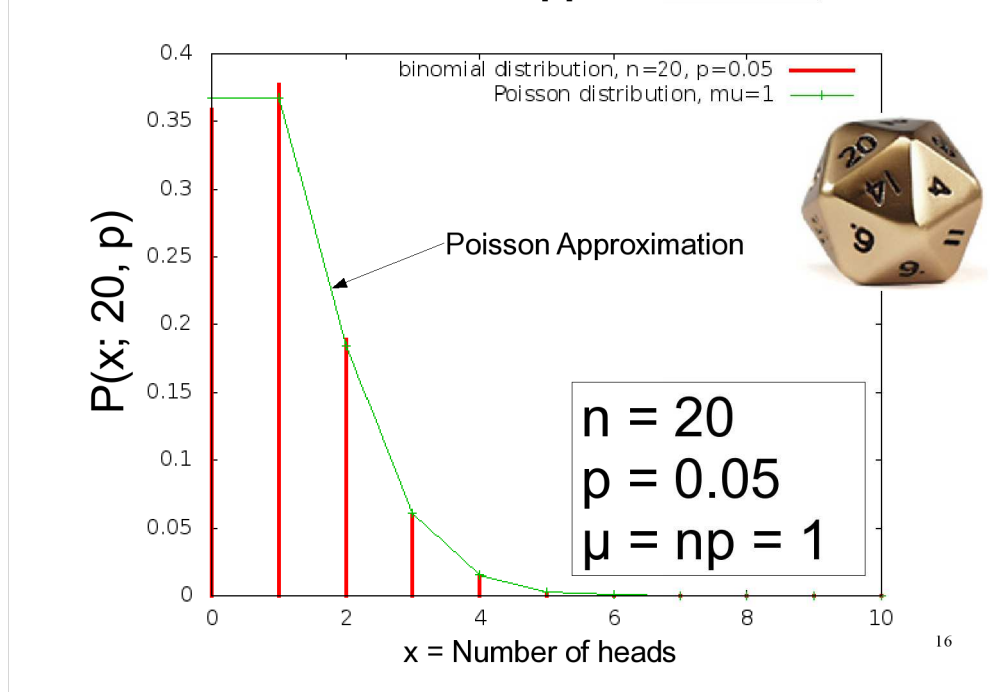


As these limits are approached, the binomial distribution can be approximated by the following (much simpler) expression:

$$P(x; \mu) = \frac{\mu^x}{x!} e^{-\mu}$$

This is called the **Poisson Distribution**, and it is valid when p is small, n is large and μ is some intermediate value.

How Good is the Poisson Approximation?



There's a rule of thumb that says the Poisson distribution is a good approximation of the Binomial distribution if n is at least 20 and p is smaller than or equal to 0.05, and an excellent approximation if $n \geq 100$ and $np \leq 10$.

So, even with a modest number of trials and a not-particularly microscopic value for p , we're already in a place where the Poisson approximation is good enough for many purposes.

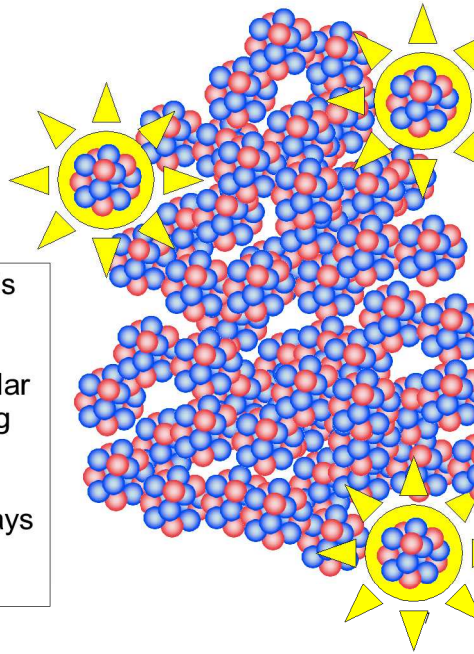
The values above ($n=20, p=0.05$) would apply if we were rolling a 20-sided die. The probability of rolling, say, "1" some number of times (" x ") is given by the graph.

Radioactive Decay:

Consider a sample of a radioactive material like Uranium.

We point a detector at the sample and count the number of radioactive decays that happen during a five-minute period.

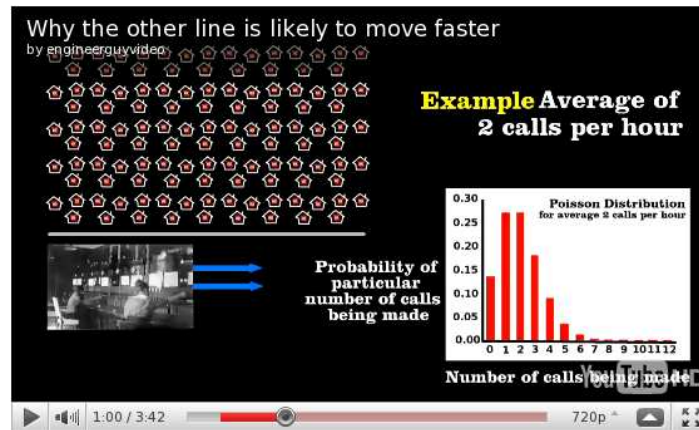
- **n is large**, on the order of Avogadro's Number.
- **p is small** (it's unlikely that a particular nucleus will decay while we're looking at it).
- **$\mu = np$** , the average number of decays in five minutes, is still an appreciable number, since n is so large.



So, if we observed a radioactive source with a Geiger counter for five minutes, then repeated this experiment several times, we'd find that the number of counts we see in each five-minute sample would be distributed in a Poisson distribution.

Networks and Queues:

Consider a network of phones. There are lots of phones (**large n**), but it's unlikely that any particular phone will be in use at a given time (**small p**):



<http://www.engineerguy.com/videos/video-lines.htm>

18

Here's another real-world example where Poisson statistics are important.

Note that the video also mentions a distribution called “Erlang-B”. This, and its companion “Erlang-C”, deals with the probability of observing a given waiting time when customers are waiting in a queue.

There's also a programming language named after Erlang:

<http://learnyousomeerlang.com/>

Mean and Variance of the Poisson Distribution:

Given the Poisson distribution:

$$P(x; \mu) = \frac{\mu^x}{x!} e^{-\mu}$$

We can derive the very simple expression for the variance of x:

$$\sigma^2 = \mu$$

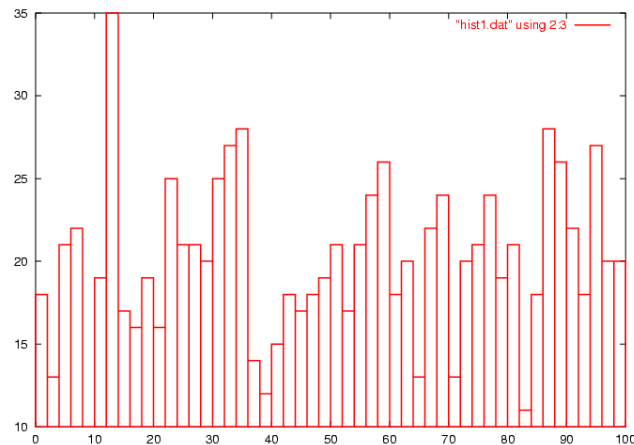
So the standard deviation is:

$$\sigma = \sqrt{\mu}$$

19

We saw earlier that the mean and variance of the Binomial distribution have really simple expressions. This is even more true of the Poisson distribution.

Implications for Histograms:



Consider the following:

We fill a histogram with a large number of entries, n .

The probability, p , that any given entry will land in a particular bin is small.

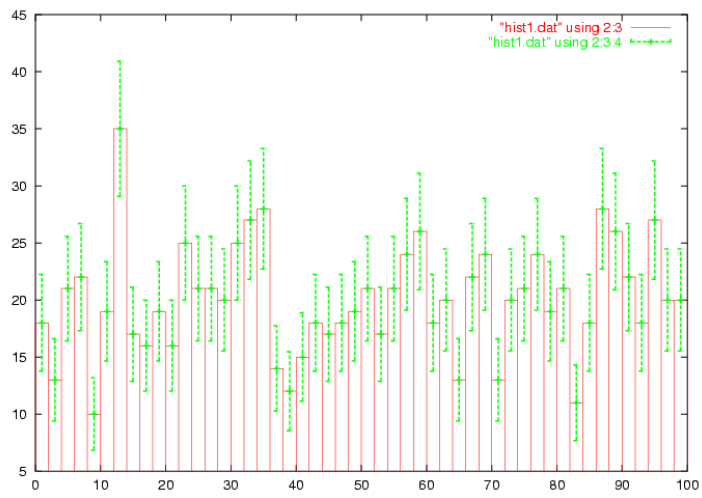
This implies that we can use Poisson statistics to describe the variations in the number of counts in a given histogram bin.

If the count in a given bin is m , then the best estimate of the uncertainty in the bin count is $\sigma = \text{sqrt}(m)$.

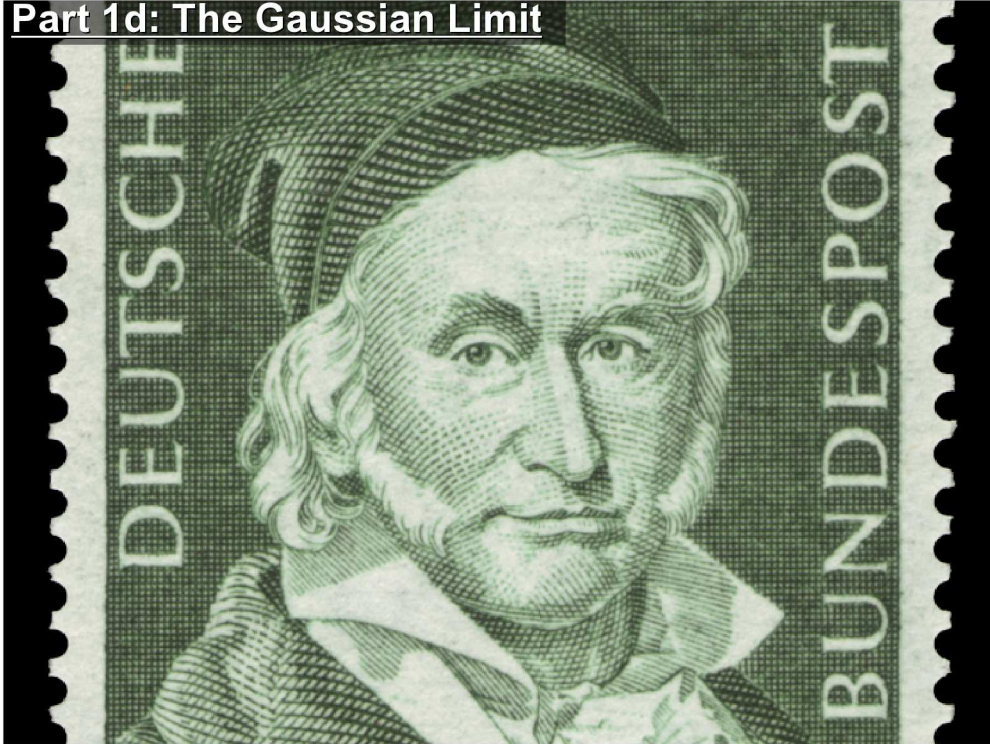
20

This is what we saw in last week's lab: When we took 100 histograms, generated identically, and compared the values in one particular bin we saw that the standard deviation of the values was just the square root of the mean.

Histogram with Error Bars:

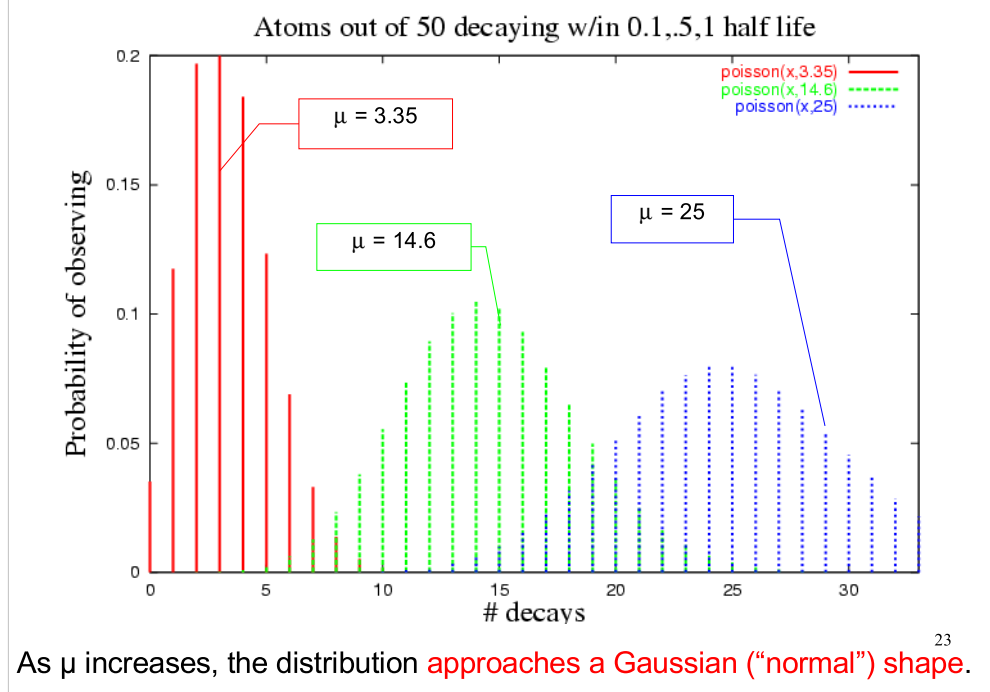


Part 1d: The Gaussian Limit



As we noticed earlier, the Binomial distribution starts to look like a Gaussian distribution as we move far away from zero.

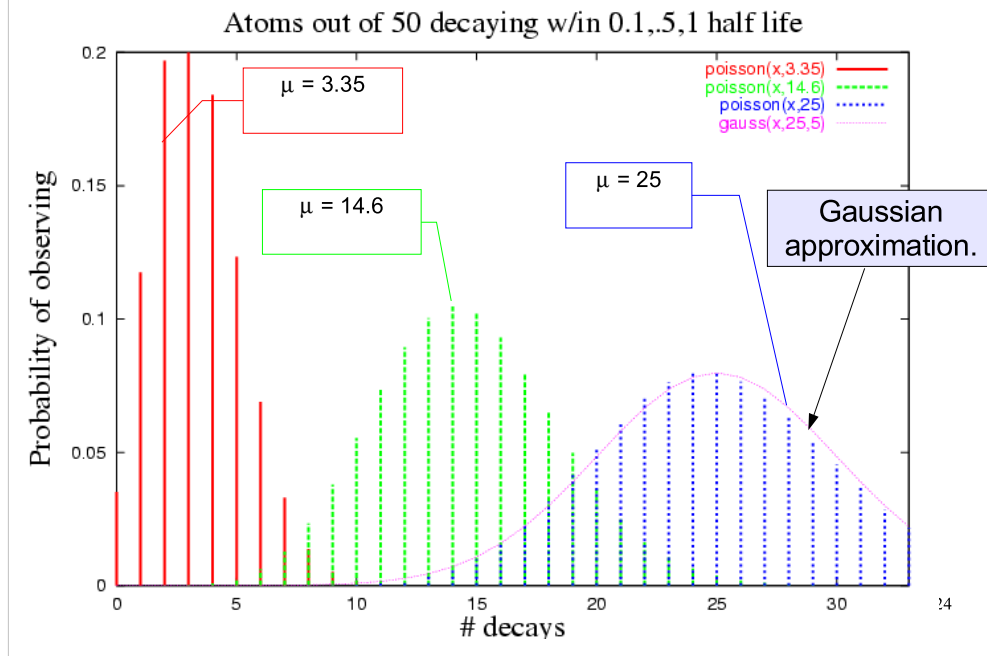
Poisson Distribution for $\mu \gg 0$:



This is true for the Poisson distribution as well. As the mean moves away from zero, the Poisson distribution becomes more symmetrical and takes on the Gaussian shape.

Note that this is a **special case** of the Gaussian distribution where we require that the variance be equal to the mean. (Remember that for the Poisson distribution, $\sigma^2 = \mu$.)

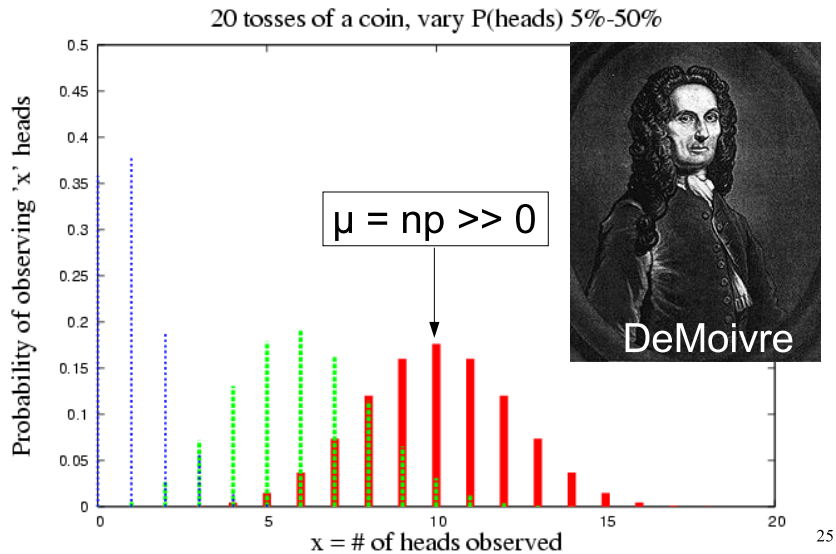
How Good is this Approximation?



As you can see, even shifting the mean a small way away from zero is enough to make the Poisson distribution pretty close to a Gaussian distribution.

The Gaussian Limit of the Binomial Distribution:

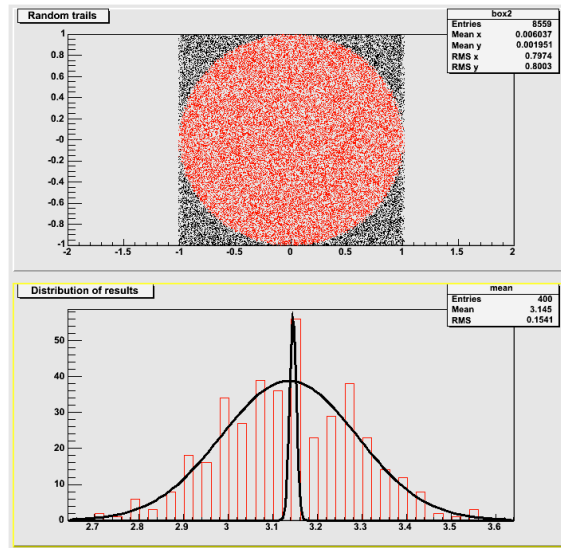
As we noticed earlier, the **Binomial distribution** itself looks like a **Gaussian** when the mean is sufficiently far away from zero.



As we noted earlier, calculations involving the Binomial distribution can be really difficult if you don't have a computer. Because of the factorials, the Binomial distribution becomes untractable when you try to apply it to large populations. For many years early researchers in probability and statistics looked for a good approximation to the Binomial distribution that was easier to calculate. Abraham DeMoivre eventually identified what we now call the Gaussian or Normal distribution.

Gaussian Distributions in Monte Carlo Results:

Because of the Gaussian limit of the binomial distribution, we see Gaussian shapes appear when we look at “coin tossing” experiments like our Monte Carlo integration examples:

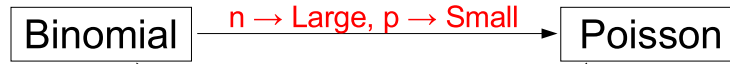


26

Statistical errors tend to follow Gaussian distributions for sufficiently large samples. This is an example of the Central Limit Theorem – a remarkable result that lies at the core of probability theory.

Relations between Distributions:

$$P(x; n, p) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x} \quad P(x; \mu) = \frac{\mu^x}{x!} e^{-\mu}$$



$n \rightarrow \text{Large},$
 $np = \mu \gg 0$

$\mu \gg 0$

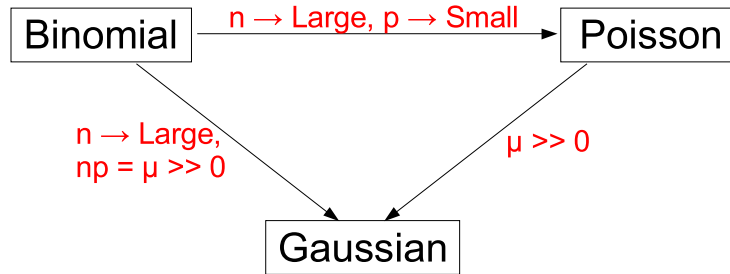
Gaussian

$$P(x; \mu; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Picking the Right Distribution:

Need full Binomial description when there is a good chance you see **events at both limits**: $x=0$ and $x=n$.

Can use Poisson limit when you are likely to observe **events at $x=0$** , but **not likely to see events at $x \gg \mu$** approaching upper limit.



Can use Normal/Gaussian limit when **number of trials is large** and observed data are **not likely to land near either the upper or the lower limits**.

Part 2: Optional Function Parameters



In the U.S., we have these three-hole wall outlets. But we can still use two-prong plugs in them. The third connector is optional.

C++ allows us to do the same sort of thing with functions. We can have optional arguments that have default values.

Defining Default Parameter Values:

Consider the following prototype statement from “hist.hpp”, one of the header files associated with our “p2660” library:

```
void h1fill(h1 *hist, double x, double wgt=1.0);
```

As we saw in lab last week, we can call “h1fill” with an optional third argument (a weighting factor).

This is possible because the function's prototype defines a **default value** (“1.0”) for the last parameter. If we don't specify a value for this parameter when we use the function, the compiler just assumes that it has the default value.

This is a feature that's only available in **C++**. It won't work in vanilla C.

Rules for Default Parameter Values in C++:

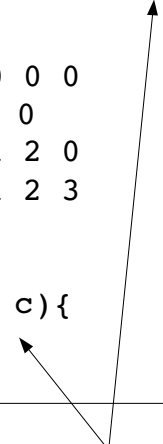
- Default values should be specified in the function's **prototype**.
- **Any number** of parameters may have defaults.
- All other parameters **after** a parameter with defaults must also have defaults.

Default Parameter Example:

```
void printstuff(int a=0, int b=0, int c=0);

int main(){
    printstuff();           // prints 0 0 0
    printstuff(1);         // prints 1 0 0
    printstuff(1,2);       // prints 1 2 0
    printstuff(1,2,3);     // prints 1 2 3
}

void printstuff (int a, int b, int c){
    printf("%d %d %d\n",a,b,c);
}
```

Two arrows originate from the function definition. One arrow points from the parameter list '(int a, int b, int c)' in the definition to the parameter list '(int a=0, int b=0, int c=0)' in the prototype. The second arrow points from the same parameter list in the definition to the closing brace '}' of the prototype.

Note that we specify the defaults in the **prototype**, not the function definition (and not both).

Part 3: More on Structures



Finally today, a little note about avoiding a potential problem when giving functions pointers to structures.

Unintended Changes:

Consider the following innocuous-looking code:

```
typedef struct{
    double re, im;
} Complex;

// Return the magnitude of the sum of z1 and z2:
double magsum(Complex *z1, Complex *z2) {
    z1->re += z2->re;
    z1->im += z2->im;
    return sqrt( z1->re*z1->re + z1->im*z1->im );
}
```

What happens if we want to use the value of z1 somewhere later in our program, after calling "magsum"?

The author of the magsum function wasn't thinking about this!

Preventing Unintended Changes with “const”:

You can protect yourself from mistakes like this by using “const”:

```
typedef struct{
    double re, im;
} Complex;

// Return the magnitude of the sum of z1 and z2:
double magsum(const Complex *z1,
              const Complex *z2) {
    z1->re += z2->re;
    z1->im += z2->im;
    return sqrt( z1->re*z1->re + z1->im*z1->im );
}
```

The compiler will make sure you don't change data you specify as constant:

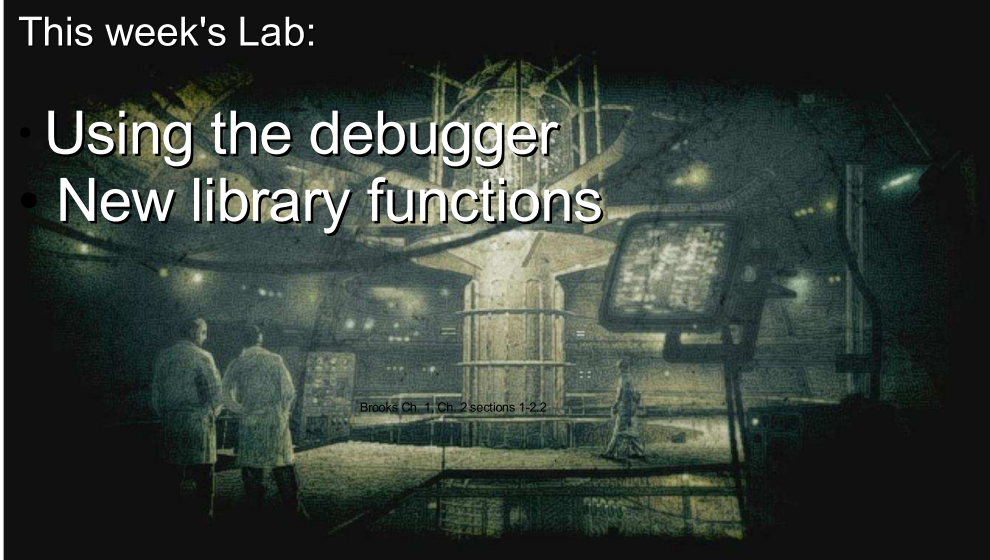
```
g++ test.cpp
test.cpp : In function `magsum':
test.cpp:9: warning: increment of read-only member `re'
```

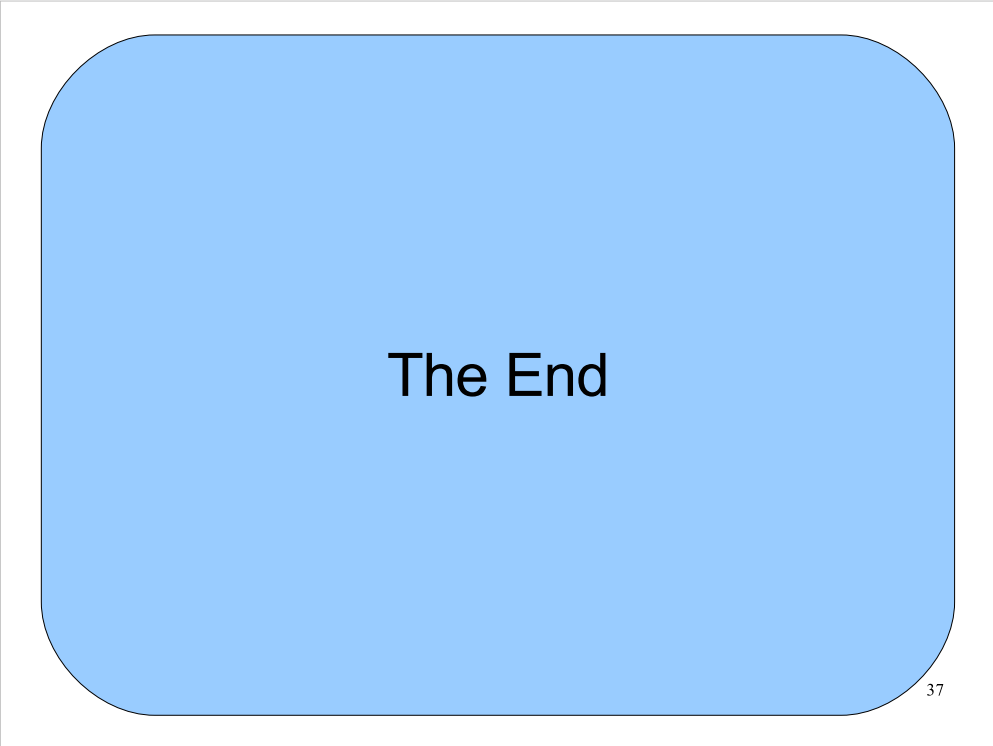
Next Time:

- “Classes” in C++
- Searching and Sorting

This week's Lab:

- Using the debugger
- New library functions





Thanks!