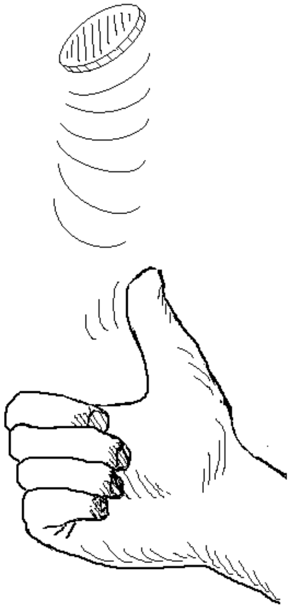
Lecture 1: Probability and Generative Models

Susan Holmes and Wolfgang Huber

September 30, 2019

Binary Data



- Mutations at each position the Reverse Transcriptase (RT) gene of HIV follows a Poisson(0.0005 = 5 . 10^{-4}) distribution.

- $\longrightarrow$ Average number of mutations in the first 10,000 positions is ?

- with a standard error of ?

Other Examples

- How many reads of DNA match a reference pattern?
- How many CG digrams we observe in a sequence.
- How many binding sites?

Using discrete probability models for binary events

We use Poisson and Binomial distributions as models for binary outcomes.

Poisson Model for rare events

In the HIV model above, suppose we want to find the probability of seeing 3 mutations in a sequence of length 10,000?

```
dpois(x=3,lambda=5)
[1] 0.1403739
```

There is a formula:

$$P(x = k) = \frac{\lambda^k e^{-\lambda}}{k!} \text{ For } \lambda = 5, \text{ we get } P(x = 3) = \frac{5^3 \times e^{-5}}{3 \times 2}.$$

```
(5^3*exp(-5))/6
[1] 0.1403739
```

```
dpois(x=3,lambda=5)
[1] 0.1403739
```

Epitope Example

The molecular sites responsible for allergic reactions are called epitopes.

We carry out an **ELISA** assay where we know the false positive rate is 0.01 (1 percent).

P(Declare Epitope | No Epitope)

We take 50 patient samples over a protein tested at 100 positions.

The data from one patient looks like this:

```
[1]   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
[35]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[69]  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

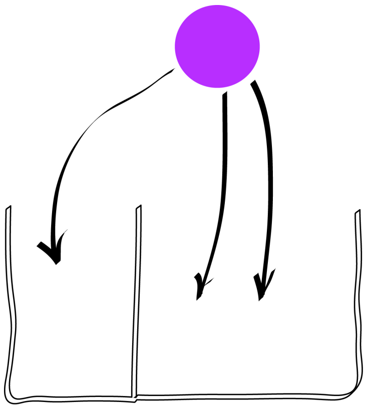
Results from 50 assays

If there are no epitopes and the counts follow a Poisson(0.01) distribution. Each individual position and patient has a probability of 1 in 100 of being a 1.

At any given position after seeing 50 patients, we expect the sum to have a Poisson distribution with parameter 0.5.

Binomial Success Counts

When we have a binary outcome (success/failure), (CpG/NonCpG), (M/F), (Y=Pyraminidine,R=Purine), (Diseased/Healthy), True/False.



We can model it as a simple random variable with probability of success p and failure $1 - p$.

`SSSSSFSSSSFFFSF` is summarized as (#Success=10,#Failures=5).

The number of successes is said to have a Binomial distribution with parameters $prob = 0.3$ and $size = 15$.

We say the random variable is categorical or that it is a **factor** variable with two levels `S` and `F`.

```
outcomes=factor(unlist(strsplit("SSSSSFSSSSFFFSF","")))
outcomes
 [1] S S S S S F S S S S F F F S F
Levels: F S
table(outcomes)
outcomes
 F  S
 5 10
```

Using R to explore the Binomial

We use special functions tailored for each type of distribution,
- `rbinom` for the binomial,
- `rmultimom` for the multinomial,
- `rpois` for the Poisson.

Each will need a different set of parameters to be fixed.

Suppose we wanted to simulate a sequence of fifteen fair coin tosses, then we write

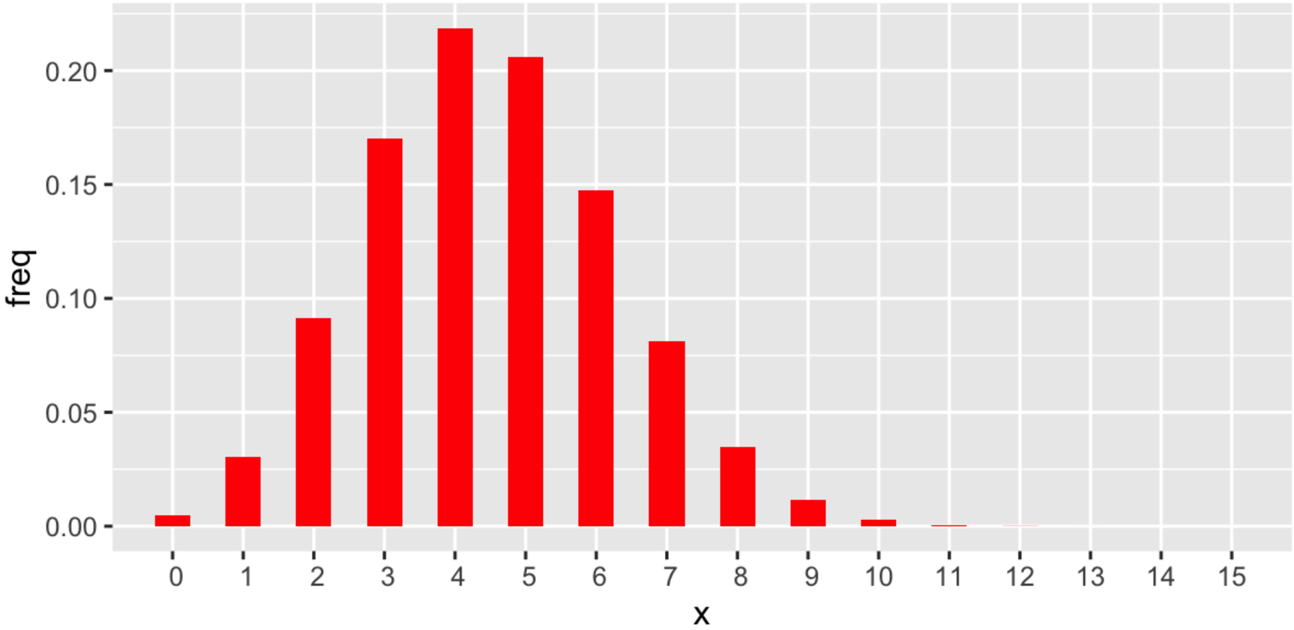
```
rbinom(15,prob=0.5,1)
 [1] 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0
```

```
set.seed(235569511)
rbinom(1,prob=0.3,15)
[1] 3
rbinom(1,prob=0.3,15)
[1] 5
rbinom(1,prob=0.3,15)
[1] 2
rbinom(1,prob=0.3,15)
[1] 5
rbinom(1,prob=0.3,15)
[1] 5
```

Notice that 5 comes out quite often, the theoretical proportion of times that 5 does come out is the value of the probability that X=5 for X a Binomial with n=15, p=0.3 and it's complete distribution can be given by typing:

```
vec15=dbinom(0:15,prob=0.3,15)
round(vec15,3)
 [1] 0.005 0.031 0.092 0.170 0.219 0.206 0.147 0.081 0.035 0.012 0.003
[12] 0.001 0.000 0.000 0.000 0.000
```

Here’s a plot of the theoretical distribution:



Theoretical Distribution of Binomial(15,0.3)

When we know the number of events, the parameter we call n and when we know the probability of success, we can compute the probability of seeing k=5 successes

```
dbinom(5,p=0.3,15)
[1] 0.2061304
```

this actually uses a closed form formula:

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

Multinomial

Several possibilities for levels: (AA, Aa, aa)

Number of levels in a categorical variable can be very large.

If we are only measuring one categorical variable on a sample, we usually tally the frequencies of the different levels in a vector of counts.

```
genotype1=c("AA", "AO", "BB", "AO", "OO", "AO", "AA", "BO", "BO", "AO", "BB",
            "AO", "BO", "AB", "OO", "AB", "BB", "AO", "AO")
table(genotype1)
genotype1
AA AB AO BB BO OO
 2  2  7  3  3  2
```

We encode these variables as `factor` variables

```
genotype=factor(genotype1)
genotype
 [1] AA AO BB AO OO AO AA BO BO AO BB AO BO AB OO AB BB AO AO
Levels: AA AB AO BB BO OO
table(genotype)
genotype
AA AB AO BB BO OO
 2  2  7  3  3  2
```

Examples with 4 categories:

```
rmultinom(1,prob=c(3/4,1/12,1/12,1/12),1)
     [,1]
[1,]    0
[2,]    0
[3,]    0
[4,]    1
rmultinom(1,prob=c(3/4,1/12,1/12,1/12),1)
     [,1]
[1,]    1
[2,]    0
[3,]    0
[4,]    0
t(rmultinom(1,prob=c(3/4,1/12,1/12,1/12),1))
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
```

We could have replaced these four draws with one draw of 4:

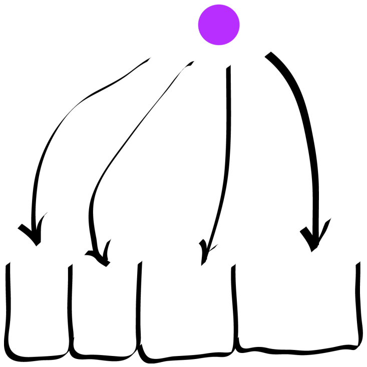
```
rmultinom(1,prob=c(3/4,1/12,1/12,1/12),size=4)
     [,1]
[1,]    3
[2,]    0
[3,]    1
[4,]    0
rmultinom(4,prob=c(3/4,1/12,1/12,1/12),size=1)
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    0
[2,]    0    0    0    1
[3,]    0    0    0    0
[4,]    0    0    0    0
```

Multinomial Distribution: the formula

Here is the formula the programs use to compute the probability of a multinomial vector of counts $(x_1, \dots, x_m)$ being observed:

$$\begin{aligned} P(x_1, x_2, \dots, x_m | p_1, \dots, p_m) &= \frac{n!}{\prod x_i!} \prod p_i^{x_i} \\ &= \binom{n}{x_1, x_2, x_m} p_1^{x_1} p_2^{x_2} \cdots p_m^{x_m} \end{aligned}$$

Multinomial distributions: the case of DNA



Just as in the binomial case the sum of the probabilities of all possible outcomes is 1, $p_A + p_C + p_G + p_T = 1$.

Suppose we have four boxes that are equally likely.

What is the probability of observing 4 in the first box, 2 in the second box, and none in the two other boxes?

```
dmultinom(c(4, 2, 0, 0), prob = rep(1/4, 4))
[1] 0.003662109
```

Simulation experiments to check whether the data we see are consistent with the simplest possible four-box model where each box has the same probability, 1/4, of occurring.

In some sense it is the strawman (nothing is happening).

We'll see more examples of this in Lecture 2. Here we use a few R commands to generate such vectors of counts. First suppose we have 8 characters of four different equally likely types:

```
pvec <- rep(1/4, 4)
t(rmultinom(1, prob = pvec, size = 8))
     [,1] [,2] [,3] [,4]
[1,]    2    5    0    1
```

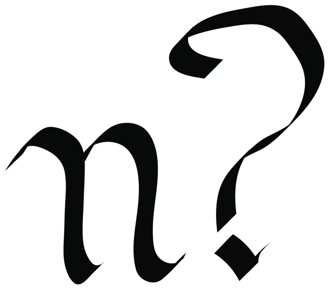
Simulating for power

Power= Probability of realizing a hypotheses is improbable if in fact it is untrue:

Power= P(reject H_0 | H_0 false)

Power is the **true positive rate**.

Using Monte Carlo and the **multinomial** in a way which is related to a problem scientists often have to solve when planning their experiments: how big a sample size do I need?



We always tell you they need more data: the larger the sample size the more sensitive the results.

However lab work is expensive, so there is a tricky cost-benefit tradeoff to be considered. This is such an important problem, we have dedicated a whole Lecture to it at the end of the book (Lecture 10).

However for now, we use this as an example of how simulating from a simple probability model can be useful.

Standard power: 80%

This means that if the planned experiment is run many times, about 20% of the time it will fail to yield significant results when it should.

We will call H_0 the null hypothesis that the DNA data we have collected comes from a *fair* process where each of the 4 nucleotides is equally likely $(p_A, p_C, p_G, p_T) = (0.25, 0.25, 0.25, 0.25)$.

So-called nucleotide bias can help detect selective pressure and drug resistance

Let's determine if, by looking at a sequence of length $n = 20$, we can detect whether the original distribution of nucleotides is fair or whether it comes from an alternative process.

We generate 1000 simulations from the null (fair) hypothesis using the function `rmultinom`, we display only the first 12 columns to save space.

```
set.seed(3439)
obsunder0 = rmultinom(1000,prob=pvec,size=20)
obsunder0[,1:12]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    6    4    5    7    5    5    4    3    7     4     7     5
[2,]    2    2    4    2    4    2    6    5    3     6     5     4
[3,]    9    3    8    4    5    7    4    7    6     8     3     6
[4,]    3   11    3    7    6    6    6    5    4     2     5     5
```

Each column in the matrix is a simulated instance. You can see that the numbers in the boxes vary a lot: some are as big as 11, whereas the expected value is 5=20/4.

Creating a test

Expected values aren't enough.

We need a measure of variability that will allow us to describe how much variability is expected and how much is too much.

$$\texttt{stat} = \frac{(E_A - x_A)^2}{E_A} + \frac{(E_C - x_C)^2}{E_C} + \frac{(E_G - x_G)^2}{E_G} + \frac{(E_T - x_T)^2}{E_T} = \sum_i \frac{(E_i - x_i)^2}{E_i}$$

How much do the first two columns of the generated data differ from what we expect? We get:

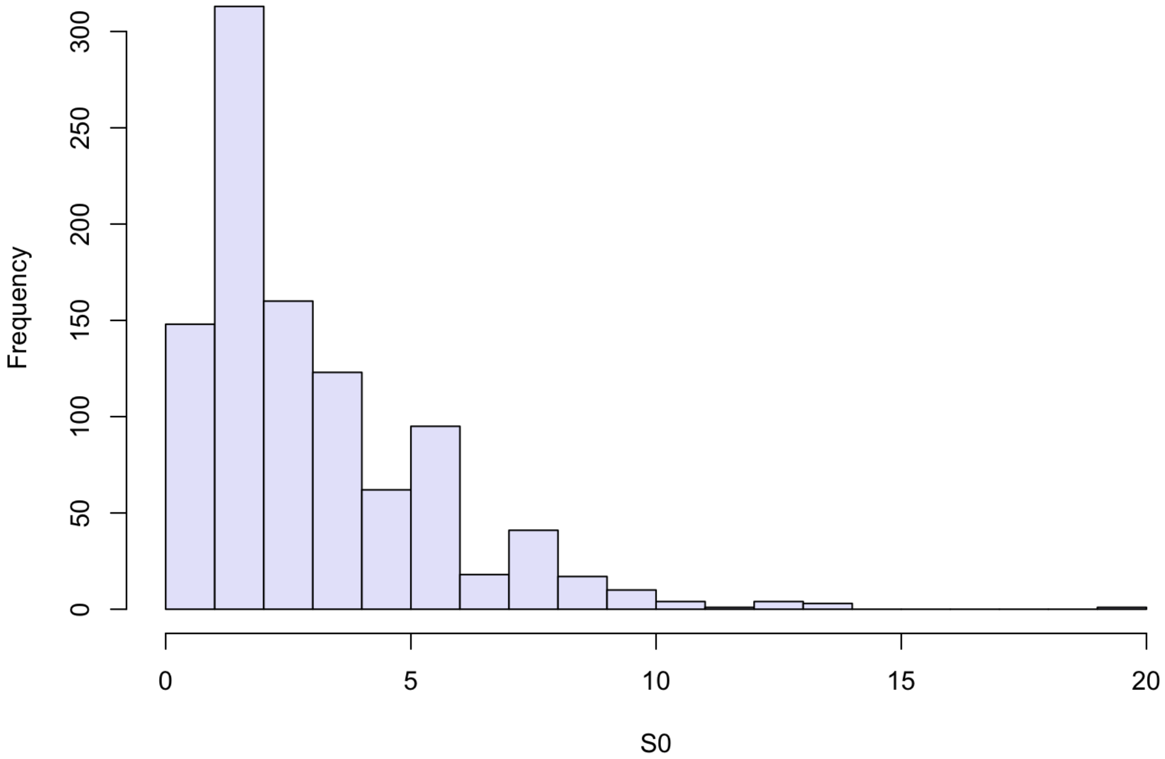
```
expected0=pvec*20
sum((expected0-obsunder0[,1])^2/expected0)
[1] 6
sum((expected0-obsunder0[,2])^2/expected0)
[1] 10
```

As we expect, the values of the measure differ.

Simulate the experiment 1000 times.

Store these values in a vector we call `S0`.

```
S0=apply(obsunder0,2,function(x)sum((expected0-x)^2/expected0))
summary(S0)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.200   2.800   3.076   4.400  19.600
hist(S0,20,col="lavender",main="")
```



S0 takes a spread of values called a distribution

Compute 95% quantile (the value that separates the smaller 95% values from the 5% larger values).

```
val=quantile(S0, probs=0.95)
val
95%
7.6
```

In response to our request for the 95% quantile, R tells us that are only 5% of the values are larger than 7.6.

Reject if the weighted sum of squares `stat` is larger than 7.6

Determining our test's power

We must compute the probability that our test – the weighted sum-of-square differences – will detect when the data do in fact come from an alternative pvecA.

To compute this probability we generate 1000 simulated instances from the alternative process. These play the role of observations.

```
pvecA = c(3/8, 1/4, 3/12, 1/8)
obsunderA = rmultinom(1000,prob=pvecA,size=20)
obsunderA[,1:12]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]    6    7    3    7    8    5   10    5    8     8    10    11
[2,]    7    7    4    4    4    7    5    9    3     6     3     4
[3,]    5    2   10    7    7    7    4    6    7     3     5     2
[4,]    2    4    3    2    1    1    1    0    2     3     2     3
apply(obsunderA,1,mean)
[1] 7.537 4.946 4.959 2.558
expectedA=pvecA*20
expectedA
[1] 7.5 5.0 5.0 2.5
```

As with the simulation from the null hypothesis, these numbers vary considerably. The question is: how often (out of 1000) instances will our test detect that the data depart from the null?

The test doesn't reject the first observation, (6, 7, 5, 2), because the value of the statistic is within the 95th percentile:

```
sum((expected0-obsunderA[,1])^2/expected0)
[1] 2.8
####Whole series of statistics under the alternative
stat1=apply(obsunderA,2,function(x)sum((x-expected0)^2/expected0))
val
95%
7.6
sum(stat1>val)
[1] 213
power=sum(stat1>val)/1000
power
[1] 0.213
```

Run across 1000 simulations, the test identified 213 as coming from an **alternative distribution** .

We've thus computed that the probability

$$P(\text{reject} \ \ H_0|H_A) = 0.213.$$

With a sequence length of $n = 20$ we have a power of about 21% to detect the difference between the fair generating process and our **alternative** .

In practice, as we mentioned, an acceptable value of power is 0.8 or more.

Need to repeat the simulation experiments and suggest a new sequence length n that will ensure that the power is acceptable.

Multinomial Model for nucleotides: DNA Frequency Calculations

In the Labs:

- Biostrings
- BSgenome: different genomes
- Run a simulation experiment (parametric bootstrap)

```
library(BSgenome.Celegans.UCSC.ce2)
Celegans
Worm genome:
# organism: Caenorhabditis elegans (Worm)
# provider: UCSC
# provider version: ce2
# release date: Mar. 2004
# release name: WormBase v. WS120
# 7 sequences:
#   chrI   chrII  chrIII chrIV  chrV   chrX   chrM
# (use 'seqnames()' to see all the sequence names, use the '$' or '[['
# operator to access a given sequence)
seqnames(Celegans)
[1] "chrI"   "chrII"  "chrIII" "chrIV"  "chrV"   "chrX"   "chrM"
afM=alphabetFrequency(Celegans$chrM)
afM[1:4]
   A    C    G    T
4335 1225 2055 6179
sum(afM[1:4])
[1] 13794
```

```
s1=rmultinom(1,100,p=c(1/4,1/4,1/4,1/4))
s2=rmultinom(1,13794,p=c(1/4,1/4,1/4,1/4))
```

Is this consistent with a uniform (1/4,1/4,1/4,1/4)?

```
mystat=function(exp,obs){
  out=sum((exp-obs)^2/exp)
  return(out) }
exp=c(3448,3449,3448,3449)
obs=afM[1:4]
obs
   A    C    G    T
4335 1225 2055 6179
mystat(exp,obs)
[1] 4385.934
exp=rmultinom(1,13794,p=c(1/4,1/4,1/4,1/4))
mystat(exp,obs)
[1] 4335.121
```

Is this larger than what randomness could explain?

Under the null model we might see an observation of:

```
x=rep(round(13794/4),4)
x=as.vector(x);names(x)=c("A","C","G","T")
rm=rmultinom(1,13794,p=c(1/4,1/4,1/4,1/4))
mystat(exp=x,obs=rm)
[1] 0.2958237
```

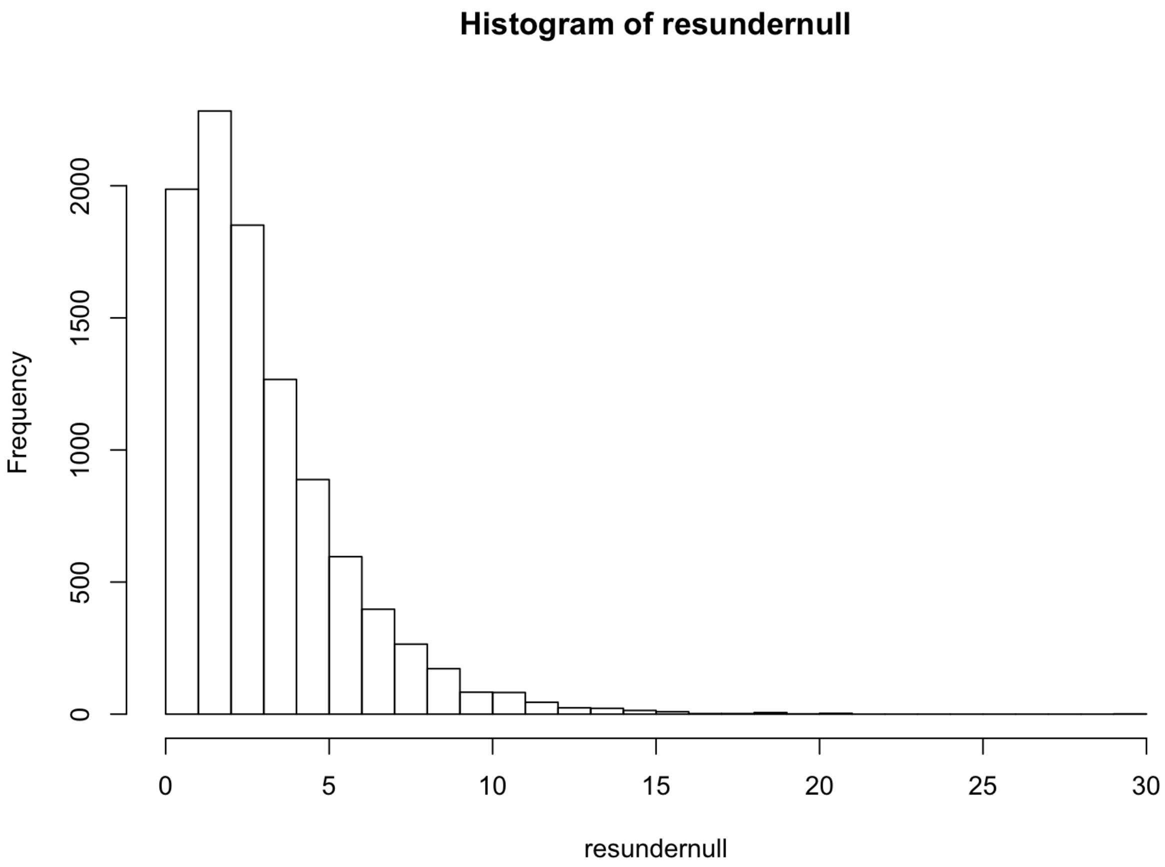
Need a whole set (distribution) of values

```
mystatsundernull=function(x,N=13794,B=1000,p=c(1/4,1/4,1/4,1/4)){
  out=rep(0,B)
  obsmatrix=rmultinom(B,N,p=c(1/4,1/4,1/4,1/4))
  for ( b in 1:B){
    out[b]=mystat(exp=x,obs=obsmatrix[,b])
    }
  return(out)
}

resundernull=mystatsundernull(x,B=10000)
```

Compare this to what we saw

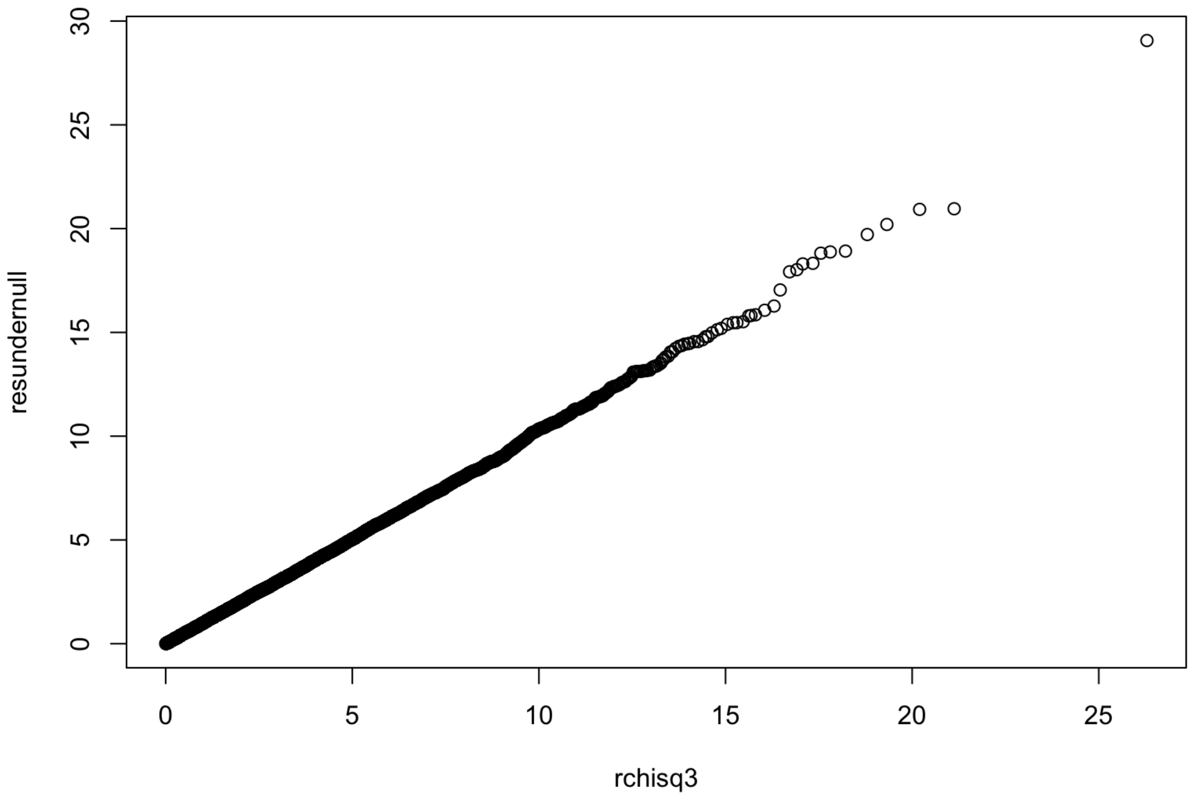
```
hist(resundernull,30)
```



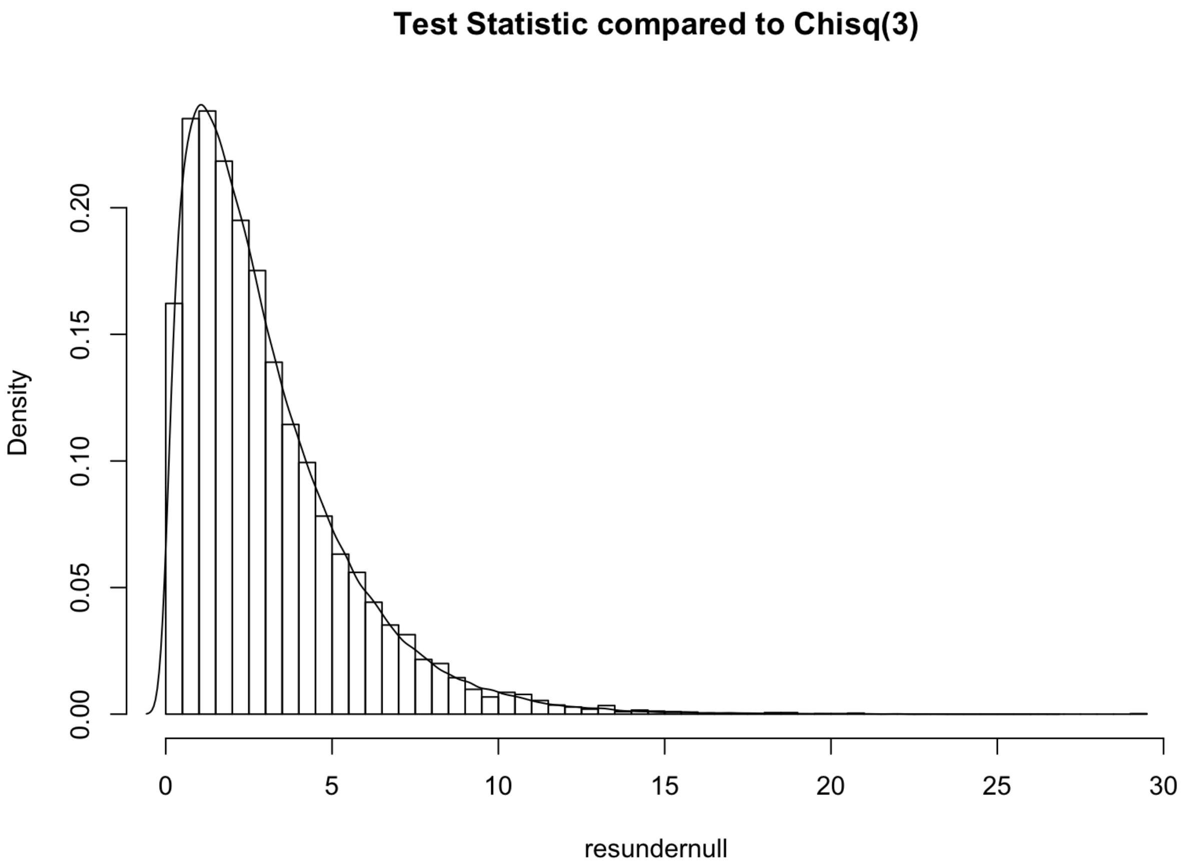
```
mystat(exp,obs)
[1] 4335.121
```

We actually didn’t need to simulate

```
rchisq3=rchisq(100000,3)
qqplot(rchisq3,resundernull)
```



```
hist(resundernull,50,prob=TRUE,main="Test Statistic compared to Chisq(3)")
lines(density(rchisq3))
```


Test Statistic compared to Chisq(3)
Density
0.00
0.05
0.10
0.15
0.20
0
5
10
15
20
25
30
resundernull

Conclusion

Shows that $P_{H_0}($ statistic as large as 4335$)$ smaller than 10^{-4}.

We call H_0 the null hypothesis that the underlying proportions of the multinomial were $(p_A, p_C, p_G, p_T) = (0.25, 0.25, 0.25, 0.25)$

We also write that probability

$P($ Observed stat $>$ 4335 $|H_0)$. We read the vertical line as **given** or **conditional on**.

We rejected the null H_0 that the underlying proportions of the multinomial were $(p_A, p_C, p_G, p_T) = (0.25, 0.25, 0.25, 0.25)$

An application of using the Poisson

When an event is quite rare, like a mutation, the number of times it occurs follows a Poisson distribution.

If we only look at one trial, most of the time, we won't see anything.

For 1000 positions of the RT gene, and each mutation occurs independently at rate 0.02 then we would expect to see 20 mutations ($1000 \times 0.02 = n \times p$).

In general if an event has probability p and and we have n trials, this number is random with a Poisson distribution with mean parameter λ equal to np.

```
sum(rpois(1000,lambda=0.02))
[1] 17
```

If you do this quite a few times, you'll how the values you get differ, we get a distribution of values centered around 20 or so.

Using probabilistic (generative) models for Epitope Detection

When testing certain pharmaceutical compounds it is important to detect proteins that provoke an allergic reaction, the sites that are responsible for such reactions are called epitopes.

The technical definition of an Epitope is: *A specific potion of a macromolecular antigen to which an antibody binds*.

An antibody http://ghr.nlm.nih.gov/ghr/glossary/antibody is type of protein made by certain white blood cells in response to the foreign substance which is called the antigen.

Each antibody can bind to only a specific antigen.

The purpose of this binding is to help destroy the antigen.

Some antibodies destroy antigens directly.

An epitope (or antigenic determinant), is the part of an antigen that is recognized by the immune system

Known Parameters of the Measurements

ELISA detect specific epitopes along proteins.
- The baseline noise level per position is 1% (False positive rate), that is the probability of declaring a hit (that we have an epitope) when it is not there is 0.01. - The length of the protein tested is 100 positions. - We are going to examine a collection of 50 patient samples.

One patient's data

The data for one patient looks like this

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
##  [35] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [69] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

where the 1 signifies a reaction or a `hit` and the zeros signify no reaction at that position.

Results from the 50 assays

Typical of 100 positions tallies of hits for all 50 patient assays.

If the counts follow a Poisson distribution.

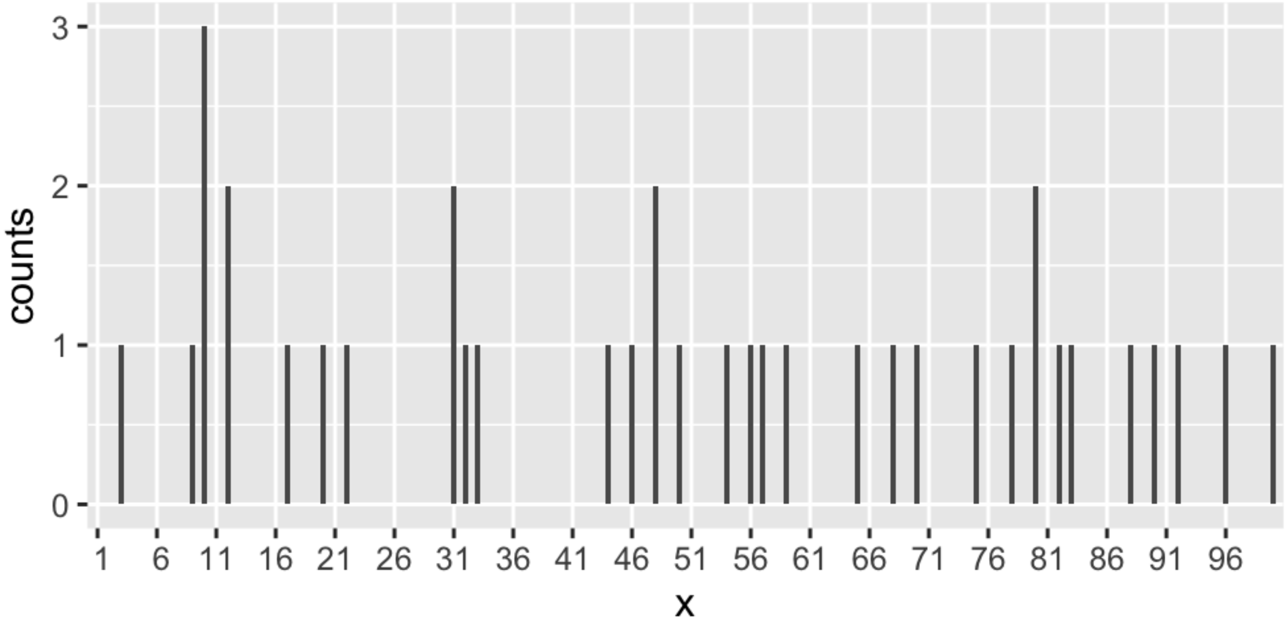
Each individual position and patient has a probability of 1 in 100 of being a 1. At any given position after seeing 50 patients, we expect the sum to have a Poisson distribution with parameter 0.5.

Question: Run a little simulation experiment to show that looking at the sum of 50 Poisson(0.01) variables will be like looking at one Poisson(0.5) random variable.

Answer

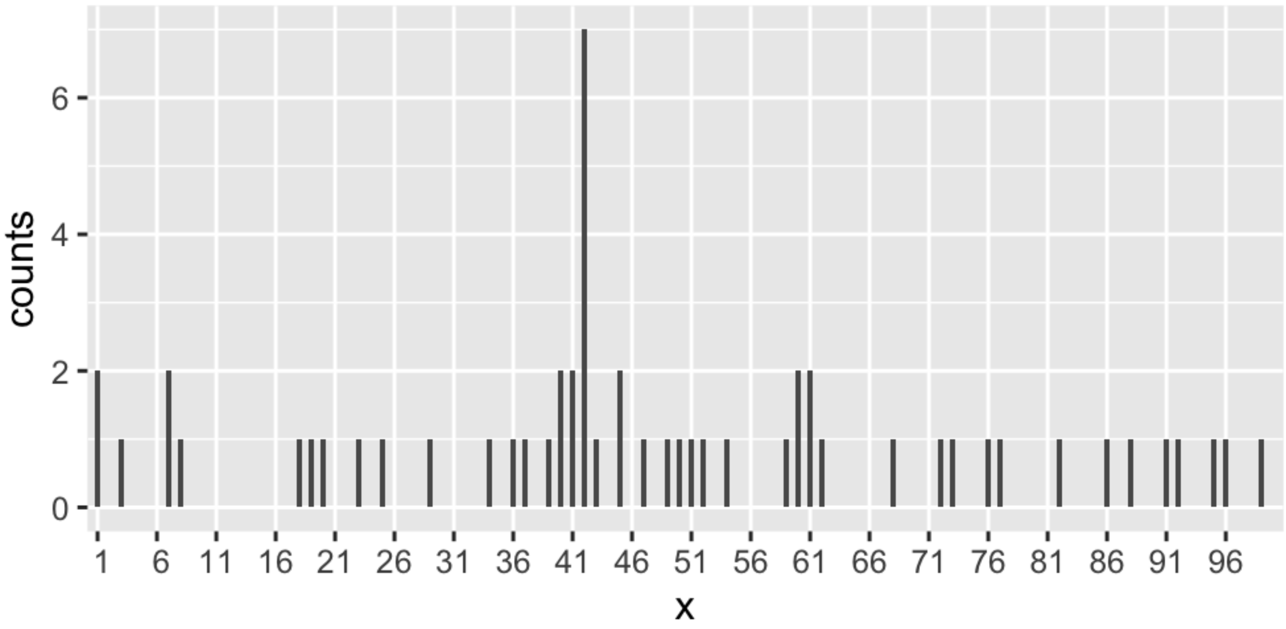
```
x50=sum(rpois(50,0.01))
x50
[1] 0
x50=sum(rpois(50,0.01))
x50
[1] 1
x50=sum(rpois(50,0.01))
x50
[1] 0
x50=sum(rpois(50,0.01))
x50
[1] 0
```

```
set.seed(5427121)
m50=matrix(rpois(5000,0.01),ncol=100,nrow=50)
s100=apply(m50,2,sum)
table(s100)
s100
 0  1  2  3
68 21  7  4
t100=rpois(100,0.5)
table(t100)
t100
 0  1  2  3
68 20 10  2
mean(s100)
[1] 0.47
mean(t100)
[1] 0.46
```



Plot of 100 draws from a Poisson(0.5)

Suppose that the actual data we see is the following:



Output of Elisa array on 50 patients in 100 positions

We see a surprising spike.

What are the chances of seeing a value as large as 7?

If we only look for the probability of seeing a number as big as 7 when considering one Poisson(0.5) random variable.

$$P(X \geq 7) = 1 - P(X \leq 6) = 1 - F_{Pois}(6) = 1 - ppois(6, 0.5)$$

Call this number ϵ, how big is it?

```
##Here are the cumulative function values for 0 to 6
round(ppois(0:6,0.5),5)
[1] 0.60653 0.90980 0.98561 0.99825 0.99983 0.99999 1.00000
##What are the chances of seeing a value strictly bigger than 6?
1-ppois(6,0.5)
[1] 1.00238e-06
```

Is this right?

No:
wrong
computation
in this case.

We looked at all 100 positions and 7 was chosen because it was the maximum, so we have to ask ourselves what are the chances of seeing a number as large in 100 trials.

We use **extreme values** :

Order the values $x_1, x_2, \dots x_n$ renaming them $x_{(1)}, x_{(2)}, x_{(3)} \dots x_{(100)}$ so that $x_{(100)}$ denotes the maximum.

For the largest to be as large as 7 is the **complementary event** (opposite) of having all 100 counts be smaller or equal to 6.

$$P(x_{(100)} \ge 7) = 1 - P(x_{(100)} \le 6) = 1 - \Pi_{i=1}^{100} P(x_i \le 6)$$

$$\Pi_{i=1}^{n} P(x_i < 7) = \left(\sum_{k=0}^{6} \frac{e^{-\lambda} \lambda^k}{k!} \right)^{100} = \left(1 - \sum_{k=7}^{\infty} \frac{e^{-\lambda} \lambda^k}{k!} \right)^{100}$$

Estimating small numbers, tail probabilities

–

Why is this better than by simulation (Monte Carlo) ?

Do not run this as it takes quite a lot of time and memory (on a 4GB machine this computation took about an hour, whereas the small analytic approximation above took 2 minutes)

```
M100=rpois(1000000000,0.5)
table(M100)
Matrix100=matrix(M100,ncol=100)
vecmaxes=apply(Matrix100,1,max)
table(vecmaxes)
```

giving an approximation of $9.48.10^{-5}$ for $P(X_{max} \geq 7)$ and 3.10^{-7} for $P(X_{max} \geq 9)$.

We can however prove that a probability is `smaller than 0.000001` for instance by doing just 10^6 computations.

```
million=rpois(1000000,0.5)
mat100=matrix(million,ncol=100)
maxes100=apply(mat100,1,max)
table(maxes100)

## maxes100
##    1    2    3    4    5    6
##    2 2338 6053 1432  163   12
```

We can only conclude that $P(max \geq 7) < 10^{-4}$. So although it is doable to find small probabilities by simulation it is not always optimal.

Everything we have done up to now was under the probabilistic model possible because we knew the false positive rate per position, we knew the number of patients assayed and the length of the protein, there were no unknown parameters.

Conclusion for this study

We postulated the Poisson distribution for the noise and were able to conclude through mathematical deduction.

This is a **Probability or Generative Model** because all the parameters were known and the mathematical theory allowed us to conclude by **deduction**

Now suppose that we knew the number of patients and the length of the proteins and we observed the data from an unknown distribution then we would have to use **Statistical Modeling** that will be developed in the next lecture.

Important things about Generative Models for Discrete Data

- Several ways to model count data, the Poisson, Binomial and Multinomials.
- Can generate random discrete data using the specialized R functions tailored for each type of distribution.
- The epitope example showed us how to use a probability model to compute a probability under a parametric model when the parameter was known.
- P values are probabilities under certain assumptions.
- Simulations under known models help design better experiments.

$$\begin{tikzcd}
& \mathcal{F}_\theta \arrow[dl, bend right] \\
x \arrow[d, bend right] & \\
\mathcal{P}(x \mid \theta) &
\end{tikzcd}$$

R note about known distributions

All known distributions can be used to simulate data using the functions that start `rXXXX` where `XXXX` could be `pois`, `binom`, `multinom`. If we need a theoretical computation of a probability under one of these models, we would use the functions `dXXXX`, such as `dpois`.

This also works for distributions that are not counts, such the Gaussian (rnorm, dnorm), the χ^2 (rchisq, dchisq) or the exponential (rexp, dexp).

R Note:

Functions are useful for generalizing useful sequences of commands

```
#########Argument by default for the Poisson take 0.5, for protein length take n=100
#########Function to compute the probability of having a maximum out of n as big as max
pmax=function(lam=0.5,n=100,max=7){
epsilon=1-ppois(max-1,lam)
proba=1-exp(-n*epsilon)
return(proba)
}
pmax(lam=0.5)
[1] 0.0001002329
pmax(lam=mean(e100))
[1] 0.0001870183
```