MSMB-Chapter1-GersteinNotes

Aleeza Gerstein

2019-09-16

Chapter 1: Generative models for discrete data

Example where we know the probability model for the process

Mutation in human genome occur at a rate $5x10^{-4}$ per nucleotide per replication cycle. After one cycle, the number of mutations in a genome of size $10^4 = 10{,}000$ nucleotides will follow a *Poisson distribution* with rate 5. Therefore the **probability model** predicts 5 mutations over one replication cycle, and the standard error is $\sqrt{5}$.

Thus the *rate parameter* $\lambda = 5$ and we can generate the probability of seeing three events as:

```
dpois(x = 3, lambda = 5)
```

```
## [1] 0.1403739
```

Generate an entire distribution of all values from 0:12 as

```
.oldopt = options(digits = 2)
dpois(x = 0:12, lambda = 5)
```

```
##  [1] 0.0067 0.0337 0.0842 0.1404 0.1755 0.1755 0.1462 0.1044 0.0653 0.0363
## [11] 0.0181 0.0082 0.0034
```

```
barplot(dpois(0:12, 5), names.arg = 0:12, col = "red")
```

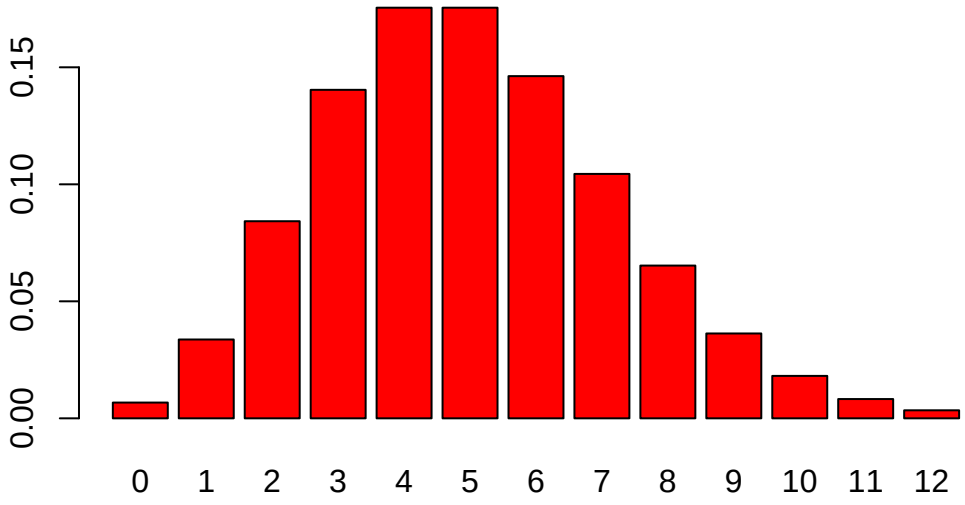


Figure 1: Probabilities of seeing 0,1,2,...,12 mutations, as modeled by the Poisson(5) distribution. The plot shows that we will often see 4 or 5 mutations but rarely as many as 12. The distribution continues to higher numbers (13, ...), but the probabilities will be successively smaller, and here we don't visualize them.

```
options(.oldopt)
```

1.3 Using discrete probability models

There are binary events such as mutations (occurs/does not occur) - a categorical variable with two **levels** Other events can have many different levels.

"When we measure a categorical variable on a sample, we often want to tally the freequencies of the different levels in a vector of counts. R calls these variables **factors**"

Capture the different blood genotypes for 19 subjects in a vector:

```
genotype <- c("AA","AO","BB","AO","OO","AO","AA","BO","BO",
              "AO","BB","AO","BO","AB","OO","AB","BB","AO","AO")
table(genotype)
```

```
## genotype
## AA AB AO BB BO OO
##  2  2  7  3  3  2
```

R automatically detects the factor levels.

```
genotypeF = factor(genotype)

#You an access the levels liek this:
levels(genotypeF)
```

```
## [1] "AA" "AB" "AO" "BB" "BO" "OO"
```

```
table(genotypeF)
```

```
## genotypeF
## AA AB AO BB BO OO
##  2  2  7  3  3  2
```

Question 1.1 What is you want to create a factor that has some levels not yet in your data?

```
genotypeF$levels <- c(levels(genotypeF), "ZZ")
```

"If the order in which the data are observed doesn't matter we call the random variable **exchangeable**. In this case the vector of frequencies is **sufficient** to capture all the relevant information in the data. It is an effective way to summarize/compress the data"

1.3.1 Bernoulli trials

When there are two possible events with (potentially) unequal probabilities.

15 bernoulli trials with success equal to 0.5:

```
rbinom(15, prob = 0.5, size = 1)
```

```
##  [1] 0 1 1 1 0 1 1 1 1 0 1 0 0 0 1
```

Called with **paremeters*; first is n, the number of trials, *prob* is the probability of success, $size = 1$ indicate each individual trial is one toss.

12 bernoulli trials with unequal probility of success:

```
rbinom(12, prob = 2/3, size = 1)
```

```
##  [1] 1 1 0 1 1 0 0 0 1 1 0 1
```

1.3.2 Binomial success counts

If you only care about how many successes are in one category then the order of the trials doesn't matter and you can set the *size* parameter to the number desired and take the sum of the cells in the output vector

```
rbinom(1, prob = 2/3, size = 12)
```

```
## [1] 7
```

Question 1.2: Repeat this function call a number of times. Why isn't the answer always the same?

Beause it is a probability function.

This is also called a random two-box model, when there are only two possible outcomes; we only need to specify p the probability of success since the *complementary* event will occur with probability 1 - p. If the events are independent of each other (*exchangeable*) we only record the number of successes.

The number of successes in 15 bernoulli trials with a probabilty of success 0.3 is called a **binomial* random variable that follows the B(15, 0.3) distribution. To generate samples:

```
set.seed(235569515)
rbinom(1, prob = 0.3, size = 15)
```

```
## [1] 5
```

Question 1.3: Repeat this function call 10 times. What seems to be the most common outcome?

```
i <- 0
trial <- c()
while(i < 11){
  i <- i +1
  trial[i] <- rbinom(1, prob = 0.3, size = 15)
}
  trial
```

```
##  [1] 5 2 2 4 5 3 5 7 2 5 2
```

or

```
.freqoutcome <- (0:15)[which.max(dbinom(0:15, prob = 0.3, 15))]
.freqoutcome
```

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

The complete **probability mass distribution** is available by typing:

```
probabilities = dbinom(0:15, prob = 0.3, size = 15)
round(probabilities, 2)
```

```
##  [1] 0.00 0.03 0.09 0.17 0.22 0.21 0.15 0.08 0.03 0.01 0.00 0.00 0.00 0.00
## [15] 0.00 0.00
```

```r
c(0:15)[which(probabilities==max(probabilities))]
```

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

Barplot of above probabilities

```r
barplot(probabilities, names.arg = 0:15, col = "red")
```

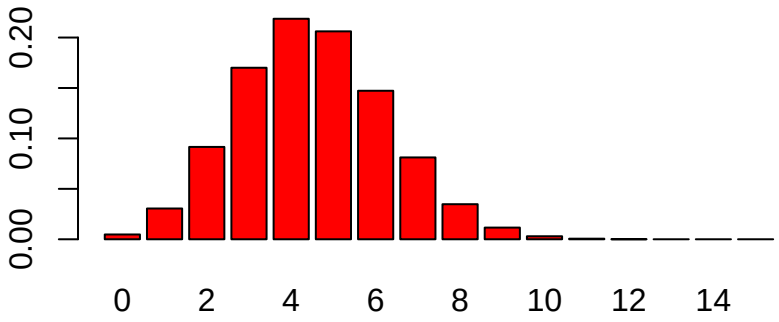


Figure 2: Theoretical distribution of $B(15, 0.3)$. The highest bar is at $x = 4$. We have chosen to represent theoretical values in red throughout.

The number of trial is the *size* parameter and is often written n while the probability of success i p. For X distributed as a binomial distribution with parameters (n, p), written $X \sim B(n, p)$, the probability of seeing $X = k$ success is

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

Question 1.4: What is the output of the formula for k = 3, p = 2/3, n = 4?

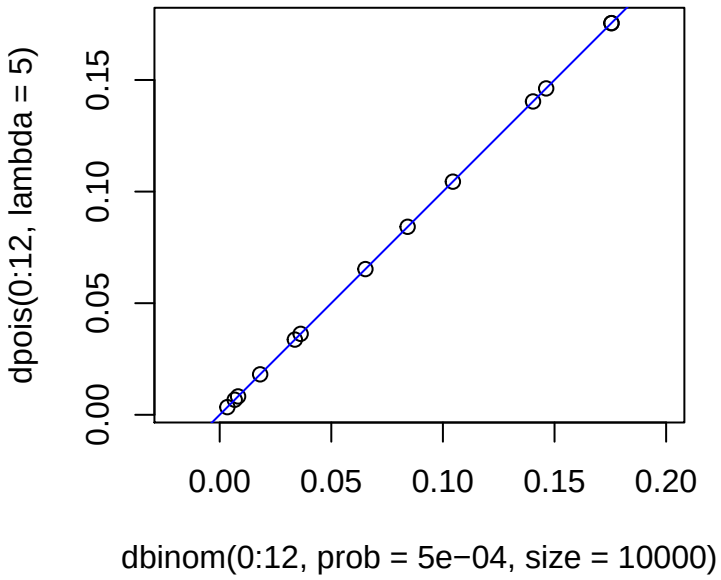
```r
dbinom(3, prob = 2/3, size = 4)
```

```
## [1] 0.3950617
```

1.3.3 Poisson distributions

When the probability of success p is small and the number of trial n is large, the binomial distribution $B(n, p)$ can be faithfully approximated by a simpler distribution, the **poisson distribution** with rate parameter $\lambda = np$.

Question 1.5: what is the probability mass distribution of observing 0:12 mutations in a genome of $n = 10^4$ nucleotides, when the probability of mutation is **p** $= 5 \times 10^{-4}$ per nucleotide? Is it similar when modeled by the binomial $B(n, p)$ distribution and by the poisson $\lambda = np$ distribution?



The Poisson depends only on the product of np. The mathematical formula is: $P(X = k) = \frac{\lambda^k e^{-k}}{k!}$

For instance, take $\lambda = 5$ and compute $P(X = 3)$

```
5^3 * exp(-5) / factorial(3)
```

```
## [1] 0.1403739
```

**

\textcolor{red}{Task: simulate a mutation process along 10, 000 positions with a mutation rate of 5×10^{-4} and count the number of mutations. Repat this many times and plot the distribution.

```
rbinom(1, prob = 5e-4, size = 10000)
```

```
## [1] 3
```

```
simulations = rbinom(n = 300000, prob = 5e-4, size = 10000)
barplot(table(simulations), col = "lavender")
```

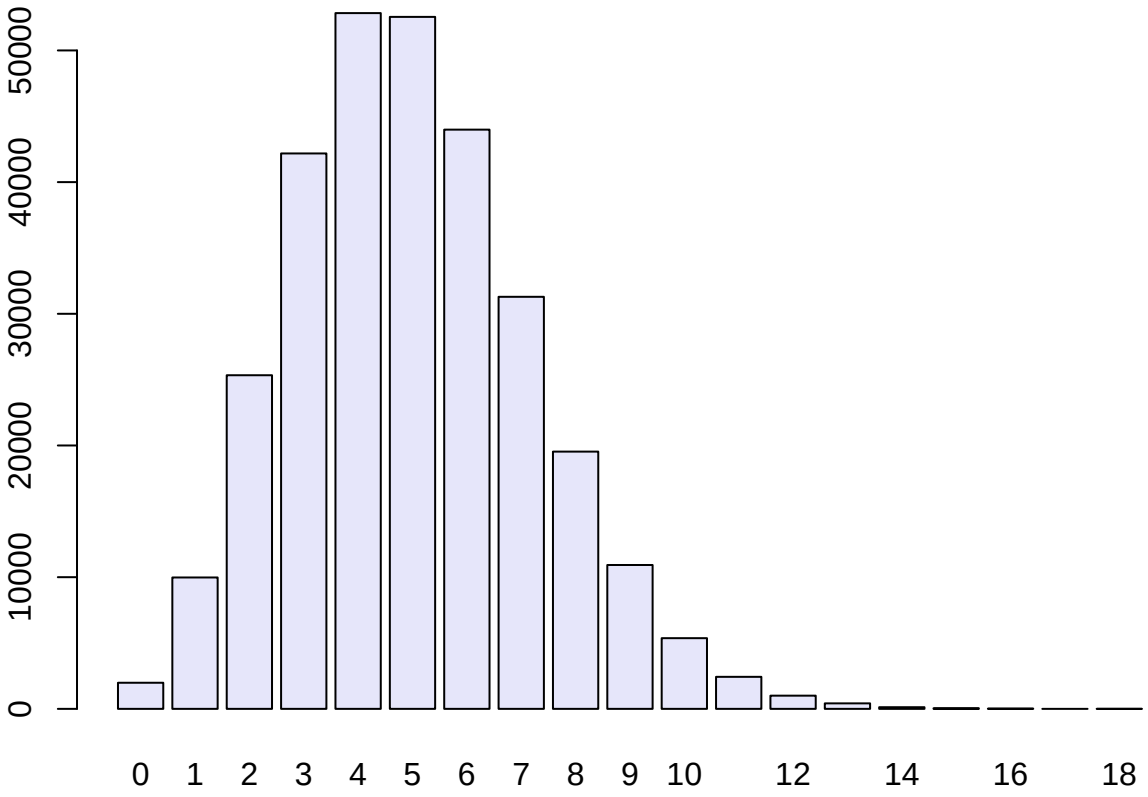


Figure 3: Simulated distribution of B(10000, 10^{-4}) for (ref:gen-simpoisson-1) simulations.

1.3.4 A generative model for epitope detection

ELISA assay the detects epitopes at 100 different independent positions. The false positive rate is 1% We can say that p(declare epitope | no epitope) = 1% or, declaring an epitope when there is no epitope is 1%. The general form is *$X|Y$ as "given" or "conditional on". Thus, "X happens conditional on Y being the case"."* . The data for one patien looks like this:

WHAT IS THIS NOMENCLATURE?

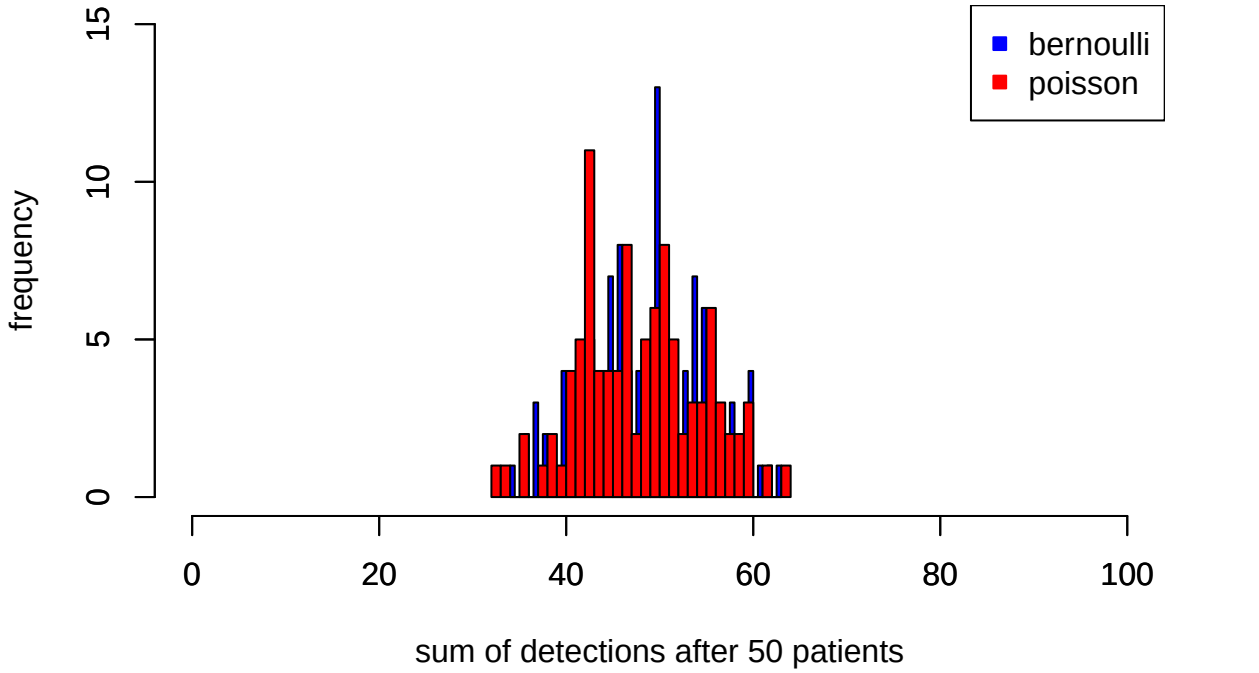
```
P1 <- `[<-` (rep(0, 100), 22, 1)
```

Task: Verify by simulation that the sum of 50 independent Bernoulli variables with p = 0.01 is - to good enough approximation - the same as a Poisson(0.5) random variable

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.

```
bernoul <- c()
poiss <- c()
for(i in 1:100){
  bernoul[i] <- sum(rbinom(n = 100, prob = 0.01, size = 50))
  poiss[i] <- sum(rpois(100, lambda = 0.5))
}
```

```
hist(bernoul, xlim=c(0, 100), breaks=50, col="blue", main="", ylim=c(0, 15), xlab ="sum of detections a
legend("topright", col=c("blue", "red"), legend=c("bernoulli", "poisson"), pch=22, pt.bg =c("blue", "re
par(new=T)
hist(poiss, breaks=40, col="red", xlim=c(0, 100), main="", ylim=c(0, 15), xlab="", ylab="")
```



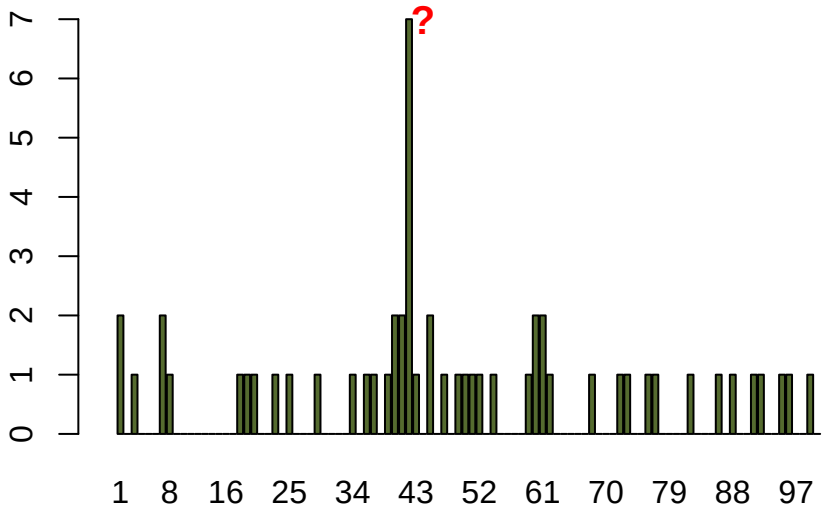


Figure 4: Output of the ELISA array results for 50 patients in the 100 positions.

What are the chances of seeing a value as large as 7, if no epitope is present?

If you look for the probability of seeing a number as big as 7 (or larger) when considering one Poisson(0.5) random variable, you can calculate the closed form as $\sum_{k=7}^{\infty} P(X = k)$

This is the same as 1 - P(X <= 6).

P(X <= 6) is the **cumulative distribution function** at 6 and we use the function `ppois` for computing it.

```r
1 - ppois(6, 0.5)
```

```
## [1] 1.00238e-06
```

```r
ppois(6, 0.5, lower.tail = FALSE)
```

```
## [1] 1.00238e-06
```

Task: What is the meaning of 'lower.tail'? "Setting lower.tail = FALSE allows to get much more precise results when the default, lower.tail = TRUE would return 1, see the example below."

```r
1 - ppois(10*(15:25), lambda = 100)  # becomes 0 (cancellation)
```

```
##  [1] 1.233094e-06 1.261664e-08 7.085799e-11 2.252643e-13 4.440892e-16
##  [6] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [11] 0.000000e+00
```

```r
ppois(10*(15:25), lambda = 100, lower.tail = FALSE)  # no cancellation
```

```
##  [1] 1.233094e-06 1.261664e-08 7.085800e-11 2.253110e-13 4.174239e-16
##  [6] 4.626179e-19 3.142097e-22 1.337219e-25 3.639328e-29 6.453883e-33
## [11] 7.587807e-37
```

?

(1.1) We use $\epsilon = P(X \geq 7) = 1 - P(X \leq 6) \approx 10^{-6}$ (1.1)

** Extreme value analysis for the poisson distirubiton **

Question 1.6: The above calculation is not correct if we want to compute the probability that we observe these data if there is no epitope

We looked at all positions to find the maximum (7), which is more likely to occur than if we looked at only one position. So instead of asking what are the chances of seeing a Poisson(0.5) as large as 7, we should ask what are the chances that the maximum of 100 Poisson(0.5) trials is as large as 7. Use **extreme value theorem** Order the data $x_1, x_2, ..x_{100}$ so that x_1 is the smallest value and x_{100} is the largest of the counts over the 100 positions. Together this set $x_1, x_2, ..x_{100}$ is called the **rank statistic** of this sample of 100 values.

Then the maximum value being as large as 7 is the *complementary event* of having all 100 counts be smaller or equal to 6. Two *complementary events* have probabilities that sum to 1.

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

Because we suppose these 100 events are independent, we can use our result from 1.1 above:

huh.

$$\Pi_{i=1}^{100} P(x_i \leq 6) = (P(x_i \leq 6))^{100} = (1 - \epsilon)^{100}$$

Actually calculating the numbers

Computing probabilities by simulation

Use a *monte carlo* method: a computer simulation based on our generative model that finds the probabilities of the events we're interested in. Essentially generate Figure 1.7 again and again and see how often the biggest spike is 7 or greater.

```r
maxes = replicate(100000, {
  max(rpois(100, 0.5))
})
table(maxes)
```

```
## maxes
##     1     2     3     4     5     6     7     9
##     8 23072 60805 14354  1604   141    15     1
```

```r
mean( maxes >= 7 )
```

```
## [1] 0.00016
```

That's the same as the number of times maxes was 7 or greater divided by 100000. Potential limitation of Monte Carlo simulations is that the inverse of the number of simulations is the 'granularity'. What we have done is an example of **probability or generative modeling**: all parameters are known and the theory allows us to work by **deduction** in a top-down fashion.

If we knew instead the number of patients and the length of the proteins but did not know the distribution of the data then we have to use **statistical modeling**. If we only have the data then we **fit** a reasonable distirubiton to describe it.

1.4 Multinomial distributions: the case of DNA

More than two outcomes

\textcolor{blue}{Task: Experiment with the random number generator that generates all possible numbers between 0 and 1 through the function called `runif`. Use it to generate random variable with four levels (A, C, T, G) where $p_A = \frac{1}{8}$, $p_B = \frac{3}{8}$, $p_C = \frac{3}{8}$, $p_G = \frac{1}{8}$

```r
?sample
```

Mathematical formulation

Question 1.7: Suppose we have four boxes that are equally likely. Using the formula, what is the probability of observing four in the first box, two in the second box and none in the two other boxes?

```r
dmultinom(c(4, 2, 0, 0), prob = rep(1/4, 4))
```

```
## [1] 0.003662109
```

Formula?

Run simulations to test whether the data are consistent with each box having the same probability. Suppose eight characters of four differentm equally likely types:

```r
pvec = rep(1/4, 4)
t(rmultinom(1, prob = pvec, size = 8))
```

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

Question 1.9: How do you interpret the difference between 'rmultinom(n=9, prob=pvec, size =1)' and 'rmultimon(n=1, prob = pvec, size =8)'?

n is the number of random draws size is the number of objects that are put into K boxes

```
t(rmultinom(n=8, prob=pvec, size =1)) #8 trials with one ball thrown into 1 of 4 boxes
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    1    0    0
## [2,]    0    1    0    0
## [3,]    0    0    0    1
## [4,]    0    1    0    0
## [5,]    0    0    0    1
## [6,]    1    0    0    0
## [7,]    0    0    1    0
## [8,]    0    1    0    0
```

```
t(rmultinom(n=1, prob = pvec, size =8)) #one trial with 8 balls thrown into 4 boxes
```

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

1.4.1 Simulating for Power

Important Q: How big a sample size do I need?

Power is the probability of detecting something if it is there, i.e., the *true positive rate* Conventionally want power of $\geq 80\%$. i.e., if the same experiment is run many times, about 20% of the time it will fail to yield significant results even it it should.

Generate 1000 simulations from the null hypothesis using the `rmultinom` function. Display only the first 11 columns.

```
obsunder0 = rmultinom(1000, prob = pvec, size = 20)
dim(obsunder0)
```

```
## [1]    4 1000
```

```
obsunder0[, 1:11]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## [1,]    7    5    3    3    2    4    6    7    4     6     4
## [2,]    4    4    8    4    5    6    6    4   10     2     7
## [3,]    3    5    7   10    7    3    3    4    2     9     6
## [4,]    6    6    2    3    6    7    5    5    4     3     3
```

Eah column is a simulated instance. Teh numbers vary a lot. The expected value is $20/4 = 5$ but in some cases it goes up much higher.

Creating a test

Knowing the expected value (1/4) isn't enough. We also need a measure of **variability** to describe how much variability is expected and how much is too much. Use the following statistic that is computed as the sum of squares of the squre of the differences between the observed calue and the expected value relative to the expected value. I.e., χ-square.

stat $= \sum_i = \frac{(E_i - x_i)^2}{E_i}$

How do the first three columns of the generated data differ from what we expect?

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

```
## [1] 2
```

```
sum((obsunder0[, 2] - expected0)^2 / expected0)
```

```
## [1] 0.4
```

```
sum((obsunder0[, 3] - expected0)^2 / expected0)
```

```
## [1] 5.2
```

The values differ. Encapsulate the formula for `stat` in a fumction.

```
stat = function(obsvd, exptd = 20 * pvec) {
  sum((obsvd - exptd)^2 / exptd)
}
stat(obsunder0[, 1])
```

```
## [1] 2
```

Then compute the measure for all 1000 instances and call it $S0$ which contains values generated under H_0. COnsider the histogram of $S0$ to be the *null distribution*.

```
S0 = apply(obsunder0, 2, stat)
summary(S0)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.200   2.600   2.904   4.000  14.000
```

```
hist(S0, breaks = 50, col = "lavender", main = "")
```

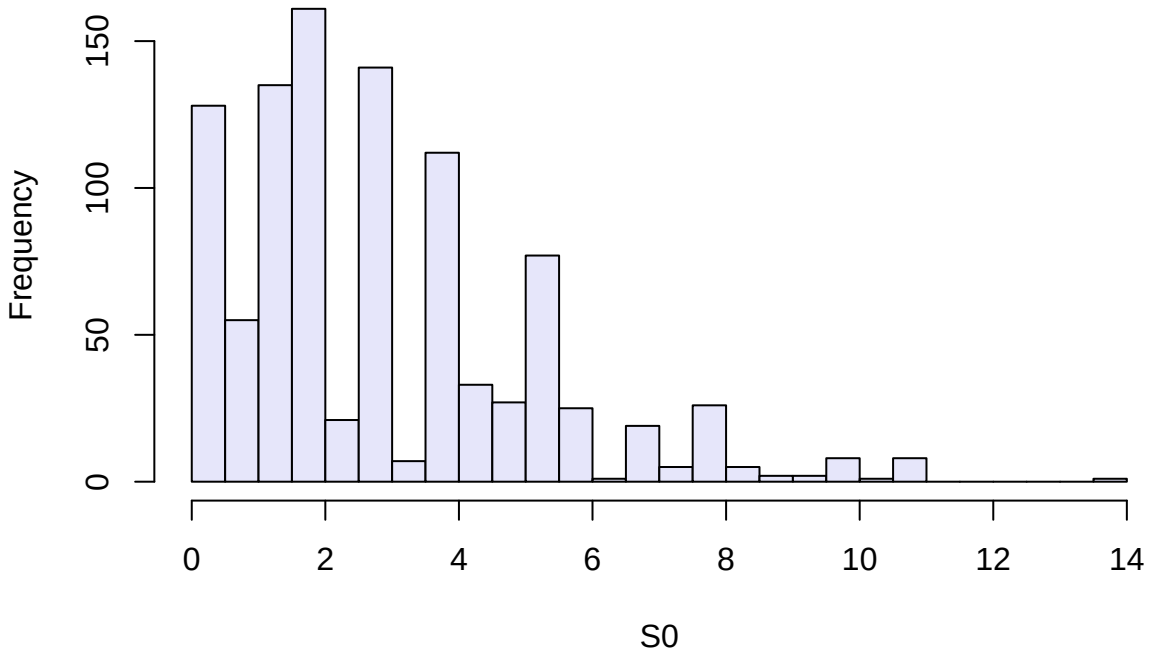


Figure 5: The histogram of simulated values `S0` of the statistic `stat` under the null (fair) distribution provides an approximation of the **sampling distribution** of the statistic `stat`.

From this can approximate the 95% quantile:

```
q95 = quantile(S0, probs = 0.95)
q95
```

```
## 95%
## 7.6
```

That tells us that 5% of the values are larger than 7.6. This becomes our criteria for testing: we wil reject the null hypothesis that the data come from a fair proess if the weighted sum of squares 'stat' is larger than 7.6.

Determine the test power Now compute the probability that our test will detect data that do not come from the null hypothesis.

```
pvecA = c(3/8, 1/4, 3/12, 1/8)
observed = rmultinom(1000, prob = pvecA, size = 20)
dim(observed)
```

```
## [1]    4 1000
```

```
observed[, 1:7]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]   10    6    6    8    6    6   10
## [2,]    6    8    3    3    5    4    4
## [3,]    3    5    7    6    8    6    5
## [4,]    1    1    4    3    1    4    1
```

```
apply(observed, 1, mean)
```

```
## [1] 7.405 4.982 5.088 2.525
```

```
expectedA = pvecA * 20
expectedA
```

```
## [1] 7.5 5.0 5.0 2.5
```

```
stat(observed[, 1])
```

```
## [1] 9.2
```

```
S1 = apply(observed, 2, stat)
q95
```

```
## 95%
## 7.6
```

```
sum(S1 > q95)
```

```
## [1] 209
```

```
power = mean(S1 > q95)
power
```

```
## [1] 0.209
```

Thus with a sequence length of 20 we have a power of about 20% to detect difference between a fair generating process and our **alternative**.

Suggest a new sequence length that will ensure the power is acceptable

```
for(i in 20:30){
  expectedA = pvecA * i
```

```
  observed = rmultinom(1000, prob = pvecA, size = i)
  S1 = apply(observed, 2, stat)
  power = mean(S1 > q95)
  print(paste0(i ," = ", power))
}
```

```
## [1] "20 = 0.235"
## [1] "21 = 0.265"
## [1] "22 = 0.318"
## [1] "23 = 0.373"
## [1] "24 = 0.465"
## [1] "25 = 0.499"
## [1] "26 = 0.591"
## [1] "27 = 0.702"
## [1] "28 = 0.798"
## [1] "29 = 0.84"
## [1] "30 = 0.911"
```

{r ProbaDiagram, eval = TRUE, echo = FALSE, fig.show = 'hold', fig.keep = 'high', fig.cap = "We have studied how a probability model has a distribution we call this F. F often depends on parameters, which are denoted by Greek letters, such as $\\theta$. The observed data are generated via the brown arrow and are represented by Roman letters such as x. The vertical bar in the probability computation stands for **supposing that** or **conditional on**"---- knitr::include_graphics(c('images/ProbaDiagram.png'))

```
dbinom(2, size = 10, prob = 0.3)
```

```
## [1] 0.2334744
```

```
pbinom(2, size = 10, prob = 0.3)
```

```
## [1] 0.3827828
```

```
sum(sapply(0:2, dbinom, size = 10, prob = 0.3))
```

```
## [1] 0.3827828
```

```
poismax(lambda = 0.5, n = 100, m = 7)
```

```
## [1] 0.0001002329
```

```
poismax(lambda = mean(e100), n = 100, m = 7)
```

```
## [1] 0.0001870183
```

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(c("Biostrings", "BSgenome.Celegans.UCSC.ce2"))
```

```
library("BSgenome.Celegans.UCSC.ce2")
Celegans
seqnames(Celegans)
Celegans$chrM
class(Celegans$chrM)
length(Celegans$chrM)
```

```
library("Biostrings")
lfM = letterFrequency(Celegans$chrM, letters=c("A", "C", "G", "T"))
lfM
```

```r
sum(lfM)
lfM / sum(lfM)
```

```r
t(rmultinom(1, length(Celegans$chrM), p = rep(1/4, 4)))
```

```r
length(Celegans$chrM) / 4
```

```r
oestat = function(o, e) {
  sum((e-o)^2 / e)
}
oe = oestat(o = lfM, e = length(Celegans$chrM) / 4)
oe
```

```r
B = 10000
n = length(Celegans$chrM)
expected = rep(n / 4, 4)
oenull = replicate(B,
  oestat(e = expected, o = rmultinom(1, n, p = rep(1/4, 4))))
```

Question 1.1

```r
genotype <- c("AA","AO","BB","AO","OO","AO","AA","BO","BO",
              "AO","BB","AO","BO","AB","OO","AB","BB","AO","AO")
#How to create a factor that has some levcels not yet in your data?
genotypeF = factor(genotype, levels = c(unique(genotype), "CC"))
levels(genotypeF)
```

```
## [1] "AA" "AO" "BB" "OO" "BO" "AB" "CC"
```

```r
table(genotypeF)
```

```
## genotypeF
## AA AO BB OO BO AB CC
##  2  7  3  2  3  2  0
```

Question 1.4

```r
dbinom(3, prob = 2/3, size = 4)
```

```
## [1] 0.3950617
```

Question 1.5

```r
probabilities_Q5_binom <- dbinom(0:12, prob = 5E-4, size = 1E4)
probabilities_Q5_poisson <- dpois(0:12, lambda = 5E-4*1E4)
```