Generative Models for Discrete Data

Brandon Kozak

2019-09-18

```
library(tidyverse)
library(BiocManager)
library(Biostrings)
library(BSgenome.Celegans.UCSC.ce2)
```

Questions we will answer in this chapter

- Given a certain model, how can we obtain the probabilities for all possible outcomes?
- How do theoretical frequencies compare with those observed in real data?
- How can we use the Poisson distribution to analyse data on epitope detection?
- How can we apply the Multinomial distribution and Monte Carlo simulation to perform power tests?

Our first example

Let X be the number of mutations along a genome of HIV.

We are told that mutations occur at a rate of .00005 per nucleotide per replication cycle. Furthermore we assume that this genome contains 10000 nucleotides per cycle.

Thus, for one cycle, X ~ Poisson(λ = .00005 * 10000 = 5)

That's cool and all, but what does this tell us about our mutations?

Quite a lot actually! For example:

- We can expect that in the long run, each cycle will contain 5 mutations (aka the expected value of X)
- We can quantify the spread of the distribution of X in two ways
  - The variance, which is equal to $\sqrt{5}$ in our example
  - The standard deviation, which is equal to 5 in our example
  - Note that the standard deviation is just the square root of the variance.

Moreover, for any value(s) of X we can find the several probabilities:

- The probability of seeing exactly x mutations in a cycle
- The probability of seeing at least x mutations in a cycle
- The probability of seeing no more than x mutations in a cycle
- The probability of seeing between x1 and x2 mutations in a cycle (x1 > x2)

In general, a function $f : A \to [0, 1]$ that takes an integer as input (in the case of discrete data) and returns a probability as output is called a **probability mass function (PDF)**. In particular, we obtain the probability of the observable X being observed to have a value of x, denoted $P(X = x)$.

Given this, lets try some examples in R.

To obtain $P(X = x)$ where X follows a Poisson(λ) distribution, we use the function called dpois(x, lambda)

For example, what is the probability that we observe exactly 5 mutations in our genome?

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

```
## [1] 0.1754674
```

If we want to obtain the probability for multiple values, say 0 through 20, we can set the first parameter (x) to be a vector.

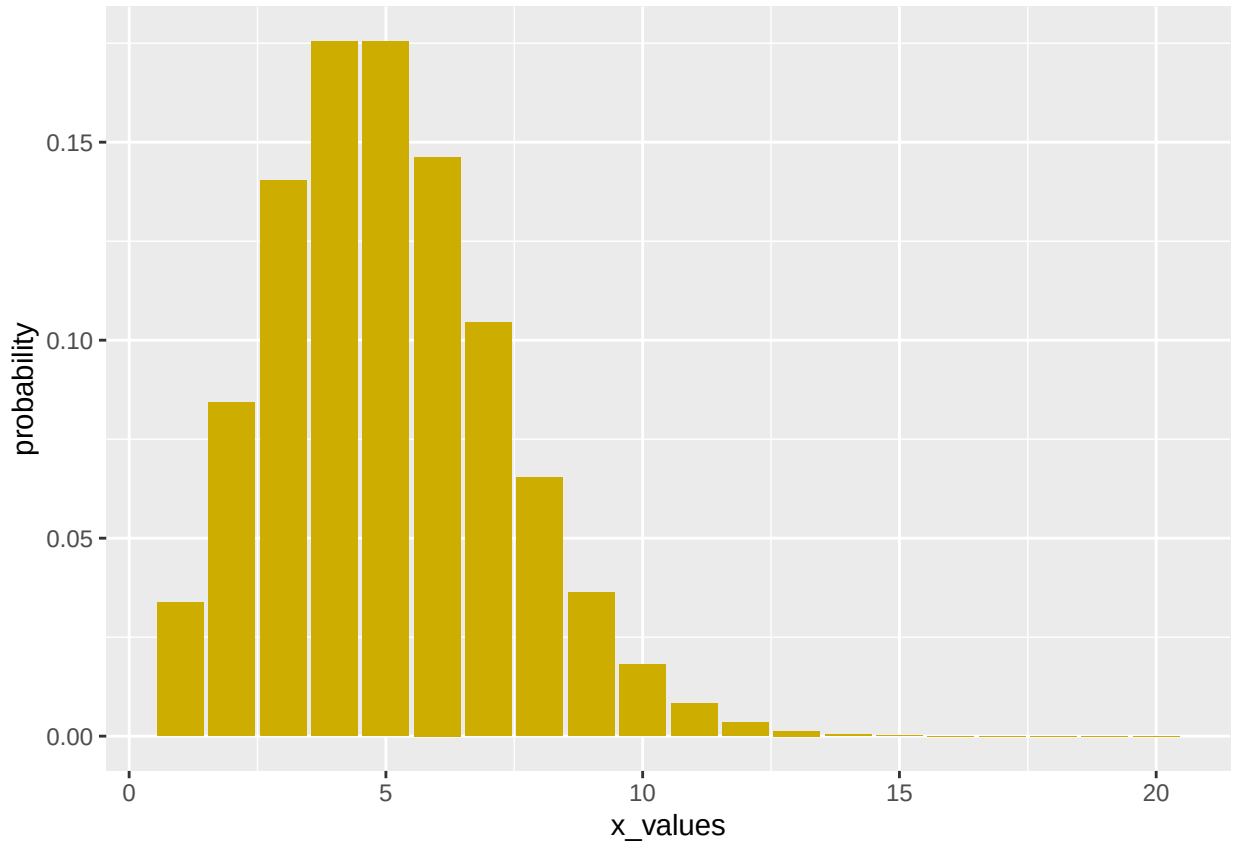
In R, we can use the colon (:) operator to generate a sequence of integers, i.e. 1:3 = (1,2,3).

```
dpois(x = 0:20, lambda = 5)
```

```
##  [1] 6.737947e-03 3.368973e-02 8.422434e-02 1.403739e-01 1.754674e-01
##  [6] 1.754674e-01 1.462228e-01 1.044449e-01 6.527804e-02 3.626558e-02
## [11] 1.813279e-02 8.242177e-03 3.434240e-03 1.320862e-03 4.717363e-04
## [16] 1.572454e-04 4.913920e-05 1.445271e-05 4.014640e-06 1.056484e-06
## [21] 2.641211e-07
```

We could use this to plot the pmf over certain values.

```
tibble(x_values = 1:20, probability = dpois(x = 1:20, lambda = 5)) %>%
  ggplot(aes(x = x_values, y = probability)) +
    geom_bar(stat = "identity", fill = "gold3")
```



Note that after some math we can show that the pmf of a Poisson distribution is $\frac{e^{-\lambda}\lambda^x}{x!}$

Factors

In this section we will discuss factors and how they work in R.

With discrete data, we often associate labels with certain values. For example:

- Sex: Male, Female
- Pain Level: Low, Medium, High
- Blood Genotypes: AA, AB, AO, BB, BO, OO
- Binary outcomes: 0 = No, 1 = Yes

We use the function factor() to convert a vector to a factor vector.

```
pain_level = c("low", "low", "medium", "high", "low", "medium", "high", "high") %>% factor()

pain_level
```

```
## [1] low    low    medium high   low    medium high   high
## Levels: high low medium
```

You'll notice that by default R sorts factors alphabetically, to fix this we set a parameter "levels" in side the call to factor()

```
pain_level = c("low", "low", "medium", "high", "low", "medium", "high", "high") %>%
  factor(levels = c("low", "medium", "high"))

pain_level
```

```
## [1] low    low    medium high   low    medium high   high
## Levels: low medium high
```

Question

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

Solution

We would make the same call to factor and include the extra factors while setting "levels"

```
pain_level = c("low", "low", "medium", "high", "low", "medium", "high", "high") %>%
  factor(levels = c("none", "low", "medium", "high"))

pain_level[1] = "none"

pain_level
```

```
## [1] none   low    medium high   low    medium high   high
## Levels: none low medium high
```

Bernoulli Trials

Consider a binary event with two possible outcomes, say whether a new born baby is male or female. Further assume that each sex is equally likely and that 0 = male, and 1 = female.

If X represents the sex of a random new born, then we can say X follows a Bernoulli distribution with probability p = .5.

Note that a Bernoulli trial with probability p is equivalent to a binomial distribution with size n and probability p. We will talk more about the binomial distribution shorty.

Lets start by generating random samples from a Bernoulli trial with p = .5. To do this, we use the rbinom() function.

Say we wanted to simulated 20 births.

```
rbinom(n = 20, prob = .5, size = 1)
```

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

Distributions in base R

At this point we've seen two ways to work with distributions in r (dpois() and rbinom()). R is actually very systematic about this and has a set of 4 functions for all the common distributions. Each function will start with one of (d, p, q, or r) and will end with an abbreviation of the name of the distribution (pois for Poisson, binom for binomial, norm for normal, etc. . .).

Here, I will state what each letter represents and how the functions work:

- d: probability distribution function, it gives $P(X = x)$ under a certain distribution.
- p: cumulative distribution function, it gives $P(X \leq x)$ under a certain distribution.
- q: quantile function, it gives the value of x for which the cumulative probability equals p under a certain distribution.
- r: random generation, it gives a random sample of size n under a certain distribution.

We will see more examples of theses functions throughout the class.

The Binomial Distribution

Exercises

1.1

Geometric Distribution:

Given a probability p, how many failures will it take to see the first success?

```
# A random sample of size 5 from a geometric distribution with p=.25
rgeom(5, .25)
```

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

```r
# What is the probability that we will see 4 failures before the first success?
dgeom(4, .25)
```

```
## [1] 0.07910156
```

```r
# What is the probability that we will see no more than 3 failures before the first success?
pgeom(3, .25)
```

```
## [1] 0.6835938
```

Hypergeometric Distribution:

Given a population of size N where K of the N objects are "success states." How many success state objects will I obtain from drawing a sample of size n without replacement?

```r
# A random sample of size 5 from a hyper geometric distribution with a population of N=25, K=5 success
# given a sample of n=10
rhyper(5, 5, 20, 10)
```

```
## [1] 2 1 1 3 3
```

```r
# What is the probability that we will see 5 success state objects?
dhyper(5, 5, 20, 10)
```

```
## [1] 0.004743083
```

```r
# What is the probability that we will see at least 1 success state object?
phyper(0, 5, 20, 10, lower.tail = F)
```

```
## [1] 0.9434783
```

1.2

P(X = 2 | X ~ Bin(10, .3))

```r
dbinom(x = 2, size = 10, p = .3)
```

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

P(X <= 2 | X ~ Bin(10, .3))

```r
# Using only dbinom()


dbinom(x = 0, size = 10, p = .3) + dbinom(x = 1, size = 10, p = .3) + dbinom(x = 2, size = 10, p = .3)
```

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

```r
# Using pbinom()
pbinom(q = 2, size = 10, p = .3)
```

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

1.3

```r
pois_max = function(n, max, lamda) {
  # First calculate P(X >= max) = 1 - P(X <= max - 1)
  prob = ppois(max-1, lamda)
  # Then, as we showed before using order statistics, calculate P(X(n) >= max) = P(X(n) <= max-1)
  prob_max = 1 - prob^n
  return(prob_max)
}
```

1.4

```r
pois_max = function(n = 100, max = 0, lamda = 1) {
  # First calculate P(X >= max) = 1 - P(X <= max - 1)
  prob = ppois(max-1, lamda)
  # Then, as we showed before using order statistics, calculate P(X(n) >= max) = P(X(n) <= max-1)
  prob_max = 1 - prob^n
  return(prob_max)
}
```

1.5

```r
# Real answer
pois_max(100, 9, .5)
```

```
## [1] 3.43549e-07
```

```r
# Simulation

pois_has_max = function(n, max, lamda) {

result_vector = rpois(n, lamda)

return(max(result_vector >= max))
}

a = replicate(1e5, pois_has_max(100, 9, .5))
mean(a)
```

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

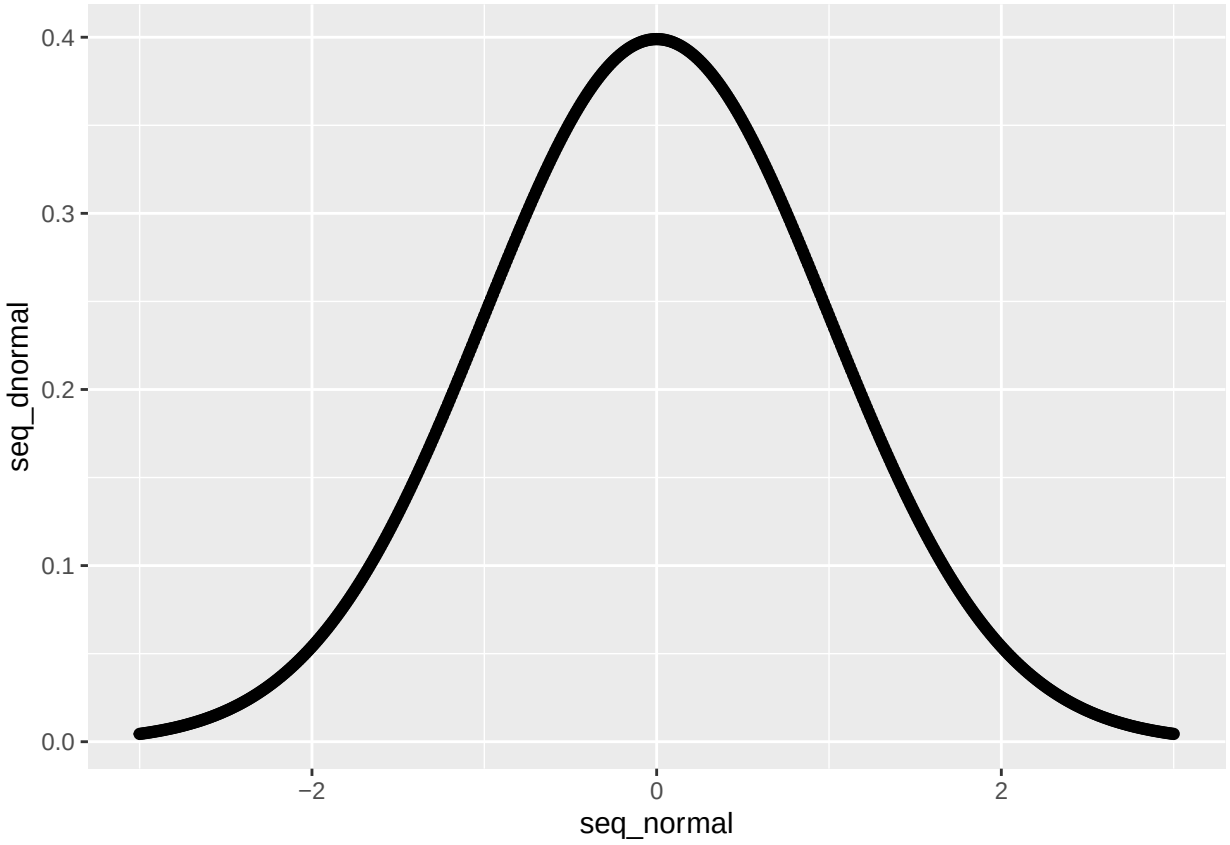
1.6

```
# Standard normal

seq_normal = seq(-3, 3, .01)

seq_dnormal = dnorm(seq_normal, 0, 1)

qplot(x = seq_normal, y = seq_dnormal)
```

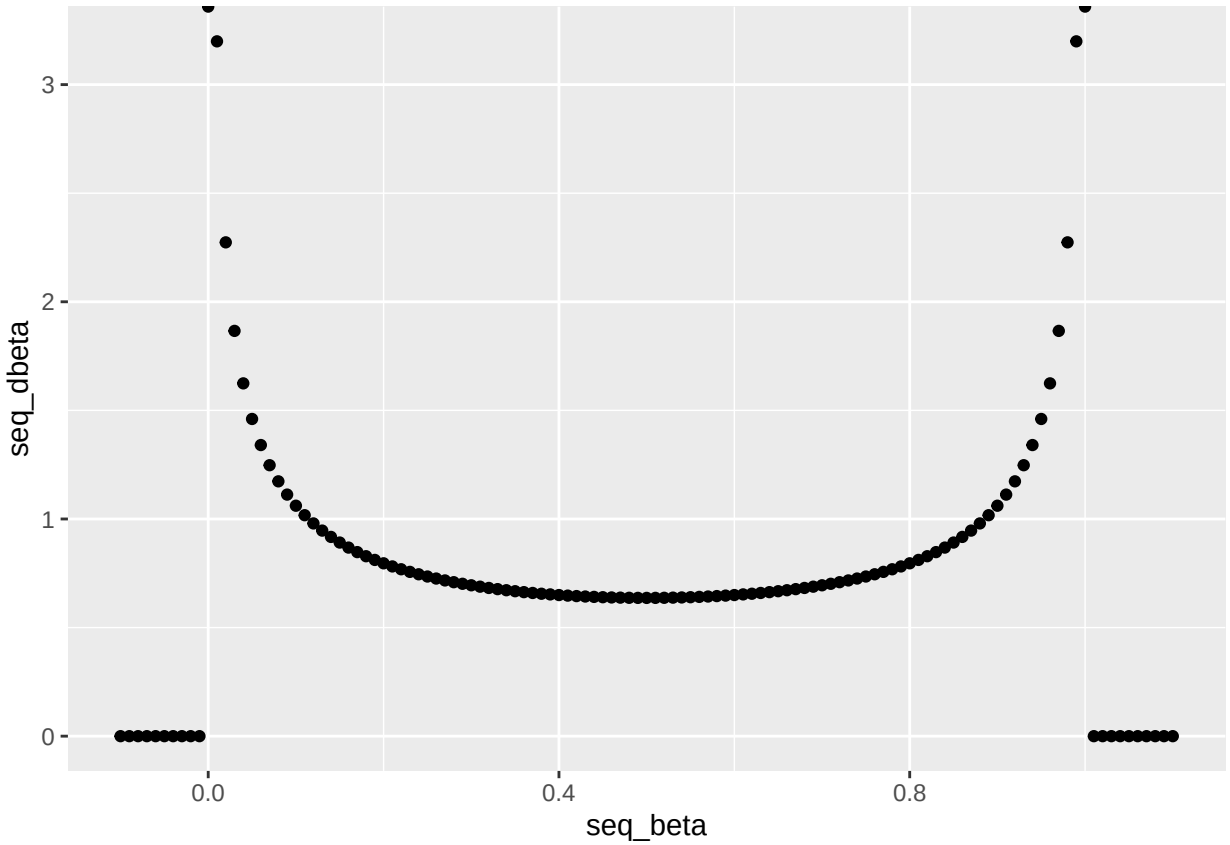



```
# Beta(.5,.5)

seq_beta = seq(-.1, 1.1, .01)

seq_dbeta = dbeta(seq_beta, .5, .5)

qplot(x = seq_beta, y = seq_dbeta)
```

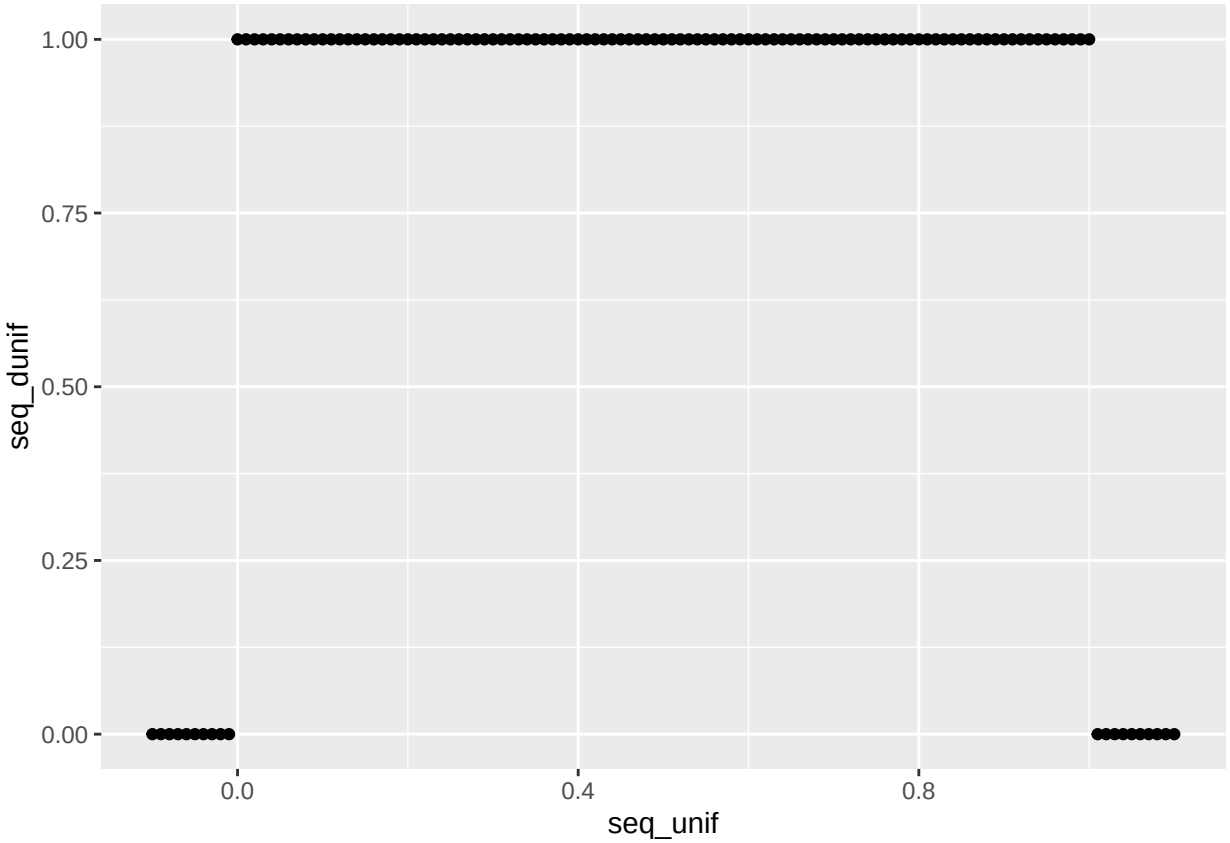



```
# Uniform (0,1)

seq_unif = seq(-.1, 1.1, .01)

seq_dunif = dunif(seq_unif, 0, 1)

qplot(x = seq_unif, y = seq_dunif)
```

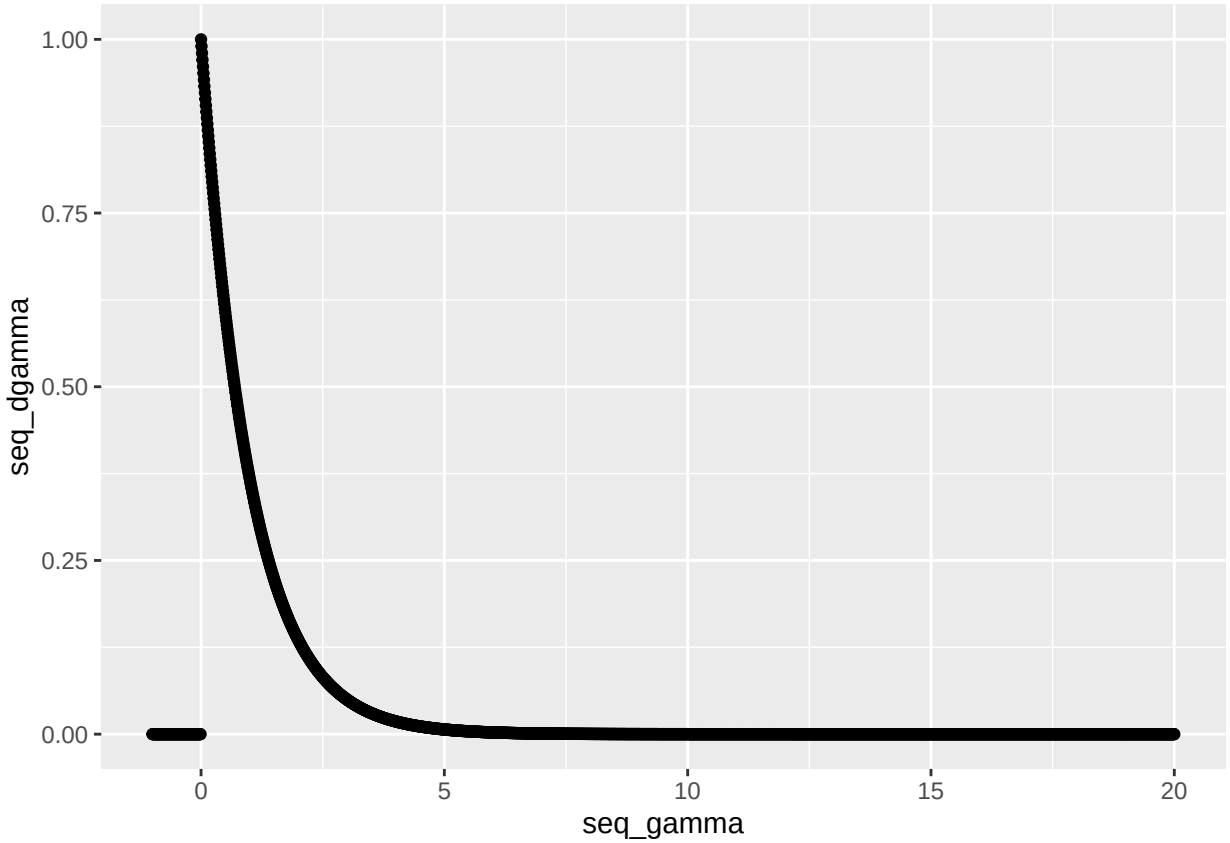



```
# Gamma(1,1)

seq_gamma = seq(-1, 20, .01)

seq_dgamma = dgamma(seq_gamma, 1, 1)

qplot(x = seq_gamma, y = seq_dgamma)
```

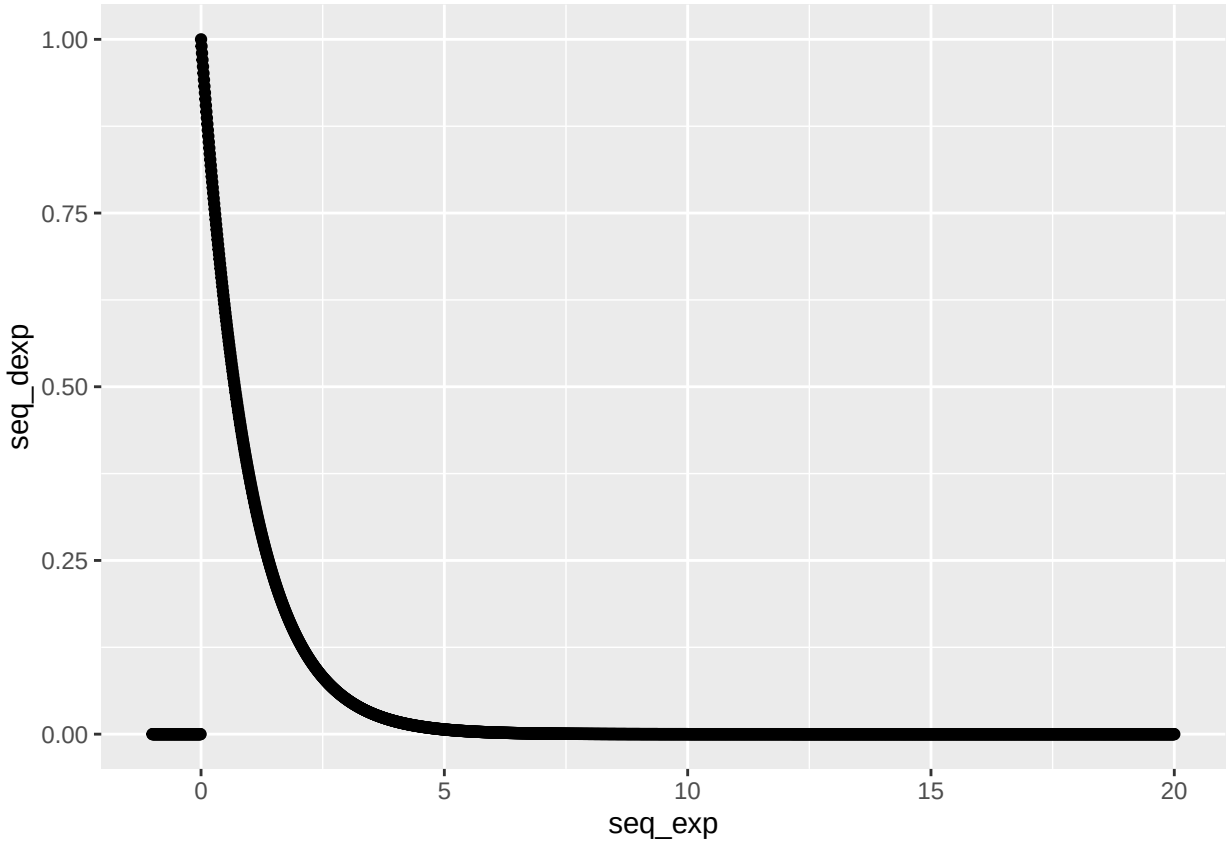



```
# Exponential(1)

seq_exp = seq(-1, 20, .01)

seq_dexp = dexp(seq_exp, 1)

qplot(x = seq_exp, y = seq_dexp)
```



1.7

Note that the mean of a pois(3) is 3, and the variance is also 3.

```
poisson_rv = rpois(100,3)

mean(poisson_rv)
```

```
## [1] 2.93
```

```
var(poisson_rv)
```

```
## [1] 3.399091
```

1.8

```
cel = BSgenome.Celegans.UCSC.ce2

dna_seq = cel$chrM
```