

# Introduction to Computational BioStatistics with R: Distributions

Erik Spence

SciNet HPC Consortium

9 October 2025

# Today's slides

To find today's slides, go to the "Introduction to Computational BioStatistics with R" page, under Lectures, "Distributions".

`https://scinet.courses/1391`

# Today's class

Today we are going to begin our adventure in the world of statistics.

- Distributions.
- R functions for calculating distributions.
- Confidence intervals.
- The \*apply family of functions.

As with all classes, please stop me if you have a question.

# Statistics

To begin at the beginning: what is statistics?

- Statistics is a collection of techniques for empirically describing populations, collections of populations, and the relationships between them.
- Usually we do not have the complete population at our disposal, we only have a sample of the population.
- We use this sample to draw conclusions about the true population from which the sample was drawn, assuming that the sample is representative of the whole population.
- We also use the sample to perform tests to determine the relationship between different populations.

Right. No problem.

# Data come from distributions

We usually use a probability distribution to model our data. What is a probability distribution?

- A probability distribution indicates the probability of a given event (or measurement, or observation) happening.
- There are two types of probability distributions:
  - ▶ discrete: data come in individual steps, there are no data points between those steps (flips of a coin, rolls of dice).
  - ▶ continuous: data are real numbers, with decimal places.
- All probability distributions have the following properties:
  - ▶ The sum of all probabilities equals one.
    - ★ Discrete:  $\sum_i P(x_i) = 1$
    - ★ Continuous:  $\int \rho(x) dx = 1$
  - ▶ One minus the probability of something is the probability of not something.  
 $P'(x) = 1 - P(x)$

# Discrete distributions

Discrete distributions apply when the data are not continuous, the data come in discrete steps.

Binomial distribution:

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

where  $p$  is the probability of success,  $k$  is the number of successes and  $n$  is the number of attempts.

An example of this would be picking  $n$  marbles out of a bag of red and black marbles, and picking  $k$  red marbles.

Coin toss:

$$P(x) = \begin{cases} 0.5, & x = \text{heads}, \\ 0.5, & x = \text{tails} \end{cases}$$

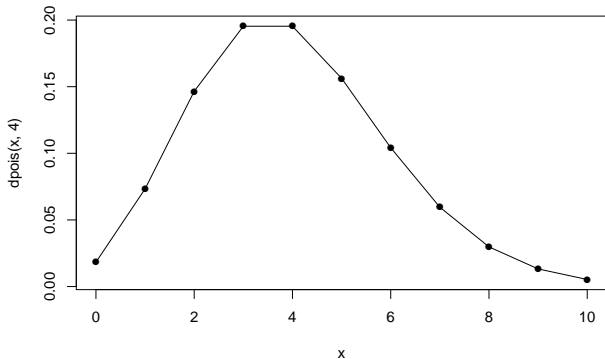
Roll of a die:

$$P(x) = \begin{cases} 1/6, & x = 1, \\ 1/6, & x = 2, \\ 1/6, & x = 3, \\ 1/6, & x = 4, \\ 1/6, & x = 5, \\ 1/6, & x = 6 \end{cases}$$

# Poisson distribution

The Poisson distribution is used for discrete events that happen infrequently. It measures the probability of a certain number of events happening in a specified time period.

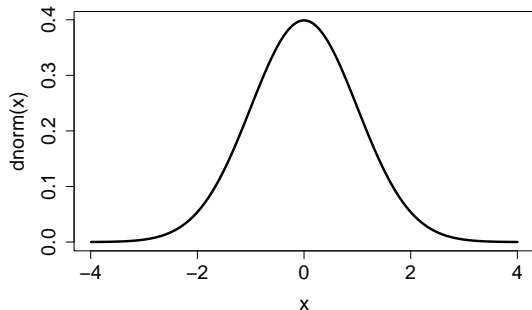
Please do not use continuous distributions for discrete variables!



Poisson distribution ( $\lambda = 4$ ):

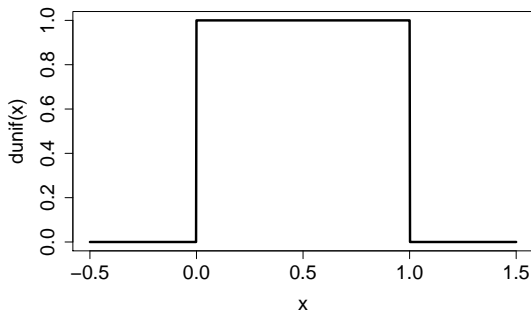
$$P(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

# Continuous distributions



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

Gaussian

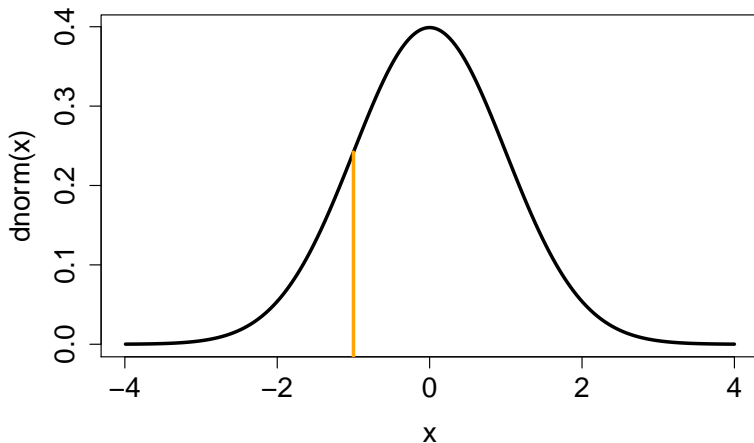


$$\rho(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

Uniform

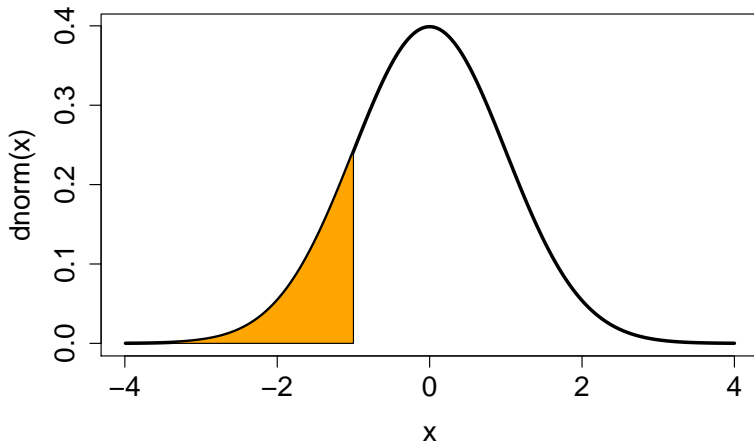


# Some distribution terminology



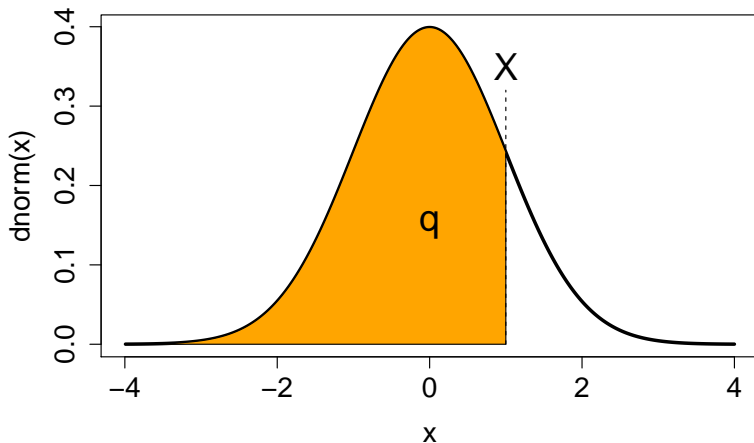
Probability density function (PDF),  $\rho(x)$ : the function which determines probability of getting a particular value,  $X$ , from a continuous distribution,  $P(x = X) = \rho(x)dx$ .

# Some distribution terminology, continued



Cumulative distribution function (CDF): the probability of getting a particular value of  $x$  below a certain value,  $P(x < X) = \int_{-\infty}^X \rho(x) dx$ .

# Some distribution terminology, continued more



Quantile function (QFn): given a probability  $q$ , the particular value of  $X$  such that  $P(x < X) \leq q$ .

# Some more distribution terminology

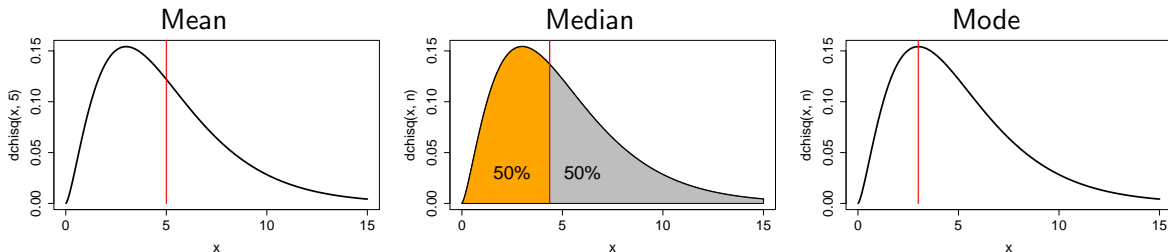
	Symbol	Continuous	Discrete
expectation value	$E[f(x)]$	$\int_{-\infty}^{\infty} f(x)\rho(x)dx$	$\sum f(x_i)P(x_i)$
mean	$\mu$	$\int_{-\infty}^{\infty} x\rho(x)dx$	$\sum x_iP(x_i)$
variance	$\sigma^2$	$E[(x - \mu)^2]$	$E[(x - \mu)^2]$

Note that for discrete distributions, where all cases are equally probable,  $P(x_i) = \frac{1}{n}$ . R comes with many built-in functions to calculate the usual quantities.

- `mean()`, `sd()`, `var()`
- `min()`, `max()`
- `summary()`

As usual, type "`help(mean)`" to learn how these functions can be used. Do NOT use these function names as variables!

# Characteristic statistics



A warning: many distributions are not centred. If the distribution of your data is not centred the concept of a 'mean' may not be meaningful.

The chi-squared distribution, shown here, is not centered or symmetric around its peak.

# Built-in datasets, an aside

R contains built-in datasets that can be used for practicing.

```
> data()
Data sets in package 'datasets':

AirPassengers      Monthly Airline Passenger Numbers 1949-1960
BJsales            Sales Data with Leading Indicator
BJsales.lead (BJsales) Sales Data with Leading Indicator
BOD                Biochemical Oxygen Demand
:
>
> str(faithful)
data.frame':   272 obs.   of 2 variables:
 $ eruptions: num 3.6 1.8 3.33 2.28 4.53 ...
 $ waiting  : num 79 54 74 62 85 55 88 85 51 85 ...
>
```

Type 'q' to get out of the 'data' menu. DO NOT use 'data' as a variable.

# R distribution functions

R has a tonne of distributions built into it. The syntax for using the distributions is fairly consistent. To access a particular distribution, you use the following suffixes:

- Uniform: `unif`
- Normal: `norm`
- Binomial: `binom`
- Poisson: `pois`
- and many many others

To access particular functions associated with those distributions, you use the prefixes:

- Probability Distribution Function (PDF): `d`
- Cumulative Distribution Function (CDF): `p`
- Quantile Function (QFn): `q`
- Random sampling from the distribution: `r`

# R distribution functions, continued

Distribution	suffix	PDF (d)	CDF (p)	QFn (q)	Sample (r)
Normal	norm	dnorm	pnorm	qnorm	rnorm
Uniform	unif	dunif	punif	qunif	runif
Exponential	exp	dexp	pexp	qexp	rexp
Poisson	pois	dpois	ppois	qpois	rpois
Binomial	binom	dbinom	pbinom	qbinom	rbinom

Note that most distributions take optional arguments which control their behaviour (use the 'help' function to get details on how to use these functions).



# R distribution functions, examples

```
>
> # Normal distribution probabilities with
> # default values (mean = 0, sd = 1)
> dnorm(c(-2, 0, 2))
[1] 0.05399097 0.39894228 0.05399097
>
> # Normal distribution probabilities with
> # mean = 1, sd = 2
> dnorm(c(-2, 0, 2), mean = 1, sd = 2)
[1] 0.0647588 0.1760327 0.1760327
>
> # Value of normal distribution which
> # has a probability of 0.025
> qnorm(0.025)
[1] -1.959964
>
```

```
>
> # 4 random samples from the normal distribution
> # with default values (mean = 0, sd = 1)
> rnorm(4)
[1] -0.01890732 -1.51366406 1.00561637 -0.27690594
>
> # 4 random samples from the normal distribution
> # with mean = 1, sd = 2
> rnorm(4, mean = 1, sd = 2)
[1] 1.70841356 2.96691253 0.07857346 -0.87288538
>
> # 4 random samples from the Poisson distribution
> # with lambda = 20
> rpois(4, lambda = 20)
[1] 26 19 25 15
>
```

# Applied examples

Suppose that a given collection of insects have weights that are normally distributed with a

- mean of 17.46 grams and
- variance of 75.67 grams<sup>2</sup>.

What is the probability that a randomly chosen insect within the collection weighs more than 19 grams?

We follow these steps:

- We use the cumulative distribution function (CDF) to get the probability of being less than 19 grams.
- We then subtract this from 1 to get the probability of being greater than 19 grams.

```
>
> p <- pnorm(19, mean = 17.46, sd = sqrt(75.67))
>
> 1 - p
[1] 0.4297405
>
```

Answer: 43%

# Applied examples, continued

Suppose some widgets produced at the Acme Factory have a probability of 0.005 of being defective. The widgets are shipped in boxes of 25.

- What is the probability that a box contains exactly 1 defective widget?
- What is the probability that a box has *at most* 1 defective widget?

Use a binomial distribution, since each widget is either defective or not. The binomial distribution takes two optional arguments:

- size (the number of samples),
- prob (the probability of something occurring).

```
>  
_____  
> dbinom(1, size = 25, prob = 0.005)  
[1] 0.1108317  
_____  
>  
_____  
> pbinom(1, size = 25, prob = 0.005)  
[1] 0.9930519  
_____  
>
```

Answer 1: 11%

Answer 2: 99%

# Sample means, continuous variables

It's very important to distinguish between sample statistics and population statistics. The population is what we want to describe, but the sample from the population is what we have.

We often want to get statistics about the mean of our sampled data,  $\bar{x}$ . For continuous variables:

- What is mean of the sampled means? Meaning, if we sampled from the population many times, what would the mean of all the sampled subpopulation means be? Well, it turns out to be just  $\mu$ , the population mean.
- What is the variance of the sampled means? It's  $\sigma^2/n$ , where  $n$  is the number of samples, and  $\sigma^2$  is the population variance.

Which means, of course, that the standard deviation of the sampled mean is  $\sigma/\sqrt{n}$ . This means that the more samples we have the better our estimate of the population mean will be.

# Sample means, continuous variables, continued

The previous slide assumed that we know  $\sigma^2$ , the population variance. But we don't. How do we get that?

- We should use the sample variance,  
$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}.$$
- Why  $n - 1$ , instead of  $n$ ?
- That's what gives us  $\sigma^2$ , meaning  $s^2$  is an "Unbiased Estimator".
- Note that the R "var" command uses  $n - 1$  in its denominator.

Using  $n - 1$  instead of  $n$  to calculate the variance of the sample data is a better estimate of the population variance.

```
> my.variance <- function(n, pop.var, divide.by) {  
+   my.data <- rnorm(n, sd = sqrt(pop.var))  
+   diffs <- (my.data - mean(my.data))**2  
+   return(sum(diffs) / divide.by)  
+ }  
  
> m <- 10000  
  
> result <- rep(0, m)  
>  
  
> for (i in 1:m) result[i] <- my.variance(20,  
+   pop.var = 8, divide.by = 19)  
  
> mean(result)  
[1] 7.982732  
>  
  
> for (i in 1:m) result[i] <- my.variance(20,  
+   pop.var = 8, divide.by = 20)  
  
> mean(result)  
[1] 7.59985
```

# Calculating confidence intervals

Suppose that you've calculated the mean,  $\bar{x}$ , of some samples. What is our confidence interval on that  $\mu$ ?

To answer this question, we estimate the Standard Error

$$SE(\bar{x})^2 = \frac{s^2}{n}$$

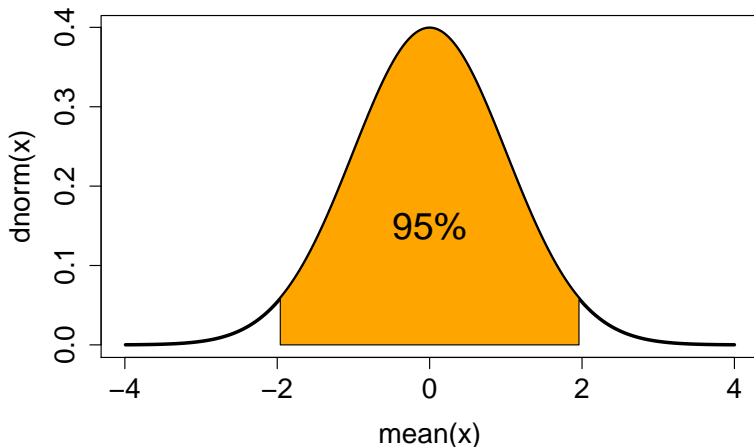
The 95% confidence interval, in which there is a 95% chance the true mean of the population lies, is given by

$$\mu \pm 1.96 SE(\bar{x})$$

This is because 1.96 standard deviations is approximately what contains 95% of the normal distribution.

```
>
> x <- trees$Girth
>
> my.mean <- mean(x)
>
> se2 <- var(x) / length(x)
>
> my.mean - 1.96 * sqrt(se2)
[1] 12.14368
>
> my.mean + 1.96 * sqrt(se2)
[1] 14.35309
>
```

# Calculating confidence intervals



We select a confidence interval that includes 95% of the area under the Gaussian.

# The \*apply family of functions

The \*apply family of functions make it very easy and fast to repeatedly apply a function to a set of individual elements.

Many parallel routines are parallel versions of these higher-level functions.

- lapply: apply a function to each element of a list or vector.
- sapply: like lapply, but return a vector instead of a list.
- apply: apply a function to rows, columns or elements of an array.
- tapply: apply a function to subsets of a list or vector.
- mapply: apply a function to the "transpose" of a list.

Using these functions, rather than regular for loops, can significantly speed up calculations.



# lapply

The function `lapply` will repeatedly apply a function to each element of a list or vector.

Note that you only specify the name of the function to be called as the second argument. You don't give the function any arguments, unless extra arguments are needed by the function, in which case they are supplied to `lapply`, not the function.

```
> mean.n.rnorm <- function(n) return(mean(rnorm(n)))  
>  
> ns <- c(1, 10, 100, 1000)  
>  
> lapply(ns, mean.n.rnorm)  
[[1]]  
[1] -0.01890732  
  
[[2]]  
[1] 0.1327366  
  
[[3]]  
[1] -0.1007226  
  
[[4]]  
[1] -0.03226481  
>
```

# sapply

I usually use sapply, as it returns a vector rather than a list, which I usually find more-convenient to use.

Note that if the function you are calling using one of the \*apply functions takes more than one argument, the extra arguments can be listed in the \*apply function after the name of the function.

```
>  
> ns <- c(1, 10, 100, 1000)  
>  
> random.numbers <- lapply(ns, rnorm)  
>  
> sapply(random.numbers, mean)  
[1] 0.34134897 0.30397851 0.03475841 -0.01775784  
>  
> sapply(random.numbers, sd, na.rm = TRUE)  
[1] NA 0.9419855 0.8996367 1.0172488  
>
```

# apply

The apply function is used on matrices and arrays. It applies a function to the rows ( $MARGIN = 1$ ), or columns ( $MARGIN = 2$ ) of an array.

```
>
> A <- matrix(1:9, nrow = 3)
>
> A
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
> apply(A, MARGIN = 1, max) # max of each row
[1] 7 8 9
>
> apply(A, MARGIN = 2, max) # max of each column
[1] 3 6 9
>
```

# apply, continued

The apply function can also be applied to each element, using `MARGIN = 1:2`.

If you have more than 2 dimensions then those dimensions can also be specified as an argument to `MARGIN`.

```
>
> A <- matrix(1:9, nrow = 3)
>
> A
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
>
> apply(A, MARGIN = 1:2, function(x) return(x**2))
      [,1] [,2] [,3]
[1,]    1   16   49
[2,]    4   25   64
[3,]    9   36   81
>
```

# Enough to get started

- Today's class went over the very beginning concepts needed to do statistics.
- There are two classes of distributions, continuous and discrete.
- All the distribution commands are built into R already. If they aren't in the base R installation, they exist in a separate package.
- Using sampled data we can get estimates and confidence intervals for the mean and variance of the population.
- The `*apply` family functions can be used to operate on vectors, matrices or data frames.