Probability and Sampling Distributions

Yebelay Berehan

Biostatistician

yebelay.ma@gmail.com

2022-06-01

Outlines

- **Random sampling**
- Built in **discrete probability distributions**
  - *Binomial* Distribution
  - *Geometric* Distribution
  - *Negative Binomial* Distribution
  - *Poisson* Distribution
- Built in **continuous probability distributions**
  - *Uniform* distribution
  - *Exponential* distribution
  - *Normal* distribution
- **Examining the distribution of a set of data**
- **Simulating** the Sample Distribution of the Mean

Random sampling

- Because R is a language built for statistics, it contains many functions that allow you generate random data
  – either from a vector of data, or from an established probability distribution.

- The standard `sample()` function used for drawing random values from a vector.

Argument definition `sample()` function

- `x`: A vector of outcomes you want to sample from.
- `size`: The number of samples you want to draw. *The default is the length of x*.
- `replace`: Should sampling be done with replacement?
- `prob`: A vector of probabilities of the same length as x indicating how likely each outcome in x is.
  - *The default is equally likely*.
- The `sample()` function allows you to draw random samples of elements (scalars) from a vector.

Let's use sample() to draw 10 samples from a vector of integers from 1 to 10.

```
# Draw with out replacement
sample(x = 1:10, size  = 5)
```

```
## [1]  3  9  5 10  6
```

```
# Draw with replacement
sample(x= 1:5, size = 10,replace=TRUE)
```

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

- If you try to draw a large sample from a vector replacement, R will return an error because it runs out of things to draw:
- To specify how likely each element in the vector x should be selected, use the prob argument.
- The length of the prob argument should be as long as the x argument.

```
#Draw 10 samples with probability of selecting “a” to be .90
sample(x = c("a", "b"), prob = c(.8, .2),
       size = 10, replace = TRUE)
```

```
##  [1] "a" "a" "a" "a" "a" "a" "a" "a" "a" "b"
```

Built in discrete probability distributions

- What is a probability distribution?
- How to generate random data from specified probability distributions.

Discrete Distribution

Distribution	R name	Parameters
Binomial	binom	size, prob
Negative binomial	nbinom	size, prob
Poisson	pois	lambda
geometric	geom	prob

Continuous Distribution

Distribution	R name	Parameters
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df2, ncp
uniform	unif	min, max
normal	norm	mean, sd
Student's t	t	df, ncp

General Syntax for Distribution Functions

- There are four basic R commands that apply to the various distributions defined in R.
- Functions are provided to evaluate the pdf (d), CDF (p), quintile (q) and simulate from the distribution (r).
- Each letter can be added as a prefix to any of the R distribution names.
- Letting dist denote the particular distribution then the basic syntax of the four basic commands are:

```
ddist (x, parameters)  # probability density of DIST evaluated at x.
qdist (p, parameters)  # returns x for Pr(DIST(parameters)>= x) = p
pdist(x, parameters)  # returns Pr(DIST(parameters) <= x)
rdist(n, parameters)  # generates n random variables from DIST (parameters)
```

R Functions for Probability Distributions

- Every distribution that R handles has four functions.
- There is a root name, for example, the root name for the normal distribution is norm.
- This root is prefixed by one of the letters
- `p`: for **probability**, the cumulative distribution function (c.d.f.)
- `q`: for **quantile**, the inverse c.d.f.
- `d`: for **density**, the density function (p.f. or p.d.f.)
- `r`: for **random**, a random variable having the specified distribution
- For the binomial distribution, these functions are `pbinom`, `qbinom`, `dbinom`, and `rbinom`.
- For a discrete distribution, the **d** function calculates the density (p.f.), which in this case is a probability

$$f(x) = P(X = x)$$

The Binomial Distribtion

- Density, distribution function, quantile function and random generation for the binomial distribution with parameters size and prob.

```
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)
```

- **Arguments**

  - `x, q` : vector of quantiles.
  - `p` : vector of probabilities.
  - `n` : number of observations.
  - `size` : number of trials (zero or more).
  - `prob` : probability of success on each trial.
  - `log, log.p` : logical; if TRUE, probabilities p are given as log(p).
  - `lower.tail` : logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Details

- The binomial distribution with `size = n` and `prob = p` has density

$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$, for $x = 0, \ldots, n$.

- `dbinom` is the R function that calculates the p.f. of the binomial distribution.

- Both of the R commands in the box below do exactly the same thing.

```
dbinom(27, size=100, prob=0.25)
```

```
## [1] 0.08064075
```

```
dbinom(27, 100, 0.25)
```

```
## [1] 0.08064075
```

- They look up P(X = 27) when X is has the Bin(100, 0.25) distribution.

Question: What is P(X = 1) when X has the Bin(25, 0.005) distribution?

- `pbinom` is the R function that calculates the c.d.f. of the binomial distribution.

```
pbinom(27, size=100, prob=0.25)
```

```
## [1] 0.7223805
```

```
pbinom(27, 100, 0.25)
```

```
## [1] 0.7223805
```

- They look up P(X `<=` 27) when X is has the Bin(100, 0.25) distribution.

Question: What is P(X <= 1) when X has the Bin(25, 0.005) distribution?

- `qbinom` is the R function that calculates the **inverse c.d.f.** of the binomial distribution.
- The quantile is defined as the smallest value x such that `F(x) >= p`, where F is the distribution function.

Example Question: What are the 10th, 20th, and so forth quantiles of the `Bin(10, 1/3)` distribution?

```r
qbinom(0.1, 10, 1/3)
```

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

```r
qbinom(0.2, 10, 1/3)
```

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

```r
# and so forth, or all at once with
qbinom(seq(0.1, 0.9, 0.1), 10, 1/3)
```

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

The Geometric Distribution

- Density, distribution function, quantile function and random generation for the geometric distribution with parameter prob.

```
dgeom(x, prob, log = FALSE)
pgeom(q, prob, lower.tail = TRUE, log.p = FALSE)
qgeom(p, prob, lower.tail = TRUE, log.p = FALSE)
rgeom(n, prob)
```

Arguments

- `x, q` : vector of quantiles representing the number of failures in a sequence of Bernoulli trials before success occurs.
- `p` : vector of probabilities.
- `n` : number of observations.
- `prob` : probability of success in each trial. 0 < prob <= 1.
- `log, log.p`:*logical*; if **TRUE**, probabilities p are given as log(p).
- `lower.tail` : *logical*; if **TRUE** (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

- The geometric distribution with `prob = p` has density

$p(x) = p(1-p)^x$, for $x = 0, 1, 2, \ldots, 0 < p \leq 1.$

- If an element of x is not integer, the result of dgeom is zero, with a warning.

- The quantile is defined as the smallest value xx such that $F(x) \geq p$, where FF is the distribution function.

Negative Binomial Distribution

- Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters size and prob.

```
dnbinom(x, size, prob, mu, log = FALSE)
pnbinom(q, size, prob, mu, lower.tail = TRUE, log.p = FALSE)
qnbinom(p, size, prob, mu, lower.tail = TRUE, log.p = FALSE)
rnbinom(n, size, prob, mu)
```

- `x` : vector of (non-negative integer) quantiles.
- `q` : vector of quantiles.
- `p` : vector of probabilities.
- `n` : number of observations.
- `size` : target for number of successful trials.
- `prob`: probability of success in each trial. 0 < prob <= 1.
- `mu`: alternative parametrization via mean: see 'Details'.
- `log, log.p`: *logical*; if **TRUE**, probabilities p are given as log(p).
- `lower.tail`: logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

- The negative binomial distribution with `size = n` and `prob = p` has density

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

for $x = 0, 1, 2, \ldots, n > 0$ and $0 < p \leq 1$.

- This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached.
- The mean is $\mu = n(1-p)/p$ and variance $n(1-p)/p^2$.

Poisson Distribution

- Density, distribution function, quantile function and random generation for the Poisson distribution with parameter lambda.

```
dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)
```

Arguments

- x : vector of (non-negative integer) quantiles.
- q : vector of quantiles.
- p : vector of probabilities.
- n : number of random values to return.
- lambda : vector of (non-negative) means.
- log, log.p: *logical*; if **TRUE**, probabilities p are given as log(p).
- lower.tail: *logical*; if **TRUE** (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

The Poisson distribution has density

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

for $x = 0, 1, 2, \ldots, .$

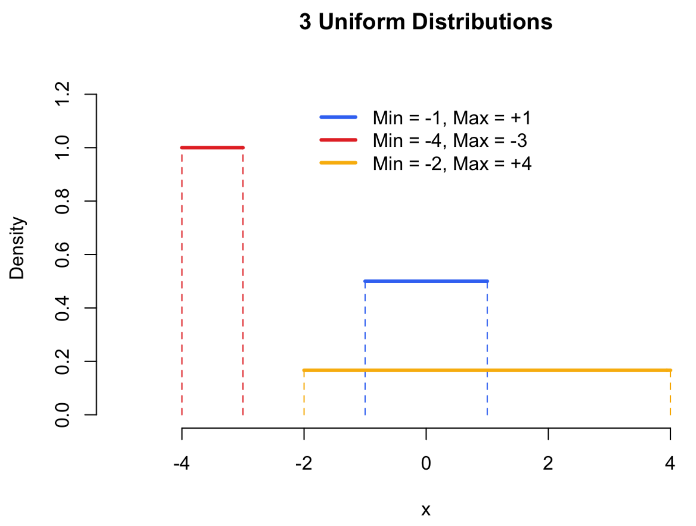
- The mean and variance are $E(X) = Var(X) = \lambda$.

- `dpois` gives the (log) density,
- `ppois` gives the (log) distribution function,
- `qpois` gives the quantile function, and
- `rpois` generates random deviates.

Built in continuous probability distributions

Uniform distribution

Next, let’s move on to the Uniform distribution.



- The Uniform distribution gives equal probability to all values between its minimum and maximum values.

- To generate samples from a uniform distribution, use the function `runif()`, the function has 3 arguments:

Argument Definition from `runif()`

- `n`: The number of observations to draw from the distribution.
- `min`: The lower bound of the Uniform distribution from which samples are drawn
- `max`: The upper bound of the Uniform distribution from which samples are drawn

```r
# 5 samples from Uniform dist with bounds at 0 and 1
runif(n = 5, min = 0, max = 1)
```

```
## [1] 0.40765415 0.75443572 0.33264727 0.06374601 0.81593532
```

```r
# 10 samples from Uniform dist with bounds at -100 and +100
runif(n = 10, min = -100, max = 100)
```

```
##  [1]   3.738195 -37.283596 -51.222569  -9.213872  66.240368 -46.478120
##  [7]   1.915849   4.445141 -36.848803 -92.220085
```

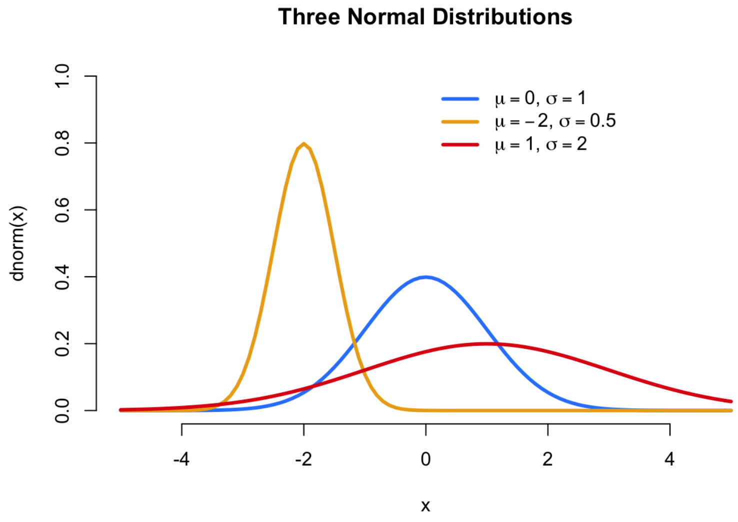
Exponential distribution

```
dexp(x, rate = 1, log = FALSE)
pexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qexp(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
rexp(n, rate = 1)
```

Normal distribution

1. Normal distribution

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p =FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```



- Three different normal distributions with different means and standard deviations

Argument Definition

- `n`: The number of observations to draw from the distribution.
- `mean`: The mean of the distribution.
- `sd`: The standard deviation of the distribution.

- The Normal distribution is bell-shaped, and has two parameters: a mean and a standard deviation.

- To generate samples from a normal distribution in R, we use the function `rnorm()`

```r
# 5 samples from a Normal dist with mean = 0, sd = 1
rnorm(n = 5, mean = 0, sd = 1)
```

```
## [1]  2.0348124  0.8668959  1.0272311 -0.1837647  0.6780077
```

```r
# 3 samples from a Normal dist with mean = -10, sd = 15
rnorm(n = 3, mean = -10, sd = 15)
```

```
## [1]  3.699722  3.806144 -3.003894
```

Random samples will always change

- Every time you draw a sample from a probability distribution, you'll `(likely)` get a different result.
- For example, see what happens when I run the following two commands

Use set.seed() to control random samples

- There will be cases where you will want to create a reproducible example of some code that anyone else can replicate exactly.
- To do this, use the `set.seed()` function.
- Using `set.seed()` will force R to produce consistent random samples at any time on any computer.

– you can set the seed to any integer you want.

```
#   always produce the same values
set.seed(100)
rnorm(3, mean = 0, sd = 1)
```

- The following examples illustrate the use of the R functions for computations involving statistical distributions:

```
rnorm(10) # draws 10 random numbers
rnorm(10, 5, 2) # N(μ= 5,sigma = 2) distribution
dnorm(2) # return pdf at z= 2.
pnorm(0) # returns cdf at t =0
qnorm(0.5) # returns the 50% quantile
```

```
mysample <- rnorm(50) # generates random numbers
mu <- mean(mysample) # computes the sample mean
sigma <- sd(mysample) # computes the sample standard
x <- seq(-4, 4, length = 500) # defines x values for the pdf
options(digits=3)

y <- round(dnorm(x, mu, sigma), digits=4) # computes the normal pdf
y
```

Repeatable Simulations

- For a simulation to be repeatable we need to specify the type of random number generator and the initial state of the generator.
- The simplest way to specify the initial state or seed is to use, `set.seed(seed)`
- The argument seed is a single integer value
- Different seeds give different pseudo-random values
- Calling `set.seed()` with the same seed produces the same results, if the sequence of calls is repeated exactly.
- If a seed is not specified then the random number generator is initialized using the time of day.

example

```
set.seed(17632)
runif(5)
rnorm(5)
set.seed(89432)
runif(5)
set.seed(17632)
runif(5)
rnorm(5)
set.seed(17632)
rnorm(5)
runif(5)
```

Simulating the Sample Distribution of the Mean

- Simulation is a numerical technique for conducting experiments on the computer.
- It uses to compare results of an inference under different assumptions
- In any of the cases, it is often needed to create repeated random samples from a specific statistical model, and see how our approach behaves.
- The central limit theorem is perhaps the most important concept in statistics.
- Samples taken from any distribution with finite mean and standard deviation, will tend towards a normal distribution around the mean of the population as sample size increases.
- Furthermore, as sample size increases, the variation of the sample means will decrease.

```
data<-rnorm(25 , 100 , 15)
mean(data)
sd(data)
```

- We know that, when the population is normal, $\mu = 100, \sigma = 15, and N = 25$, the sample mean has a normal distribution with mean 100 and standard deviation 3.

- Let's verify that with a statistical simulation.

```
mean(rnorm(25 , 100 , 15))
replicate(10,mean(rnorm(25, 100, 15)))  # replicate 10 times
data<-replicate(100000,mean(rnorm(25,100, 15))) #replicate 100000 times
mean(data )
sd(data )
```

- Those results are very close to our theoretical expectation.

- Let's look at histogram of our means.

```
hist(data, breaks=100) #Or
plot (density(data))   #Density plot of data
```

- It certainly looks normal
- We can easily induce R to superimpose the precise probability density function on top of this graph. I'm making my line dotted red.