Continous random variables & simulation

The normal distribution

dnorm(x, mean, sd)

The dnorm function returns the value of the probability density function (pdf) of a normal distribution, given a random variable **x** a population mean μ and a population standard deviation σ

```
#find the value of the standard normal distribution pdf at x=0
dnorm(x = 0, mean = 0, sd = 1)
```

```
## [1] 0.3989423
```

```
#by default, R uses mean=0 and sd=1
dnorm(x = 0)
```

```
## [1] 0.3989423
```

```
#find the value of the normal distribution pdf at x=10 with mean=20 and sd=5
dnorm(x = 10, mean = 20, sd = 5)
```

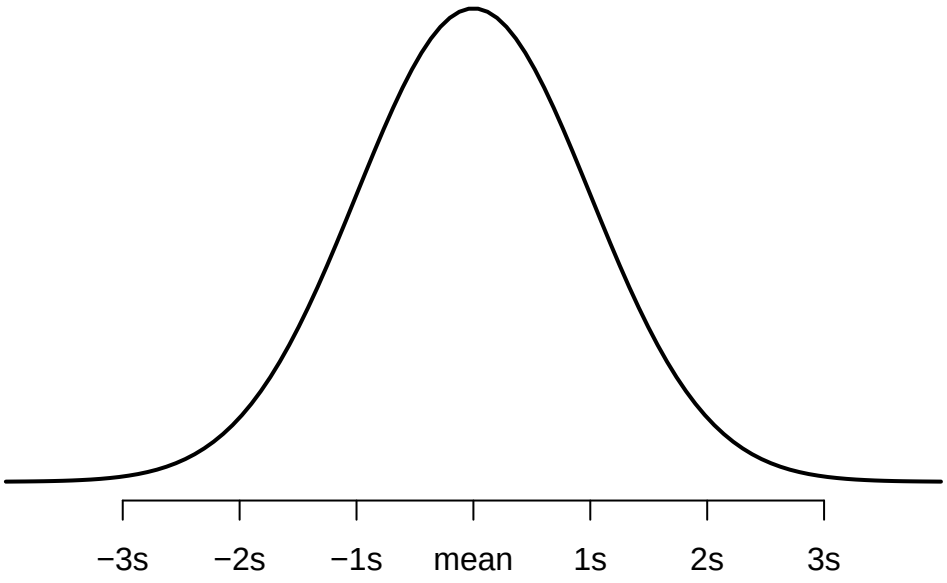
```
## [1] 0.01079819
```

`dnorm` is not that useful for solving questions about probability using the normal distribution. It is however useful to create a normal distribution plot:

```
#Create a sequence of 100 equally spaced numbers between -4 and 4
x <- seq(-4, 4, length = 100)

#create a vector of values that shows the height of the probability distribution
#for each value in x
y <- dnorm(x)

#plot x and y as a scatterplot with connected lines (type = "l") and add
#an x-axis with custom labels
plot(x, y, type = "l", lwd = 2, axes = FALSE, xlab = "", ylab = "")
axis(1, at = -3:3, labels = c("-3s", "-2s", "-1s", "mean", "1s", "2s", "3s"))
```



`pnorm(q, mean, sd)`

The function pnorm returns the value of the cumulative density function (cdf) of the normal distribution given a certain random variable `q`, a population mean μ and population standard deviation σ.

Put simply, pnorm returns the area to the left of a given value x in the normal distribution. If you're interested in the area to the right of a given value `q`, you can simply add the argument `lower.tail = FALSE` i.e., `pnorm(q, size, prob, lower.tail = FALSE)`

Example:
Suppose the height of males at a certain school is normally distributed with a mean of $\mu = 70$ inches and a standard deviation of $\sigma = 2$ inches. Approximately what percentage of males at this school are taller than 74 inches?

```
#find percentage of males that are taller than 74 inches in a population with
#mean = 70 and sd = 2
pnorm(74, mean = 70, sd = 2, lower.tail = FALSE)
```

```
## [1] 0.02275013
```

At this school, **2.275%** of males are taller than **74 inches**.

Example 2:
Suppose the weight of a certain species of otters is normally distributed with a mean of $\mu = 30$ lbs and a standard deviation of $\sigma = 5$ lbs. Approximately what percentage of this species of otters weight less than 22 lbs?

```
#find percentage of otters that weight less than 22 lbs in a population with
#mean = 30 and sd = 5
pnorm(22, mean = 30, sd = 5)
```

```
## [1] 0.05479929
```

Approximately** 5.4799%** of this species of otters weigh less than **22 lbs**.

`qnorm(p, mean, sd)`

The function pnorm returns the value of the inverse cumulative density function (cdf) of the normal distribution given a certain random variable `q`, a population mean μ and population standard deviation σ.

Put simply, you can use qnorm to find out what the Z-score is of the p^{th} quantile of the normal distribution.

```
#find the Z-score of the 99th quantile of the standard normal distribution
qnorm(.99, mean = 0, sd = 1)
```

```
## [1] 2.326348
```

```
#by default, R uses mean=0 and sd=1
qnorm(.99)
```

```
## [1] 2.326348
```

```
#find the Z-score of the 95th quantile of the standard normal distribution
qnorm(.95)
```

```
## [1] 1.644854
```

```
#find the Z-score of the 10th quantile of the standard normal distribution
qnorm(.10)
```

```
## [1] -1.281552
```

rnorm(n, mean, sd)

The function rnorm generates a vector of normally distributed random variables given a vector length `n`, a population mean μ and population standard deviation σ

```
#generate a vector of 5 normally distributed random variables with mean=10
#and sd=2
five <- rnorm(5, mean = 10, sd = 2)
five
```

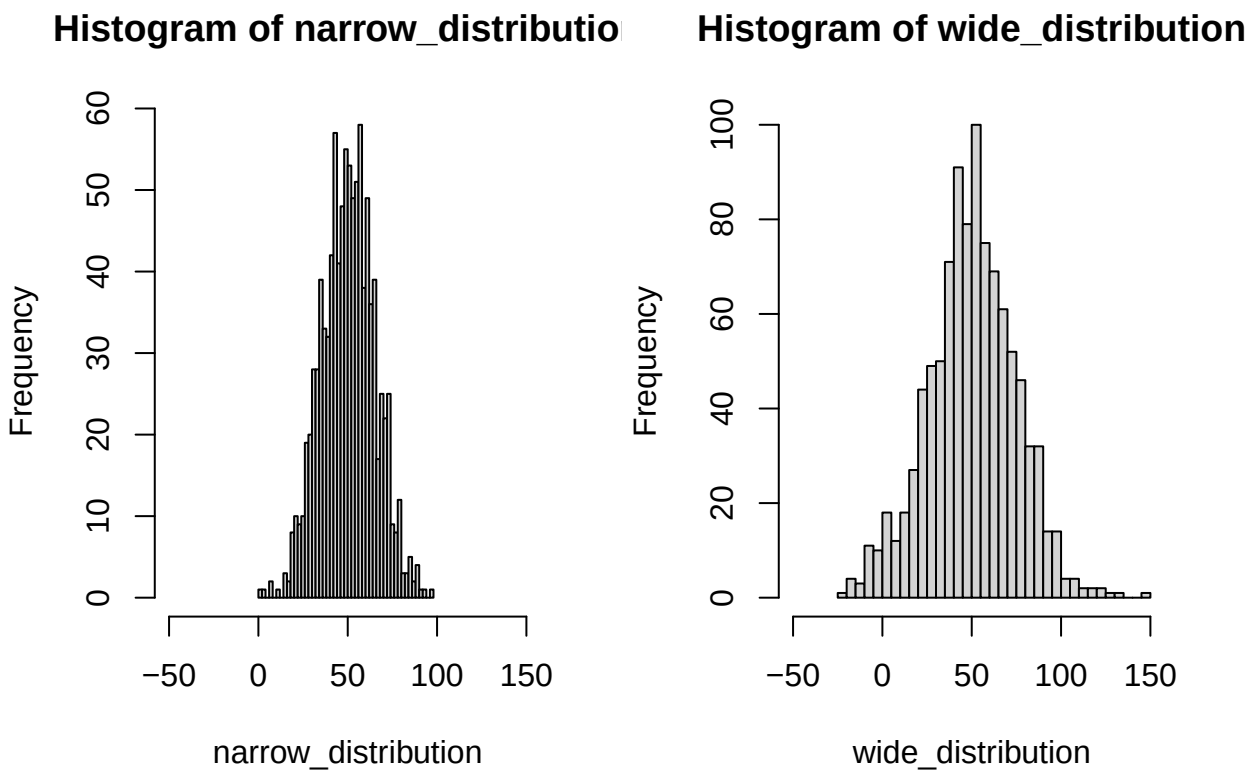
```
## [1]  6.214588 12.770330 10.442763 12.774084  5.684590
```

```
# [1] 10.658117 8.613495 10.561760 11.123492 10.802768

#generate a vector of 1000 normally distributed random variables with mean=50
#and sd=5
narrow_distribution <- rnorm(1000, mean = 50, sd = 15)

#generate a vector of 1000 normally distributed random variables with mean=50
# and sd=25
wide_distribution <- rnorm(1000, mean = 50, sd = 25)

#generate two histograms to view these two distributions side by side, specify
#50 bars in histogram and x-axis limits of -50 to 150
par(mfrow = c(1, 2)) #one row, two columns
hist(narrow_distribution, breaks = 50, xlim = c(-50, 150))
hist(wide_distribution, breaks = 50, xlim = c(-50, 150))
```



Notice how the wide distribution is much more spread out compared to the narrow distribution. This is because we specified the standard deviation in the wide distribution to be 25 compared to just 15 in the narrow distribution. Also notice that both histograms are centered around the mean of 50.

The binomial (Bernoulli) distribution

`dbinom(x, size, prob)`

The function dbinom returns the value of the probability density function (pdf) of the binomial distribution given a certain random variable x, number of trials (size) and probability of success on each trial (prob).

Put simply, dbinom finds the probability of getting a certain number of successes (`x`) in a certain number of trials (`size`) where the probability of success on each trial is fixed (`prob`).

Example:
Bob makes 60% of his free-throw attempts. If he shoots 12 free throws, what is the probability that he makes exactly 10?

```
#find the probability of 10 successes during 12 trials where the probability of
#success on each trial is 0.6
dbinom(x = 10, size = 12, prob = .6)
```

```
## [1] 0.06385228
```

The probability that he makes exactly 10 shots is **0.0639**.

`pbinom(q, size, prob)`

The function pbinom returns the value of the cumulative density function (cdf) of the binomial distribution given a certain random variable q, number of trials (size) and probability of success on each trial (prob).

Put simply, pbinom returns the area to the left of a given value q in the binomial distribution. If you're interested in the area to the right of a given value q, you can simply add the argument `lower.tail = FALSE` i.e, `pbinom(q, size, prob, lower.tail = FALSE)`

Example:
Ando flips a fair coin 5 times. What is the probability that the coin lands on heads more than 2 times?

```
#find the probability of more than 2 successes during 5 trials where the
#probability of success on each trial is 0.5
pbinom(2, size = 5, prob = .5, lower.tail = FALSE)
```

```
## [1] 0.5
```

```
#find the probability of 4 or fewer successes during 10 trials where the
#probability of success on each trial is 0.3
pbinom(4, size = 10, prob = .3)
```

```
## [1] 0.8497317
```

The probability that the coin lands on heads more than **2** times is **0.5**.

Example 2:
Suppose Tyler scores a strike on 30% of his attempts when he bowls. If he bowls 10 times, what is the probability that he scores 4 or fewer strikes?

```
#find the probability of 4 or fewer successes during 10 trials where the
#probability of success on each trial is 0.3
pbinom(4, size = 10, prob = .3)
```

```
## [1] 0.8497317
```

The probability that he scores 4 or fewer strikes is **0.8497**.

qbinom(q, size, prob)

The function qbinom returns the value of the inverse cumulative density function (cdf) of the binomial distribution given a certain random variable q, number of trials (size) and probability of success on each trial (prob).

Put simply, you can use qnorm to find out the p^{th} quantile of the binomial distribution.

```
#find the 10th quantile of a binomial distribution with 10 trials and prob
#of success on each trial = 0.4
qbinom(.10, size = 10, prob = .4)
```

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

```
#find the 40th quantile of a binomial distribution with 30 trials and prob
#of success on each trial = 0.25
qbinom(.40, size = 30, prob = .25)
```

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

rbinom(n, size, prob)

The function rbinom generates a vector of binomial distributed random variables given a vector length n, number of trials (size) and probability of success on each trial (prob).

Example:

```
#generate a vector that shows the number of successes of 10 binomial
#experiments with
#100 trials where the probability of success on each trial is 0.3.
results <- rbinom(10, size = 100, prob = .3)
results
```

```
##  [1] 29 24 25 34 36 32 33 26 36 31
```

```
#find mean number of successes in the 10 experiments (compared to expected
#mean of 30)
mean(results)
```

```
## [1] 30.6
```

```
#generate a vector that shows the number of successes of 1000 binomial
#experiments
#with 100 trials where the probability of success on each trial is 0.3.
results <- rbinom(1000, size = 100, prob = .3)

#find mean number of successes in the 100 experiments (compared to expected
#mean of 30)
mean(results)
```

```
## [1] 30.039
```

Notice how the more random variables we create, the closer the mean number of successes is to the expected number of successes.

Note: "Expected number of successes" $= n \cdot p$ where n is the number of trials and p is the probability of success on each trial.

The geometric distribution

dgeom(x, prob)

The dgeom function finds the probability of experiencing a certain amount of failures before experiencing the first success in a series of Bernoulli trials given (x) the number of failures before the first success and (prob) the probability of success on each trial

A researcher is waiting outside of a library to ask people if they support a certain law. The probability that a given person supports the law is $p = 0.2$. What is the probability that the fourth person the researcher talks to is the first person to support the law?

```
dgeom(x = 3, prob = .2)
```

```
## [1] 0.1024
```

The probability that the researchers experiences 3 "failures" before the first success is 0.1024.

pgeom(q, prob)

The pgeom function finds the probability of experiencing a certain amount of failures or less before experiencing the first success in a series of Bernoulli trials ggiven (q) the number of failures before the first success and (prob) the probability of success on each trial.

A researcher is waiting outside of a library to ask people if they support a certain law. The probability that a given person supports the law is p = 0.2. What is the probability that the researcher will have to talk to more than 5 people to find someone who supports the law?

```
1 - pgeom(q = 5, prob = .2)
```

```
## [1] 0.262144
```

```
# same as
 pgeom(q = 5, prob = .2, lower.tail = FALSE)
```

```
## [1] 0.262144
```

The probability that the researcher will have to talk to more than 5 people to find someone who supports the law is **0.262144**.

qgeom(p, prob)

The qgeom function finds the number of failures that corresponds to a certain percentile given (p) the percentile and (prob) the probability of success on each trial.

A researcher is waiting outside of a library to ask people if they support a certain law. The probability that a given person supports the law is p = 0.2. We will consider a "failure" to mean that a person does not support the law. How many "failures" would the researcher need to experience to be at the 90th percentile for number of failures before the first success?

```
qgeom(p = .90, prob = 0.2)
```

```
## [1] 10
```

The researcher would need to experience **10** "failures" to be at the 90th percentile for number of failures before the first success.

rgeom(n, prob)

The rgeom function generates a list of random values that represent the number of failures before the first success given (n) the number of values to generate and (prob) the probability of success on each trial.

A researcher is waiting outside of a library to ask people if they support a certain law. The probability that a given person supports the law is $p = 0.2$. We will consider a "failure" to mean that a person does not support the law. Simulate 10 scenarios for how many "failures" the researcher will experience until she finds someone who supports the law.

```
set.seed(0) #make this example reproducible

rgeom(10, 0.2)
```

```
##  [1]  1  2  1 10  7  4  1  7  4  1
```

The way to interpret this is as follows:

- During the first simulation, the researcher experienced 1 failure before finding someone who supported the law.
- During the second simulation, the researcher experienced 2 failures before finding someone who supported the law.
- During the third simulation, the researcher experienced 1 failure before finding someone who supported the law.
- During the fourth simulation, the researcher experienced 10 failures before finding someone who supported the law.
- and so on..