MTH 3220 R Notes 9

10 Random Numbers, Simulation, and Probability Distributions

10.1 Sampling Elements from a Vector

- To generate a random sample from the elements of a vector, we use:

```
sample()      # Generate a random sample from the elements of a vector.
```

- `sample()` takes arguments `x`, a vector, and `size`, the sample size. It returns a sample drawn ***without replacement*** from the elements of `x`. An optional argument `replace` can be set to `TRUE` to sample ***with replacement***.
- For example, to generate a random sample of size $n = 10$ from the numbers $1, 2, \ldots, 100$, type:

```
sample(x = 1:100, size = 10)

##  [1] 19 25 58 34 23 63 84 56 81 87
```

- Setting `size` equal to `length(x)` returns a random ***permutation*** (ordering) of the elements of `x`. For example, to place the numbers $1, 2, 3, 4, 5$ in a random order, type:

```
sample(x = 1:5, size = 5)

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

10.2 Duplicating a Random Sample Using `set.seed()`

- Sometimes we'll need to *reproduce* a set of random numbers later. We can do it if we use:

```
set.seed()      # Set the random seed, which determines which numbers
                # will be generated by R's random number generator.
```

- `set.seed()` takes a positive integer argument `seed` (any value will do) that determines which numbers will be generated by R's random number generator.
- For example, below we set the "seed" to 15 *before* using `sample()`:

```
set.seed(15)
sample(x = 1:100, size = 10)

##  [1] 61 20 95 64 36 94 77 24 97 76
```

Now to regenerate the *same* set of random numbers, we set the "seed" to 15 again before generating the sample:

```
set.seed(15)
sample(x = 1:100, size = 10)

##  [1] 61 20 95 64 36 94 77 24 97 76
```

10.3 Random Variables and Probability Distributions

- R has built-in functions that generate a random values, compute probabilities, determine percentiles, and evaluate density or probability mass functions for any of the common continuous or discrete probability distributions. Here's the syntax for the most important distributions:

```
r, p, q, or d followed by:

  unif()        # Uniform distribution, with parameters specified by
                # the arguments min and max.
  norm()        # Normal distribution, with parameters specified by
                # the arguments mean and sd.
  exp()         # Exponential distribution, with parameter specified
                # by the argument rate.
  gamma()       # Gamma distribution, with parameters specified by
                # the arguments shape and rate.
  cauchy()      # Cauchy distribution, with parameters specified by
                # the arguments location and scale.
  lnorm()       # Lognormal distribution, with parameters specified
                # by the arguments meanlog and sdlog.
  t()           # T distribution, with parameter specified by the
                # argument df.
  f()           # F distribution, with parameters specified by the
                # arguments df1 and df2.
  chisq()       # Chi-square distribution, with parameter specified
                # by the argument df.
  binom()       # Binomial distribution, with parameters specified
                # by the arguments size and prob.
  pois()        # Poisson distribution, with parameter specified by
                # the argument lambda.
  nbinom()      # Negative binomial distribution, with parameters
                # specified by the arguments size and prob.
  geom()        # Geometric distribution, with parameter specified by
                # the argument prob.
  hyper()       # Hypergeometric distribution, with parameters
                # specified by the arguments m, n, and k.
```

- In each case, the function name is prefaced by `r` to generate a random sample (e.g. `runif()`), `p` to find a probability (e.g. `punif()`), `q` to determine a percentile, or quantile (e.g. `qunif()`), and `d` to evaluate the probability density function or probability mass function (e.g. `dunif()`).

  A distribution's parameter values are set via arguments to the function (e.g. the `min` and `max` arguments of the `unif()` functions).

  Some of these functions are illustrated in the subsections that follow.

10.3.1 Uniform Random Variables and Probability Distributions

- `runif()` takes arguments `n`, `min`, and `max`, and returns a sample of size `n` randomly selected over the interval from `min` to `max`.

  For example, to generate a random sample of size $n = 3$ from a uniform(0, 1) distribution, type:

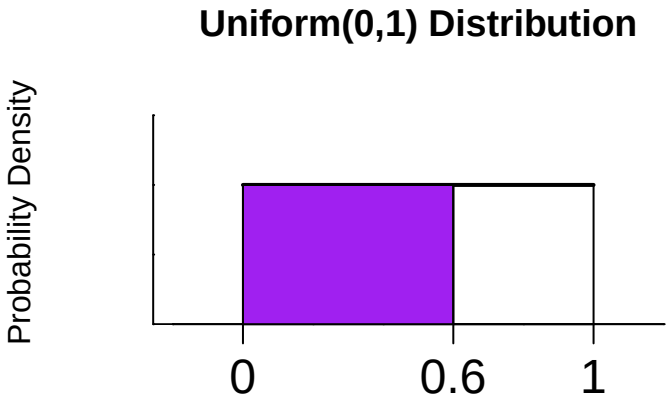
```
runif(n = 3, min = 0, max = 1)

## [1] 0.4161184 0.6947637 0.1488006
```

- `punif()` returns probabilities of the form $P(U \leq u)$, where U is a uniformly distributed random variable and u is specified by the argument `q`.

  For example, the probability $P(U \leq 0.6)$, where U is a uniform(0, 1) random variable, is obtained by typing:

```
punif(q = 0.6, min = 0, max = 1)

## [1] 0.6
```

  Thus the probability is 0.6, or 60%. This is the shaded area below.

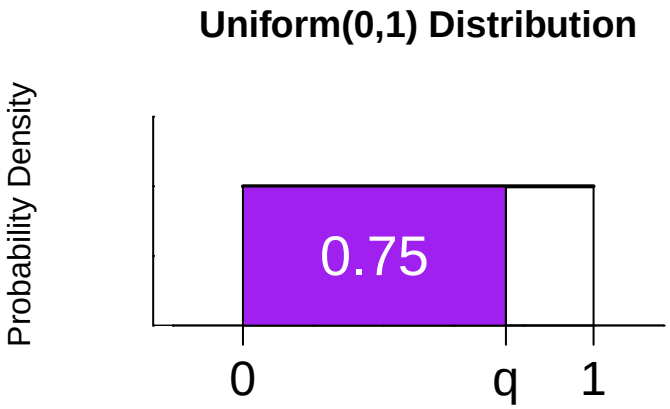


- `qunif()` takes an argument `p`, and returns the **100*pth percentile*** (or ***pth quantile***) of a uniform distribution.

  For example, to find the 75th percentile of the uniform(0, 1) distribution, we type:

```
qunif(p = 0.75, min = 0, max = 1)

## [1] 0.75
```

  Thus the percentile is $q = 0.75$. This is represented by the value q in the figure below.



- `dunif()` takes an argument `x`, and returns the uniform(a, b) ***probability density function***

$$f(x) = \frac{1}{b-a},$$

(where a and b are specified by `min` and `max` in `dunif()`). For example, the uniform(0, 1) density at $x = 0.5$ is:

```
dunif(x = 0.5, min = 0, max = 1)

## [1] 1
```

Thus $f(0.5) = 1$.

10.3.2 Normal Random Variables and Probability Distributions

- `rnorm()`, `pnorm()`, `qnorm()`, and `dnorm()` return random samples, probabilities, percentiles, and the density function for the normal(μ, σ) distribution, with μ and σ specified via `mean` and `sd`.

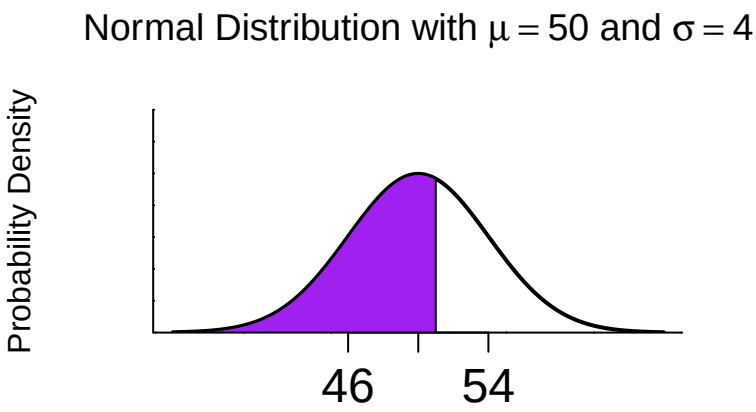
For example, to generate a random sample of size $n = 3$ from a normal(μ, σ) distribution with $\mu = 50$ and $\sigma = 4$, we type:

```
rnorm(n = 3, mean = 50, sd = 4)

## [1] 57.62865 54.57951 46.94188
```

As another example, to find the probability $P(X \leq 51)$, where X is a normal(50, 4) random variable, we type:

```
pnorm(q = 51, mean = 50, sd = 4)

## [1] 0.5987063
```

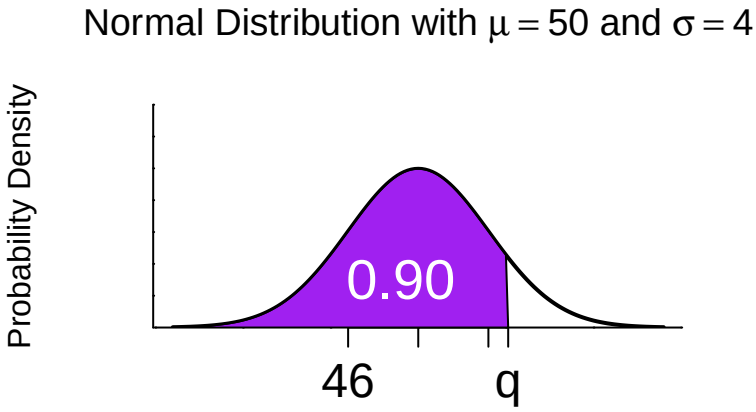
Thus the probability is 0.59871, or about 59.9%. This is the shaded area in the figure below.



As another example, to find the 90th percentile of the normal(50, 4) distribution, we type:

```
qnorm(p = 0.90, mean = 50, sd = 4)

## [1] 55.12621
```

Thus the percentile is $q = 55.12621$. This is represented by the value q in the figure below.



As a final example, to obtain the normal(μ, σ) ***probability density function***

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

at the value $x = 46$, using $\mu = 50$ and $\sigma = 4$, we type:

```
dnorm(x = 46, mean = 50, sd = 4)

## [1] 0.06049268
```

Thus $f(46) = 0.06049268$.

10.3.3 Binomial Random Variables and Probability Distributions

- `rbinom()`, `pbinom()`, `qbinom()`, and `dbinom()` return random samples, probabilities, percentiles, and the probability mass function for the binomial(n, p) distribution, with n and p specified via `size` and `prob`.

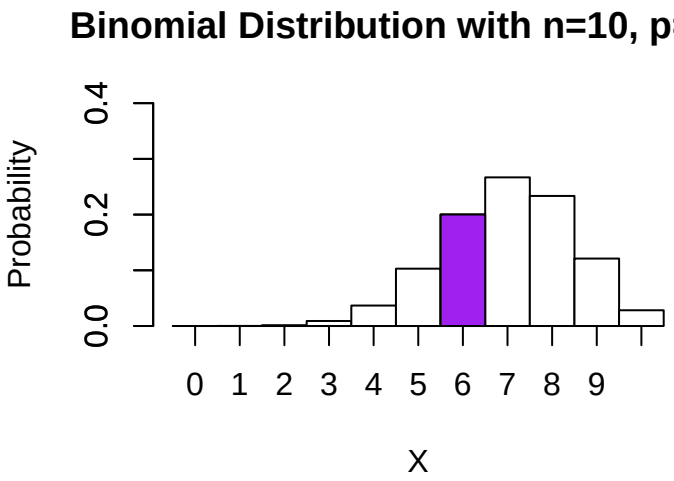
  The function `dbinom()` computes probabilities of the form $P(X = x)$, where X is a binomial(n, p) random variable, which are given by the binomial(n, p) ***probability mass function***

$$P(X = x) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}.$$

  For example, to find $P(X = 6)$, where X is a binomial(10, 0.7) random variable, we type:

```
dbinom(x = 6, size = 10, prob = 0.7)

## [1] 0.2001209
```

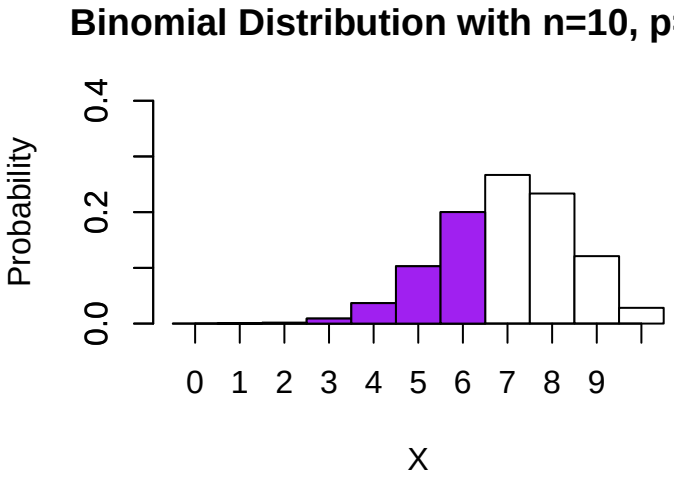
  Thus the desired probability is 0.20012. This is represented by the height of the shaded bar in the figure below.



  As another example, to find $P(X \leq 6)$, where X is a binomial(10, 0.7) random variable, we type:

```
pbinom(q = 6, size = 10, prob = 0.7)

## [1] 0.3503893
```

  Thus the desired probability is 0.35039, which is represented by the sum of the heights of the shaded bars in the figure below.



Section 10.3 Exercises

Exercise 1 Recall that `set.seed()` is used when we want to reproduce a set of random values later.

a) Use `set.seed()` to set the "seed" (any positive integer will do). Then use `sample()` to generate $n = 5$ random numbers from $1, 2, \ldots, 100$:

```
sample(1:100, size = 5)
```

Now set the "seed" again (to the same value), and use `sample()` to regenerate the five random numbers. Confirm that you get the same five values that you got the first time.

b) What would've happened in part *b* if you hadn't set the "seed" prior to generating the random values? Try it.

Exercise 2 Use `runif()` to generate $n = 300$ random values between 0 and 1. Save them in a vector `x`.

Now use `rnorm()` to generate $n = 300$ random values from the normal(μ, σ) distribution, with with $\mu = 0$ and $\sigma = 1$. Save them in a vector `y`.

Now make a histograms of the uniform and normal random values:

```
hist(x)
dev.new()      # Opens a new graphics window (graphics "de-
vice") and makes it "active"
hist(y)
```

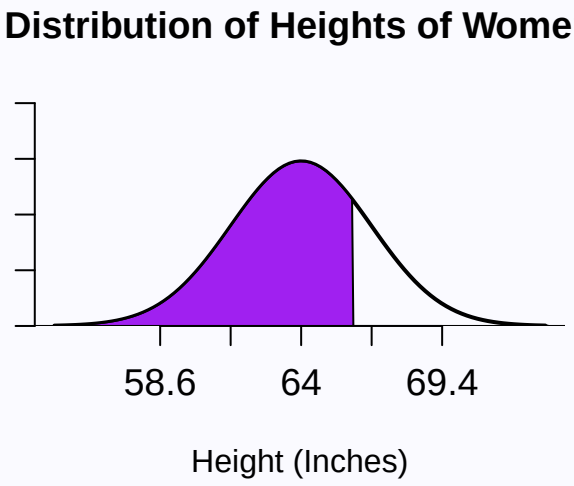
Describe the shapes of the distributions in the histograms.

When you're done, close the second graphics window:

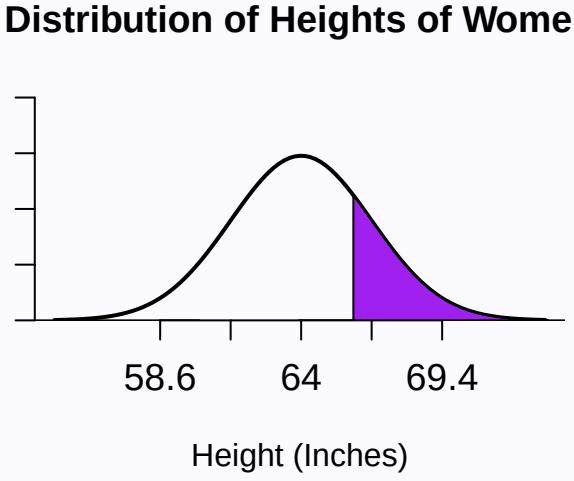
```
dev.off()      # Closes the current ("active") graphics window (graphics "de-
vice")
```

Exercise 3 Heights of women aged 20 to 29 follow a normal(μ, σ) distribution with mean $\mu = 64$ inches and standard deviation $\sigma = 2.7$ inches.

a) Use `pnorm()` to determine the percentage of women that are shorter than 66 inches (i.e. the probability that a randomly selected woman will be shorter than than 66 inches). This is represented by the shaded area below.

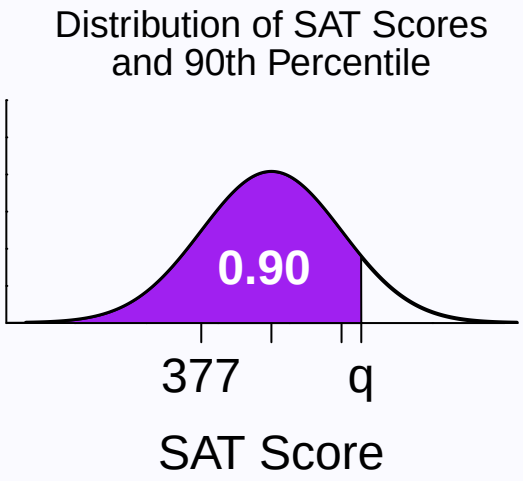


b) Use `1 - pnorm()` to find out what percentage of women are *taller* than 66 inches (i.e. the probability that a randomly selected woman will be *taller than* than 66 inches). This is represented by the shaded area below. You could also use `pnorm()` with `lower.tail = FALSE`.



Exercise 4 Scores on the verbal Scholastic Aptitude Test (SAT) follow a normal(μ, σ) distribution, with mean $\mu = 475$ and standard deviation $\sigma = 98$.

The *90th percentile* of the distribution of SAT scores is the score below which 90% of all scores lie. It's the value marked q on the horizontal axis below.



Use `qnorm()` to find the value of the 90th percentile of the distribution of SAT scores.

Exercise 5 For a random variable T following the t distribution with 16 df, the table of t distribution *tail areas* in the textbook gives the following values:

$$P(T > 1.6) = 0.065 \quad P(T > 1.8) = 0.045$$
$$P(T > 2.6) = 0.010 \quad P(T > 3.1) = 0.003$$

Verify that the following commands involving `pt()` give the same values:

```
pt(q = 1.6, df = 16, lower.tail = FALSE)
pt(q = 1.8, df = 16, lower.tail = FALSE)
pt(q = 2.6, df = 16, lower.tail = FALSE)
pt(q = 3.1, df = 16, lower.tail = FALSE)
```

Exercise 6 The table of t distribution *critical values* in the textbook gives the following:

$$t_{0.05,24} = 1.711 \quad t_{0.025,24} = 2.064$$
$$t_{0.01,24} = 2.492 \quad t_{0.005,24} = 2.797$$

Verify that the following commands involving `qt()` give the same values:

```
qt(p = 0.95, df = 24)
qt(p = 0.975, df = 24)
qt(p = 0.99, df = 24)
qt(p = 0.995, df = 24)
```

Exercise 7 We'll compare critical values for the $t(n-1)$ and $\mathrm{N}(0,1)$ distributions.

a) Guess which critical value, $t_{0.025,4}$ or $z_{0.025}$, will be larger. ***Hint***: Which distribution, the $t(4)$ distribution or the $\mathrm{N}(0,1)$ distribution, has more probability in its tails? Check your answer using:

```
qt(p = 0.975, df = 4)
```

and

```
qnorm(p = 0.975, mean = 0, sd = 1)
```

b) Guess how the values of $t_{0.025,1000}$ and $z_{0.025}$ will compare. ***Hint***: What is true about the $t(n-1)$ distribution, relative to the $\mathrm{N}(0,1)$ distribution, when the degrees of freedom $n-1$ is large? Check your answer using:

```
qt(p = 0.975, df = 1000)
```

and

```
qnorm(p = 0.975, mean = 0, sd = 1)
```

Exercise 8 Suppose X is a binomial$(n = 8, p = 0.7)$ random variable.

a) Write a command involving `dbinom()` that computes $P(X = 6)$.

b) Write a command involving `pbinom()` that computes $P(X \leq 6)$.