Lab 4

Y. Samuel Wang

2/19/2022

Distributions in R

In R, there are many functions which allow us to sample from or compute with probability distributions. Today we'll look at the normal, T, and χ^2 (pronounced `chi squared'' where`chi'' is pronounced like the first part of "cayanne pepper'') but the structure applies to many more types of distributions.

The function names have the same type of structure: the letter 'd', 'p', 'q', or 'r' followed by a name of the distribution.

- 'r': samples **r**andom values from the distribution
- 'd': computes the value of the **d**ensity at specific value
- 'p': computes the value of the cumulative distribution function at a specific value; i.e., the **p**robability $P(X < x)$
- 'q': computes the value of the **q**uantile function; i.e., given some value $0 \leq \alpha \leq 1$, what is the value of x such that $P(X < x) = \alpha$.

Random draws: rnorm

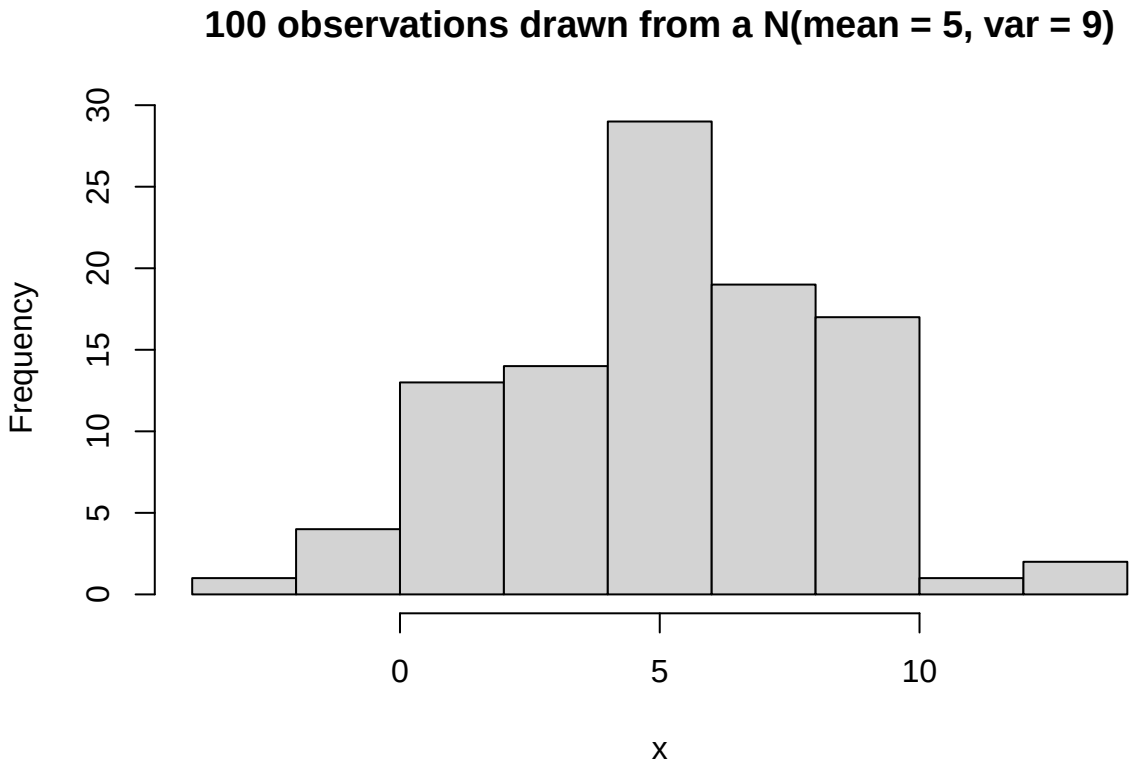
We used the `rnorm` function last week to draw random values from a normal distribution. The `rnorm` function takes 3 arguments:

- n: the number of random observations to draw
- mean: the mean of the normal distribution that the observatios will be drawn from
- sd: the standard deviation of the normal distribution that the observatios will be drawn from

If you don't enter in values, R will use the default values of mean = 0, and sd = 1. Often when we write out a normal distribution in mathematical notation, we use something like $N(3, 4)$ to indiciate a normal distribution with mean 3 and variance 4. Specifying the variance is different than the way that R specifies the normal distribution with the standard deviation so be careful when you are coding!

```
# Draw 100 observations from a normal distirbution with mean = 5, and sd = 3
x <- rnorm(100, mean = 5, sd = 3)

# plot observations
hist(x, main = "100 observations drawn from a N(mean = 5, var = 9)")
```




```
## the mean and variance of the sample aren't exactly the same as the population values
## and the values that you get when you run this will change each time
mean(x)
```

```
## [1] 5.099103
```

```
var(x)
```

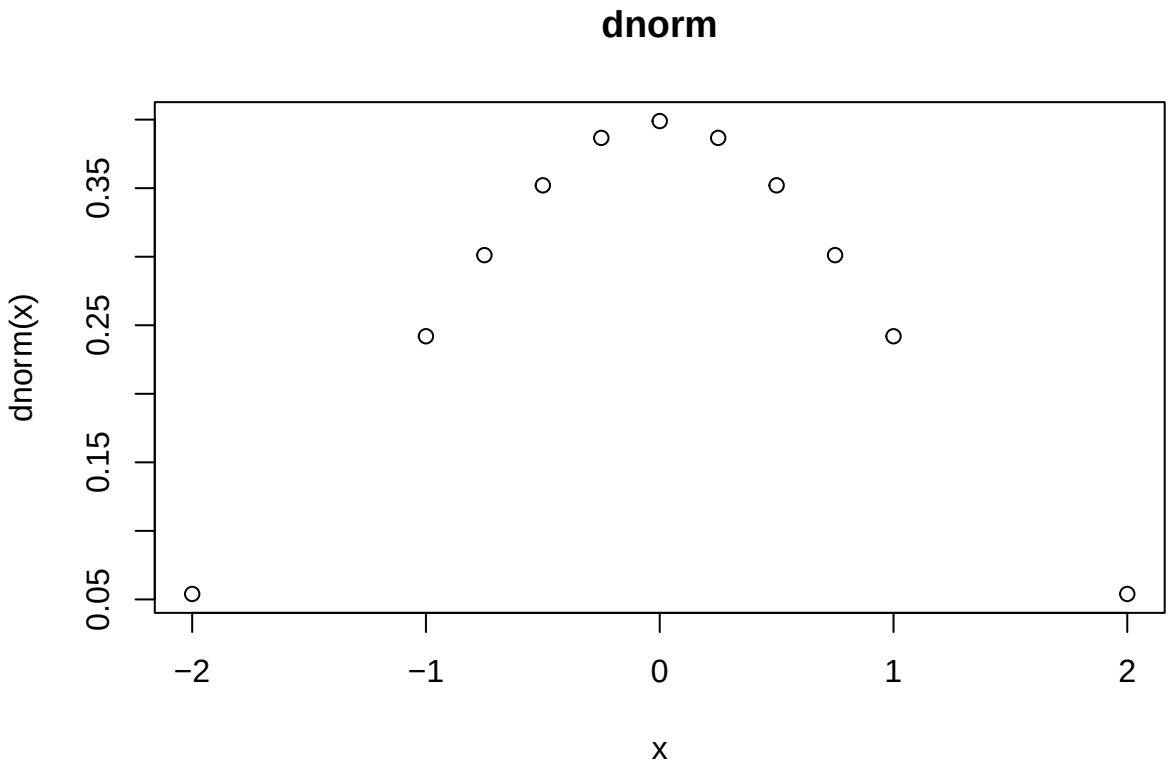
```
## [1] 9.516851
```

Density functions: dnorm

The `dnorm` takes a point (or vector of points) and evalutes the density function at that point. We will also use the `seq` function which creates a sequence of points

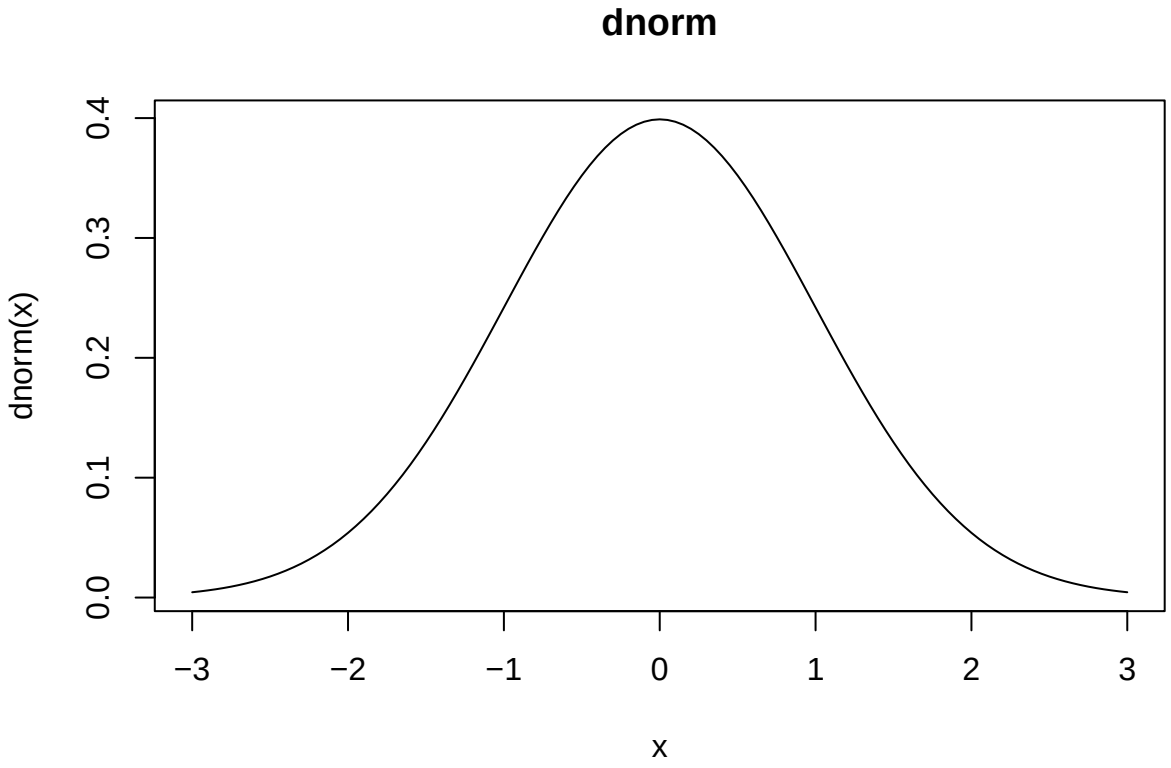
```
x <- c(-2, -1, -.75, -.5, - .25, 0, .25, .5, .75, 1, 2)

# plot the density evaluated at certain points
# since we don't specify a mean and sd, R uses the defaults
# using 'type=l' makes it a line plot
plot(x, dnorm(x), main = "dnorm")
```




```
# seq creates of numbers which starts at 'from' and goes to 'to'
# each number is spaced apart by the 'by' argument
x <- seq(from = -3, to= 3, by = .05)

# we use the 'type = l' argument to use a line plot instead of points
plot(x, dnorm(x), main = "dnorm", type = "l")
```

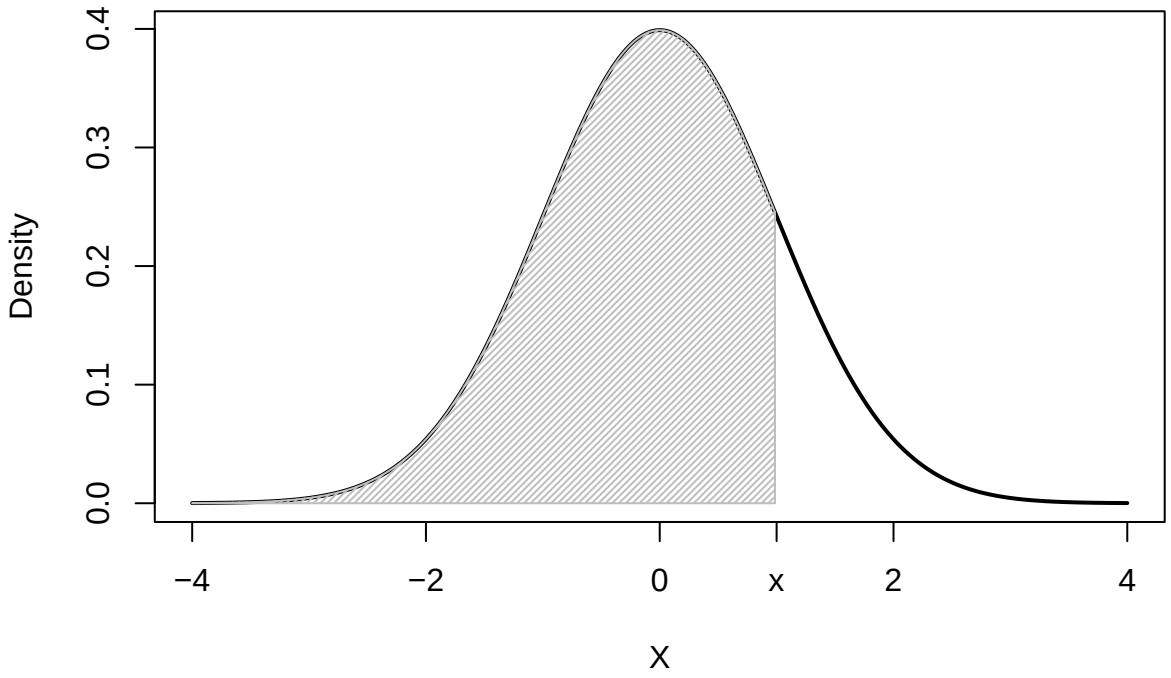

dnorm
dnorm(x)
0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3
x

Cumulative distribution functions: pnorm

The `pnorm` takes a point (or vector of points) and evalutes the cumulative distribution function at that point. This means, given some point x, what is the probability that a random draw from a given distribution is less than x. Written out in mathematical notation, this is:

$$P(X < x).$$

In the plot below, `pnorm(x, 0, 1)` would return the area of the shaded region where the density plotted corresponds to a normal distribution with mean 0 and standard deviation 1.



For example, we can see that the probability that an observation less than 0 is drawn from a normal distribution with mean = 0 and sd = 1 is .5 because the median of the normal distribution is also the mean.

```
pnorm(0, mean = 0, sd = 1)
```

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

The probability that an observation from $N(0, 1)$ is less than 1—i.e., $P(X < 1)$—is the value:

```
pnorm(1, mean = 0, sd = 1)
```

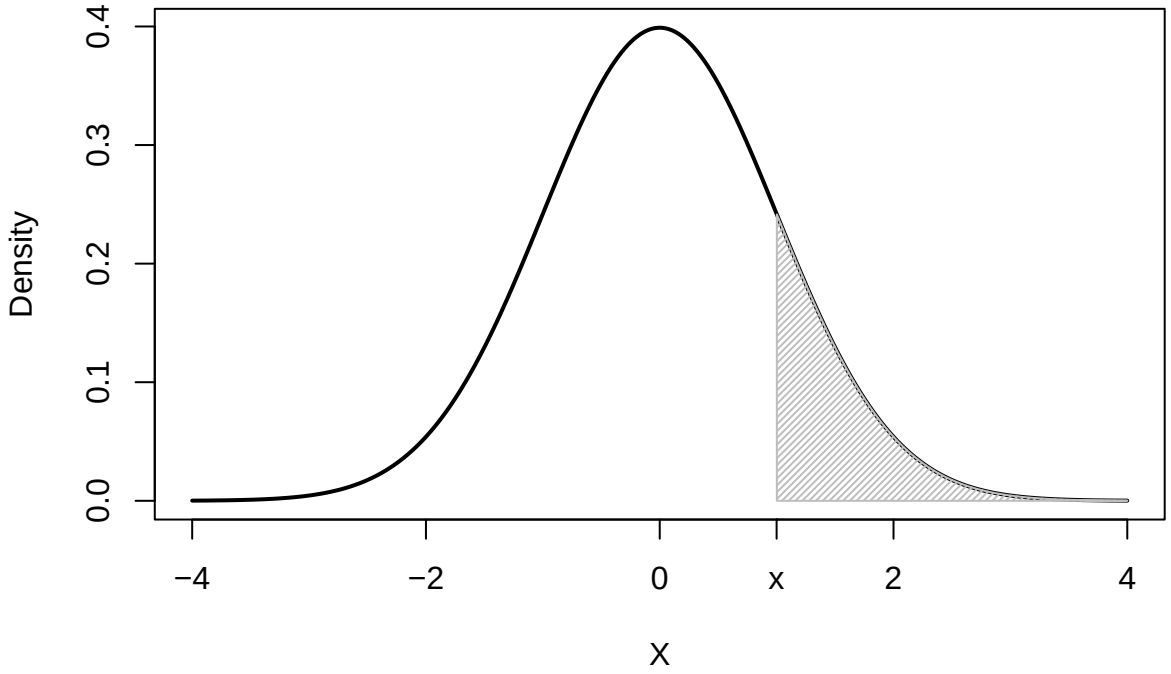
```
## [1] 0.8413447
```

Questions

- Suppose X is a normal distribution with mean .5 instead of 0 but the standard deviation is still 1—i.e., $N(.5, 1)$—do you think the probability $P(X < 1)$ increases or decrease compared to when X is $N(0, 1)$? Why?
- Suppose X is a normal distribution with mean 0 but the standard deviation is 2 instead of 1—i.e., $N(0, 2)$—do you think the probability $P(X < 1)$ increases or decreases compared to when X is $N(0, 1)$? Why?
- Using R, check whether you are correct
- Evaluate the following probabilities
  - If X is drawn from $N(2, 2)$ what is $P(X < 3)$?
  - If X is drawn from $N(2, 2)$ what is $P(X > 3)$?

– If X is drawn from $N(-2, 1)$ what is $P(-3 < X > -1)$?

By default, the `lower.tail` argument is TRUE, so we calculate the area under the density function that is in the lower tail; i.e., $P(X < x)$. We could set `lower.tail = FALSE` to calculate the area under the density function that is in the upper tail. This would give us $P(X > x)$



Since the total area under the density is always equal to 1, we know that the area to the left of a value is always equal to 1 minus the area to the right of a value.

```
# area to the left of 1
pnorm(1, mean = 0, sd = 1)
```

```
## [1] 0.8413447
```

```
# area to the right of 1
pnorm(1, mean = 0, sd = 1, lower.tail = F)
```

```
## [1] 0.1586553
```

```
# 1 minus area to the right of 1
1 - pnorm(1, mean = 0, sd = 1, lower.tail = F)
```

```
## [1] 0.8413447
```

To get the probability that a random observation, X, is between two different numbers, x_1 and x_0,

$$P(x_0 < X < x_1),$$

we can simply subtract the area to the left of x_0 from the area to the left of x_1

```
pnorm(1, mean = 0, sd = 1)
```

```
## [1] 0.8413447
```

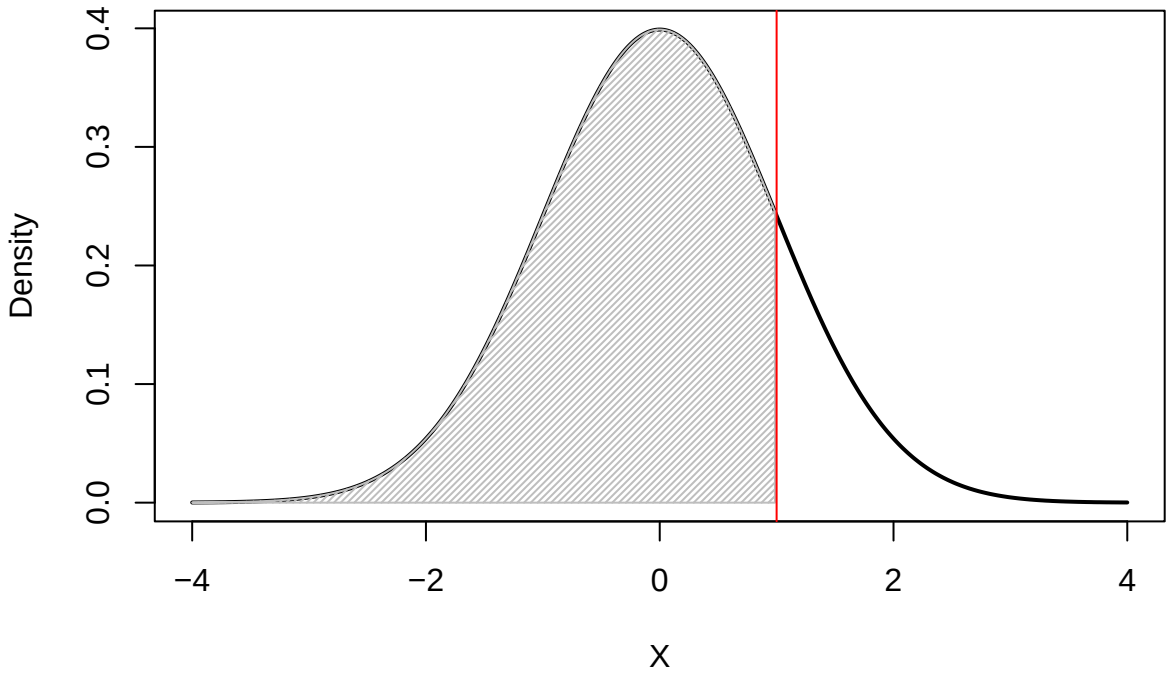
```
pnorm(0, mean = 0, sd = 1)
```

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

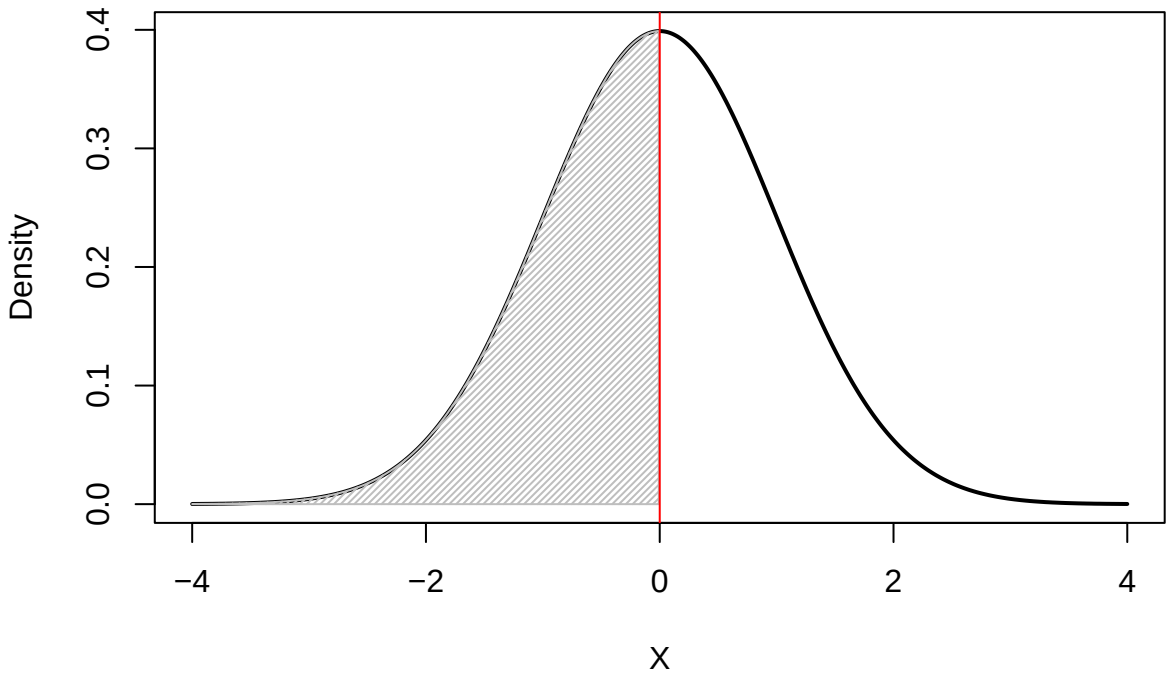
```
pnorm(1, mean = 0, sd = 1) - pnorm(0, mean = 0, sd = 1)
```

```
## [1] 0.3413447
```

```
shadenorm(below = 1, justbelow = T)
abline(v = 1, col = "red")
```

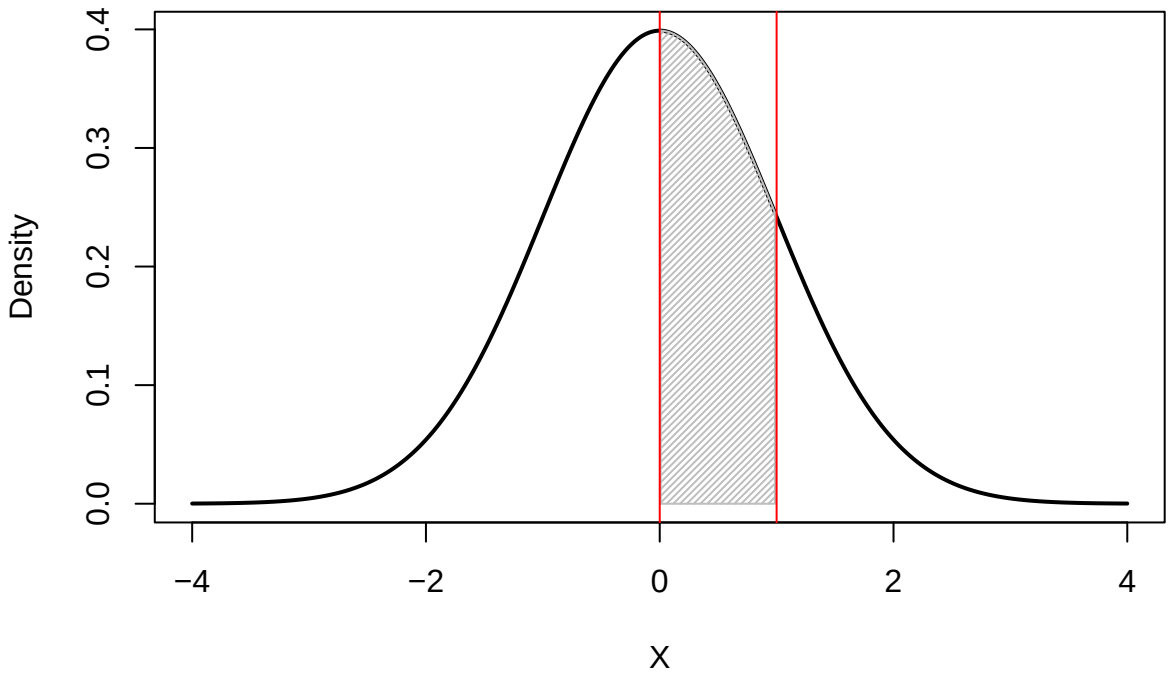



```
shadenorm(below = 0, justbelow = T)
abline(v = 0, col = "red")
```




```
shadenorm(between = c(0, 1))
abline(v = 1, col = "red")
```

```r
abline(v = 0, col = "red")
```



Quantile function: qnorm

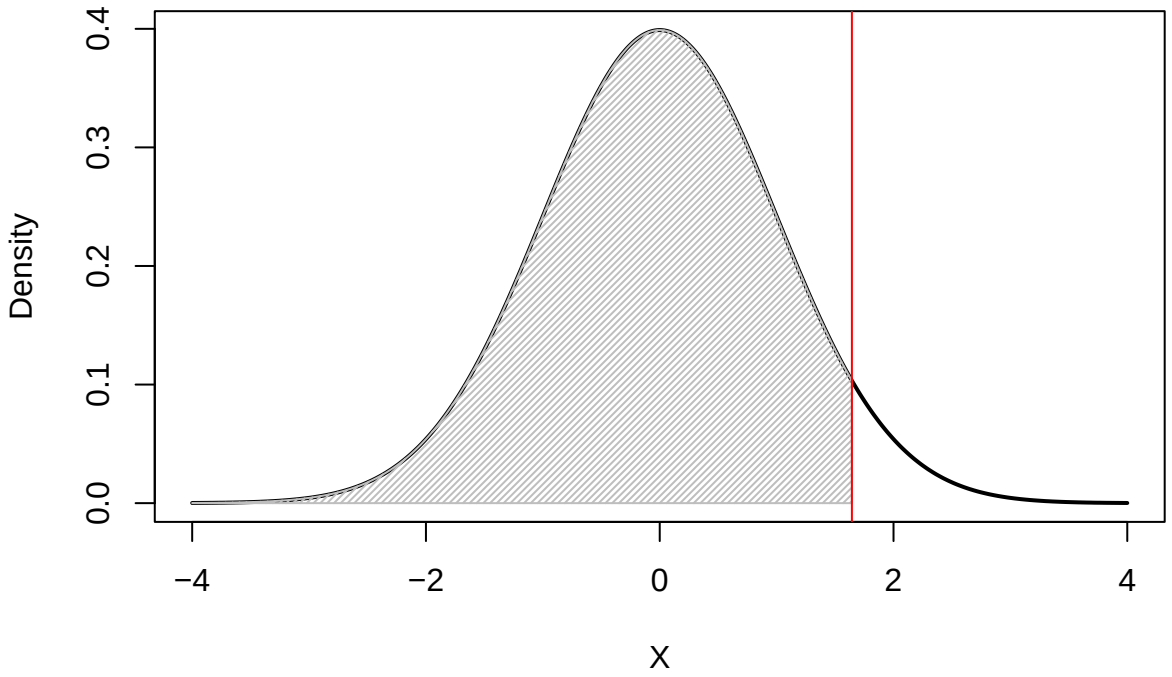
The `qnorm` function is the inverse of the pnorm function. Specifically, given a proportion, α it returns the value x such that $P(X < x) = \alpha$. If we want to know what value is larger than .95 of draws from a normal distribution with mean 0 and sd = 1:

```
qnorm(.95, mean = 0, sd = 1)
```

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

So `qnorm(.95, mean = 0, sd = 1)` finds the value such that the shaded portion has an area equal to .95

```
shadenorm(below = 1.644, justbelow = T)
abline(v = 1.644, col = "red")
```

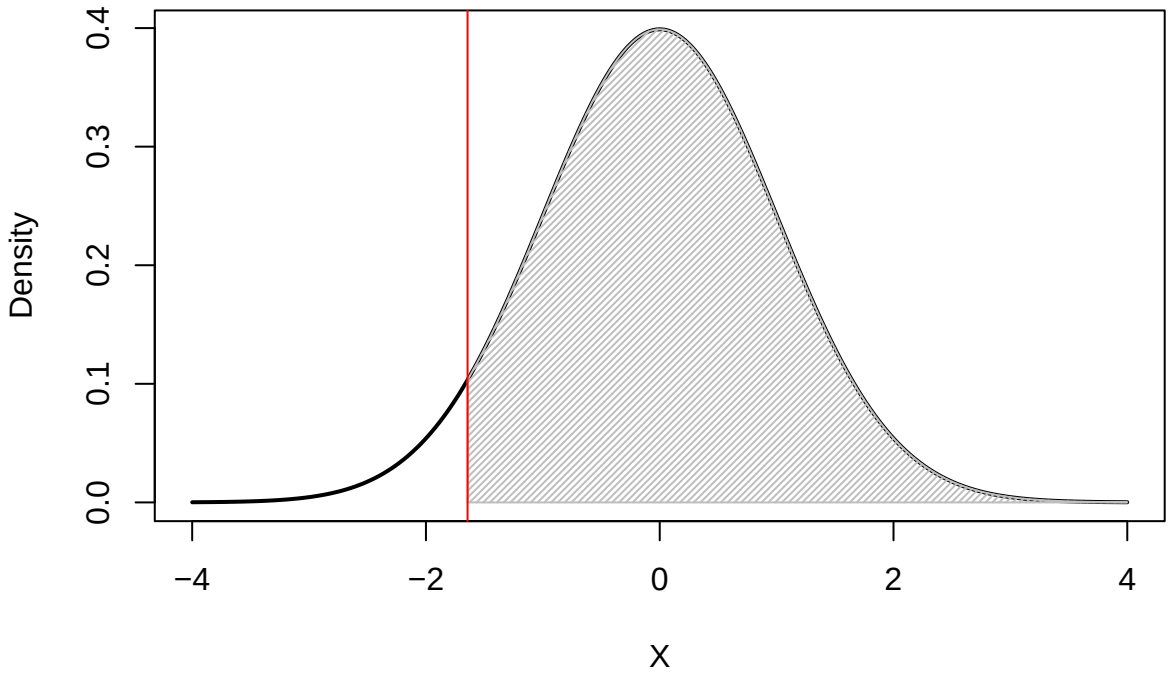


Similar to `pnorm`, the `lower.tail` option by default is set to `TRUE` so it assumes the area which is equal to α is in the lower.tail. If we set `lower.tail` to false, `qnorm` will calculate what value of x we'd need such that $P(X > x) = \alpha$.

```
qnorm(.95, mean = 0, sd = 1, lower.tail = F)
```

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

```
shadenorm(above = -1.644, justabove = T)
abline(v = -1.644, col = "red")
```



Questions

- Suppose X is a normal distribution with mean .5 instead of 0 but the standard deviation is still 1—i.e., $N(.5, 1)$—do you think `qnorm(.95, mean = .5, sd = 1)` increases or decrease compared to when X is $N(0, 1)$? Why?
- Suppose X is a normal distribution with mean 0 but the standard deviation is 2 instead of 1—i.e., $N(0, 2)$—do you think `qnorm(.95, mean = .5, sd = 1)` increases or decreases compared to when X is $N(0, 1)$? Why?
- Using R, check whether you are correct
- Evaluate the following probabilities
  - If X is drawn from $N(2, 2)$ what is x such that for $\alpha = .025$ we have $P(X < x) = \alpha$?
  - If X is drawn from $N(2, 2)$ what is x such that for $\alpha = .025$ we have $P(X > x) = \alpha$?

Different Distributions

We can do the same thing, but with different distributions. In particular, we can use the T distribution and the χ^2 distribution.

- `rt`, `dt`, `pt`, `qt`
- `rchisq`, `dchisq`, `pchisq`, `qchisq`

Instead of specifying the mean and sd of these distributions, the T distribution and the χ^2 distribution have a single parameter called the *degrees of freedom* which we sepcify using the `df` argument.

Sampling distribution of $\hat{\sigma}^2_\varepsilon$

Let's take a look at the sampling distribution of $\hat{\sigma}^2_\varepsilon$. We are re-using the same code from last week, but with a few modifications.

In particular, we now let $p = 3$ where p is the number of covariates. We also record three different estimators of the variance of ε_i:

- An estimate which uses the true errors. In practice, we can't compute this since we won't know the true errors, but since this is simulated data, we can.

$$\frac{1}{n}\sum_i \varepsilon_i^2$$

- An estimate which uses the residuals, $\hat{\varepsilon}_i = y_i - \sum_k^p \hat{b}_k x_{i,k}$, but doesn't adjust for the fact that we are using residuals and not the true errors.

$$\frac{1}{n}\sum_i \hat{\varepsilon}_i^2$$

- An estimate which uses the residuals $\hat{\varepsilon}_i = y_i - \sum_k^p \hat{b}_k x_{i,k}$, but does adjust for the fact that we are using residuals and not the true errors by dividing by $n - p - 1$

$$\frac{1}{n-p-1}\sum_i \varepsilon_i^2$$

```r
# Number of times we will simulate a new data set
sim.size <- 10000

# number of observations
n <- 20
# number of covariates
p <- 3
# standard deviation of the X values
x.sd <- 1
# drawing the covariates from a normal distribution
X <- matrix(rnorm(n * p, sd = x.sd), n, p)

# We include a column of all 1's into the matrix of observations
# that column corresponds to the intercept term
X <- cbind(rep(1, n), X)
# coefficients are all set to 1 (including the intercept)
beta <- rep(1, p + 1)

# recording the estimated values for each simulated data set
rec <- matrix(0, sim.size, 3)

for(i in 1:sim.size){

  # Sample errrors from a normal distribution with mean 0 and sd = 1
  errs <- rnorm(n, mean = 0, sd = 1)

  # Form the dependent variable Y
  Y.norm <- X %*% beta + errs

  # Fit the regression
  # we include the -1 term to tell R not to add in an intercept term
  # since we've manually included the column of 1's in the matrix X
  reg_norm <- lm(Y.norm ~X - 1)

  # RSS(b) / n: we can calculate this using the true errors, which we know because
  # it's a simulation, but in practice we would need to know the true coefficients
  # to calculate the true errors
  true_errors <- sum(errs^2) / n
```

```
  # RSS(b hat) / n: we can calculate this using the residuals, but we don't
  # adjust for the fact that we are using residuals and not the true errors
  resid_unadjust <- sum(reg_norm$res^2) / n

  # RSS(b hat) / (n-p): we can calculate this using the residuals, and now we
  # adjust for the fact that we are using residuals and not the true errors
  resid_adjust <- sum(reg_norm$res^2) / (n-p - 1)

  # record each of the estimators
  rec[i, ] <- c(true_errors,
                resid_unadjust,
                resid_adjust)
}
```

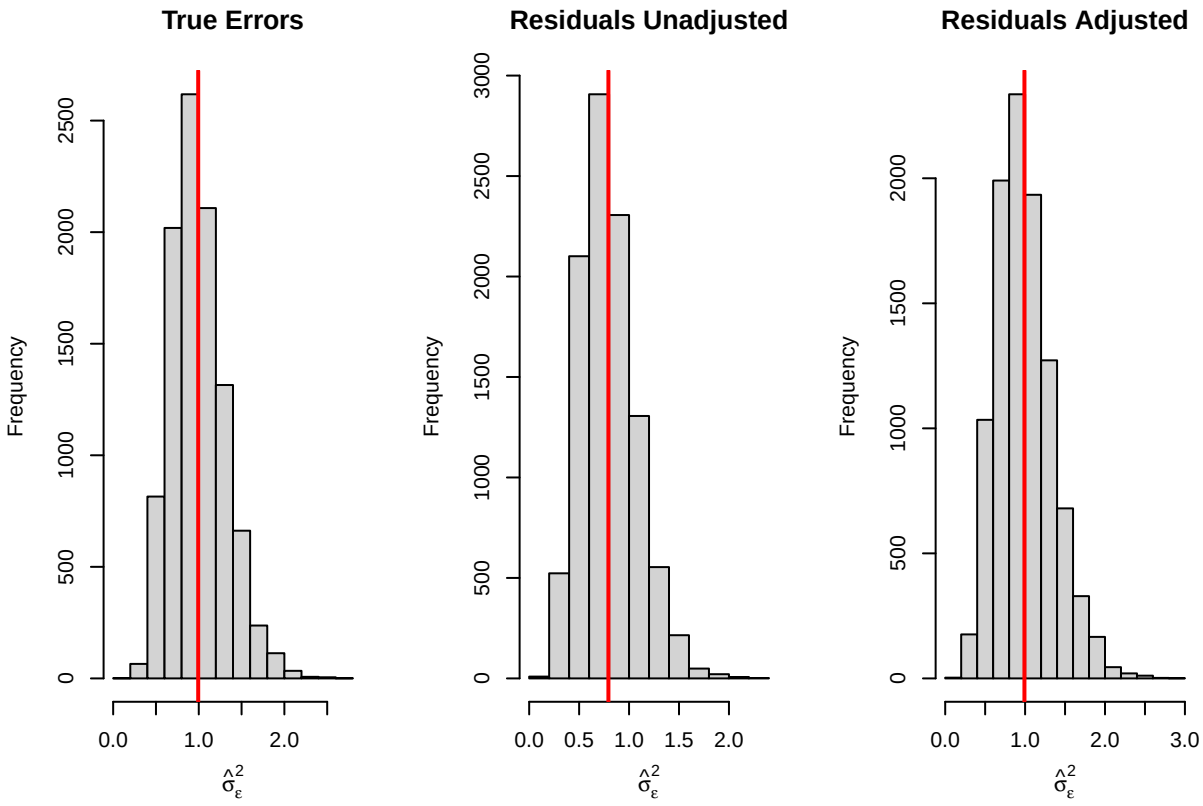
We can plot histograms of each of the estimators

```
par(mfrow = c(1,3))
# We're using some fancy code to label the axis
# We won't cover this because of time, but the following is a good tutorial
# if you are interested in learning more:
# https://www.dataanalytics.org.uk/axis-labels-in-r-plots-using-expression/

# Histogram of estimator using true errors
hist(rec[, 1], main = "True Errors", xlab = expression(hat(sigma)[epsilon]^2))
# draw a red vertical line at the mean
abline(v = mean(rec[, 1]), col = "red", lwd = 2)

# Histogram of estimator using residuals, but unadjusted
hist(rec[, 2], main = "Residuals Unadjusted", xlab = expression(hat(sigma)[epsilon]^2))
# draw a red vertical line at the mean
abline(v = mean(rec[, 2]), col = "red", lwd = 2)

# Histogram of estimator using residuals, but adjusted
hist(rec[, 3], main = "Residuals Adjusted", xlab = expression(hat(sigma)[epsilon]^2))
# draw a red vertical line at the mean
abline(v = mean(rec[, 3]), col = "red", lwd = 2)
```



We can calculate the mean and variance of each of the estimators. As we can see, the mean of the estimators using the true errors and the mean of the estimator which uses the residuals and adjusts for them are pretty close to the actual value of $\sigma^2_\varepsilon = 1$. However, the mean of the estimator using the residuals and not adjusting is further from the true value.

```
# mean and variance of the estimator using the true errors
mean(rec[, 1])
```

```
## [1] 0.9934826
```

```
var(rec[, 1])
```

```
## [1] 0.09932175
```

```
# mean and variance of the estimator using the residuals but not adjusting
mean(rec[, 2])
```

```
## [1] 0.7933105
```

```
var(rec[, 2])
```

```
## [1] 0.07873932
```

```
# mean and variance of the estimator using the residuals and adjusting
mean(rec[, 3])
```

```
## [1] 0.9916382
```

```
var(rec[, 3])
```

```
## [1] 0.1230302
```

Questions:

- Keep $n = 20$ but increase p to be $5, 10, 15$. What happens to the mean of each of the estimators? What happens to the variance of each of the estimators? Keep $p = 3$ but increase n to be $50, 100, 150$. What happens to the mean of each of the estimators? What happens to the variance of each of the estimators?