**Doing Normal Calculations in R**

In class we learned about the normal distribution and how to do probability calculations involving the normal distribution. Recall that the steps involved standardizing our value to find the z-score (by subtracting the mean and dividing by the standard deviation) and then looking up the value in our charts.



Then, depending upon whether we were looking for the probability we were “greater than,” “less than,” or “between” particular values, we performed the appropriate algebraic operation. Also, if we wanted to find the value above or below which a certain percentage of the data was located, we would do the inverse of this calculation.

As a refresher, let’s do a quick normal calculation by hand.

**Your turn:**

Human body temperatures for healthy individuals have approximately a normal distribution with mean 98.250F and standard deviation of 0.750F.

a) What percentage of healthy individuals have a body temperature below 970F?

b) The 10% of people with the highest body temperatures have a body temperature of at least how many degrees?

Um…. Jeff?

Yes, Trent?

This is a computer lab. You’re not going to be having them do calculations by hand, are you?

Ummm…. No?

**The Normal Distribution in R**

Fortunately, we don’t have to perform normal calculations by hand (except perhaps on tests). R has several commands that allow you to perform the mathematics necessary without using charts. In fact, R also has several commands that deal with distributions besides the normal. These commands are all based around four letters: **d, p, q,** and **r.** Each of these commands serves a particular purpose for which the letter is an abbreviation:

d = density

p = (cumulative) probability

q = quantile

r = random

Let’s go over the four commands as they apply to the normal distribution and show how they’re used.

**dnorm(x,mu,sigma)**

For our purposes, the dnorm command will not be that useful. You may recall that the normal distribution is defined by the following density function:



The dnorm command allows you to calculate the value of this function if you give R the value at which you want to calculate it (x), the mean (mu), and the standard deviation (sigma).

The applications of this function are generally beyond the scope of this course, but one thing we can do with the command is use it to create plots of the normal curve. For example, suppose we wish to plot the standard normal curve (mu=0, sigma=1) from -4 to 4. The following code will suffice:

x = seq(-4, 4, by = 0.01) # Note that I will evaluate the function every hundredth.

y = dnorm(x, 0, 1) # This allows me to evaluate the standard normal curve

# (mu=0, sigma=1) at every value of x and save it as y.

plot(x, y, type='l') # I create a scatterplot and connect the points with lines.

**Your turn:**

a) Run the above code. How does the figure look?

b) **Thought Challenge!** Can you think of any possible applications of the dnorm command?

**pnorm(x,mu,sigma)**

The pnorm command allows us to calculate the cumulative probability of the normal distribution. This is the answer to the following integral:



where *f(t)* is the normal distribution we discussed in the section about the dnorm command. This calculation is essential to finding the probability of obtaining a value less than a particular number. Of course, no one wants to do that integral by hand. Previously, we solved this problem by first standardizing our normal calculation and then looking up the value on the z-chart (where the integral had already been done for us). With a computer at hand, we don’t even need to use the z-chart! The computer will give us the value directly.

For example, suppose I wish to find the probability of obtaining a value less than 1.5 from a standard normal distribution. That is,  The command in R is simply:

pnorm(1.5, 0, 1)

**Note:** You still need to perform any applicable algebra to find the proper probability. R will only give you the probability that you are *less than* (or to the *left*) of a particular value since it returns the *cumulative* probability. Thus, you will need to use the following code in the following situations:

|  |  |
| --- | --- |
| Probability desired | Appropriate code |
| P(Z<x) | pnorm(x, 0, 1) |
| P(Z>x) | 1 – pnorm(x, 0, 1) |
| P(x<Z<y) | pnorm(y, 0, 1) – pnorm(x, 0, 1) |

**Your turn:**

Find the following probabilities using R. Compare the results to your z-tables.

a) 

b) 

c) 

d) 

Um…. Jeff?

Yes, Trent?

Why are you having them only do calculations with Z? Is that all you think pnorm is useful for?

Wait, wait. I’m getting there!

-------

We just showed that you don’t need to use your z-tables to look up the probabilities. The better part is we don’t even have to standardize our normal distribution first! R will do everything for us, as long as we provide the correct mean (mu) and standard deviation (sigma).

Let’s return to our first problem, involving human body temperatures. In order to find the percentage of healthy individuals have a body temperature below 970F, we would simply run the following code:

pnorm(97, 98.25, 0.75)

where the second and third numbers are the mean and standard deviation respectively. That’s a lot easier! (And it’s also how I do my answer keys, although I do show you all the steps.) Note that as before you will need to perform any applicable algebra to find the correct probability for your problem as pnorm only gives the *cumulative* probability.

**Note:** If you omit the mean and standard deviation when you use the pnorm command, the default settings are mu=0 and sigma=1. That is, it will assume you want to use the standard normal distribution. (On the previous page, I included the 0 and 1 for completeness, even though they weren’t necessary.)

**Your turn:**

The plasma cholesterol level (mg/dL) for patients with no prior evidence of heart disease is normally distributed with mean 200 and standard deviation 35. Suppose you randomly select an individual of this type. What is the probability that the plasma cholesterol level:

a) Is at least 250?

b) Is between 300 and 400?

c) **Mini-challenge!** Differs from the mean by at least 1.5 standard deviations?

**qnorm(p,mu,sigma)**

The qnorm command allows us to perform the inverse calculation for the cumulative probability. Thus, instead of finding the probability that we obtain a value less than a given number, we find the value (x) such that the probability we obtain a value less than it is equal to our desired probability (p). Like the pnorm command, qnorm does not require us to standardize our normal distribution first. We only need to provide the correct mean and standard deviation.

For example, let’s solve the second part of the original body temperature problem: the 10% of people with the highest body temperatures have a body temperature of at least how many degrees? One thing to note here is that the 10% of the people with the highest body temperature have temperature warmer than 90% of the population. Thus, our desired probability is 0.90 as opposed to 0.10 when we do the inverse calculation. The correct code is:

qnorm(0.9, 98.25, 0.75)

Similarly to pnorm, if you omit the mean and standard deviation, the default is the standard normal distribution.

**Note:** Since the first argument (p) is a probability, only valid probabilities will be accepted by R. Thus, the values must be between 0 and 1.

**Your turn:**

Suppose that the diameter at breast height of trees of a certain type is normal distributed with mean equal to 8.8 inches and standard deviation equal to 2.8 inches.

a) Find the 85th percentile of the distribution.

b) Find the 8th percentile of the distribution.

c) Below what diameter are the narrowest 23% of the trees?

d) Suppose I had said that the variance of the diameter of the trees was 5. What would you have had to do to solve the above problems?

**rnorm(n,mu,sigma)**

The final base function associated with the normal distribution is rnorm. This command allows you to generate a random draw from a normal distribution with the appropriate mean and standard deviation. You indicate how large of a sample you desire by adjusting the value of n.

For example, suppose I wish to draw 100 observations from a standard normal distribution and save them as the vector *z*. The simple code to do this is:

z = rnorm(100, 0, 1)

(And if I’m really lazy in this case, I don’t even need the 0 or 1.)

How closely does our generated data match the standard normal curve? Let’s check it out by creating a histogram and then superimposing the normal curve on top of it!

hist(z, probability = TRUE) # We create a density as opposed to a frequency histogram.

x = seq(-4, 4, by = 0.01)

y = dnorm(x, 0, 1) # Using dnorm!

lines(x, y) # Using the lines command to add the lines to the existing

# histogram.

**Your turn:**

a) Run the above code. How well does the histogram match the density curve?

b) Increase the size of your random sample and rerun the code. Now how does it match?

c) **Thought Challenge!** Can you think of any situations where you would want to randomly generate a sample from a normal distribution?