Introduction to dnorm, pnorm, qnorm, and rnorm for new biostatisticians

Sean Kross

October 1, 2015

Today I was in Dan's office hours and someone asked, "what is the equivalent in R of the back of the stats textbook table of probabilities and their corresponding Z-scores?" (This is an example of the kind of table the student was talking about.) This question indicated to me that although we've been asked to use some of the distribution functions in past homeworks, there may be some misunderstanding about how these functions work.

Right now I'm going to focus on the functions for the normal distribution, but you can find a list of all distribution functions by typing `help(Distributions)` into your R console.

`dnorm`

As we all know the probability density for the normal distribution is:

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

The function `dnorm` returns the value of the probability density function for the normal distribution given parameters for x, μ, and σ. Some examples of using `dnorm` are below:

```r
# This is a comment. Anything I write after the octothorpe is not executed.

# This is the same as computing the pdf of the normal with x = 0, mu = 0 and
# sigma = 0. The dnorm function takes three main arguments, as do all of the
# *norm functions in R.

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

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

```r
# The line of code below does the same thing as the same as the line of code
# above, since mean = 0 and sd = 0 are the default arguments for the dnorm
# function.

dnorm(0)
```

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

```r
# Another exmaple of dnorm where parameters have been changed.

dnorm(2, mean = 5, sd = 3)
```

```
## [1] 0.08065691
```

Although x represents the independent variable of the pdf for the normal distribution, it's also useful to think of x as a Z-score. Let me show you what I mean by graphing the pdf of the normal distribution with `dnorm`.

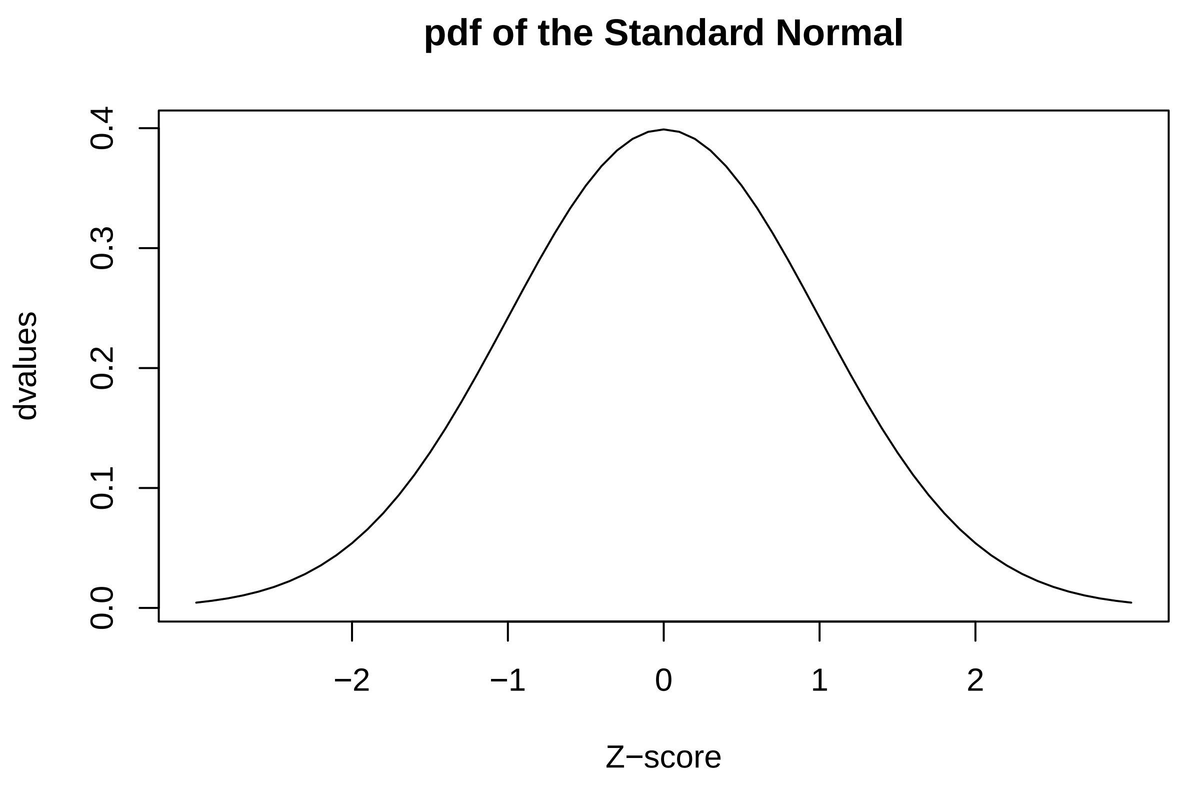
```r
# First I'll make a vector of Z-scores
z_scores <- seq(-3, 3, by = .1)

# Let's print the vector
z_scores
```

```
##  [1] -3.0 -2.9 -2.8 -2.7 -2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -1.9 -1.8 -1.7 -1.6
## [16] -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1
## [31]  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4
## [46]  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9
## [61]  3.0
```

```r
# Let's make a vector of the values the function takes given those Z-scores.
# Remember for dnorm the default value for mean is 0 and for sd is 1.
dvalues <- dnorm(z_scores)

# Let's examine those values
dvalues
```

```
##  [1] 0.004431848 0.005952532 0.007915452 0.010420935 0.013582969 0.017528300
##  [7] 0.022394530 0.028327038 0.035474593 0.043983596 0.053990967 0.065615815
## [13] 0.078950158 0.094049077 0.110920835 0.129517596 0.149727466 0.171368592
## [19] 0.194186055 0.217852177 0.241970725 0.266085250 0.289691553 0.312253933
## [25] 0.333224603 0.352065327 0.368270140 0.381387815 0.391042694 0.396952547
## [31] 0.398942280 0.396952547 0.391042694 0.381387815 0.368270140 0.352065327
## [37] 0.333224603 0.312253933 0.289691553 0.266085250 0.241970725 0.217852177
## [43] 0.194186055 0.171368592 0.149727466 0.129517596 0.110920835 0.094049077
## [49] 0.078950158 0.065615815 0.053990967 0.043983596 0.035474593 0.028327038
## [55] 0.022394530 0.017528300 0.013582969 0.010420935 0.007915452 0.005952532
## [61] 0.004431848
```

```r
# Now we'll plot these values
plot(dvalues, # Plot where y = values and x = index of the value in the vector
     xaxt = "n", # Don't label the x-axis
     type = "l", # Make it a line plot
     main = "pdf of the Standard Normal",
     xlab= "Z-score")

# These commands label the x-axis
axis(1, at=which(dvalues == dnorm(0)), labels=c(0))
axis(1, at=which(dvalues == dnorm(1)), labels=c(-1, 1))
axis(1, at=which(dvalues == dnorm(2)), labels=c(-2, 2))
```



As you can see, `dnorm` will give us the "height" of the pdf of the normal distribution at whatever Z-score we provide as an argument to `dnorm`.

`pnorm`

The function `pnorm` returns the integral from $-\infty$ to q of the pdf of the normal distribution where q is a Z-score. Try to guess the value of `pnorm(0)`. (`pnorm` has the same default `mean` and `sd` arguments as `dnorm`).

```
# To be clear about the arguments in this example:
# q = 0, mean = 0, sd = 1
pnorm(0)
```

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

The `pnorm` function also takes the argument `lower.tail`. If `lower.tail` is set equal to `FALSE` then `pnorm` returns the integral from q to ∞ of the pdf of the normal distribution. Note that `pnorm(q)` is the same as `1-pnorm(q, lower.tail = FALSE)`

```
pnorm(2)
```

```
## [1] 0.9772499
```

```
pnorm(2, mean = 5, sd = 3)
```

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

```
pnorm(2, mean = 5, sd = 3, lower.tail = FALSE)
```

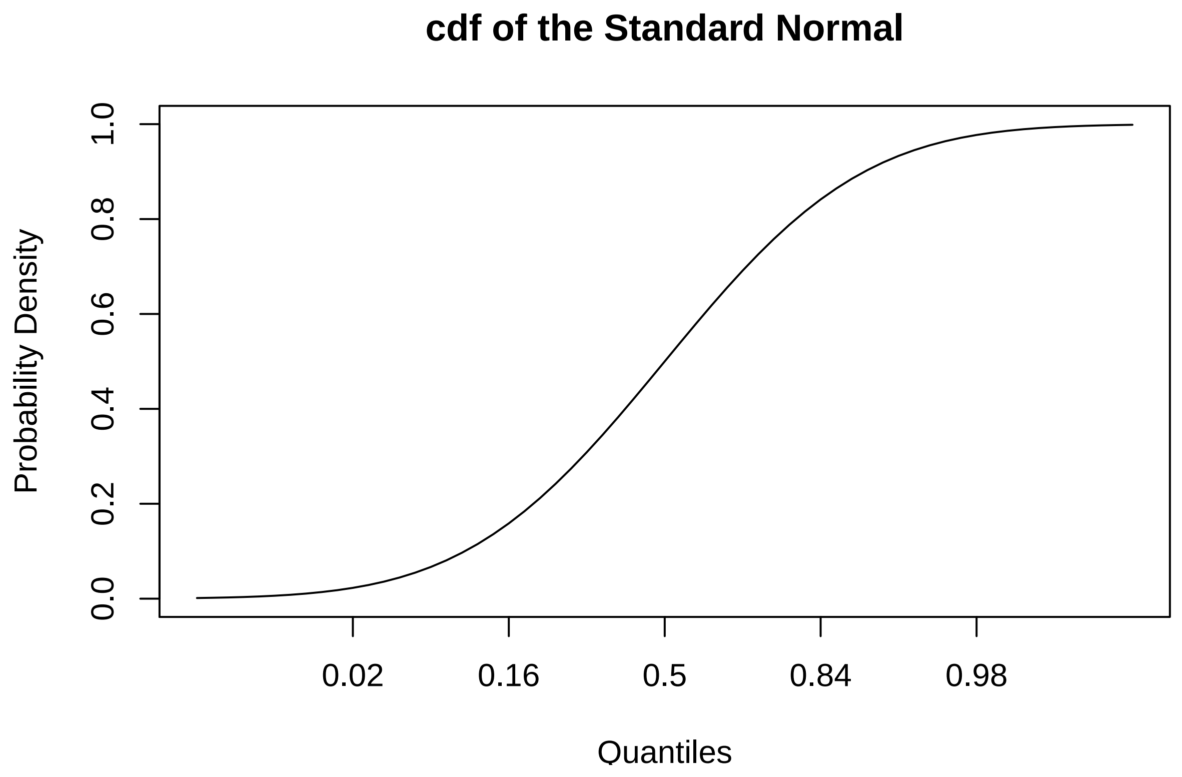
```
## [1] 0.8413447
```

```
1 - pnorm(2, mean = 5, sd = 3, lower.tail = FALSE)
```

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

`pnorm` is the function that replaces the table of probabilites and Z-scores at the back of the statistics textbook. Let's take our vector of Z-scores from before (`z_scores`) and compute a new vector of "probability masses" using `pnorm`. Any guesses about what this plot will look like?

```
pvalues <- pnorm(z_scores)

# Now we'll plot these values
plot(pvalues, # Plot where y = values and x = index of the value in the vector
     xaxt = "n", # Don't label the x-axis
     type = "l", # Make it a line plot
     main = "cdf of the Standard Normal",
     xlab= "Quantiles",
     ylab="Probability Density")

# These commands label the x-axis
axis(1, at=which(pvalues == pnorm(-2)), labels=round(pnorm(-2), 2))
axis(1, at=which(pvalues == pnorm(-1)), labels=round(pnorm(-1), 2))
axis(1, at=which(pvalues == pnorm(0)), labels=c(.5))
axis(1, at=which(pvalues == pnorm(1)), labels=round(pnorm(1), 2))
axis(1, at=which(pvalues == pnorm(2)), labels=round(pnorm(2), 2))
```



It's the plot of the cumulative distribution function of the normal distribution! Isn't that neat?

`qnorm`

The `qnorm` function is simply the inverse of the cdf, which you can also think of as the inverse of `pnorm`! You can use `qnorm` to determine the answer to the question: What is the Z-score of the *pth* quantile of the normal distribution?

```r
# What is the Z-score of the 50th quantile of the normal distribution?
qnorm(.5)
```

```
## [1] 0
```

```r
# What is the Z-score of the 96th quantile of the normal distribution?
qnorm(.96)
```

```
## [1] 1.750686
```

```r
# What is the Z-score of the 99th quantile of the normal distribution?
qnorm(.99)
```

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

```r
# They're truly inverses!
pnorm(qnorm(0))
```

```
## [1] 0
```

```r
qnorm(pnorm(0))
```

```
## [1] 0
```

Let's plot `qnorm` and `pnorm` next to each other to further illustrate the fact they they are inverses.

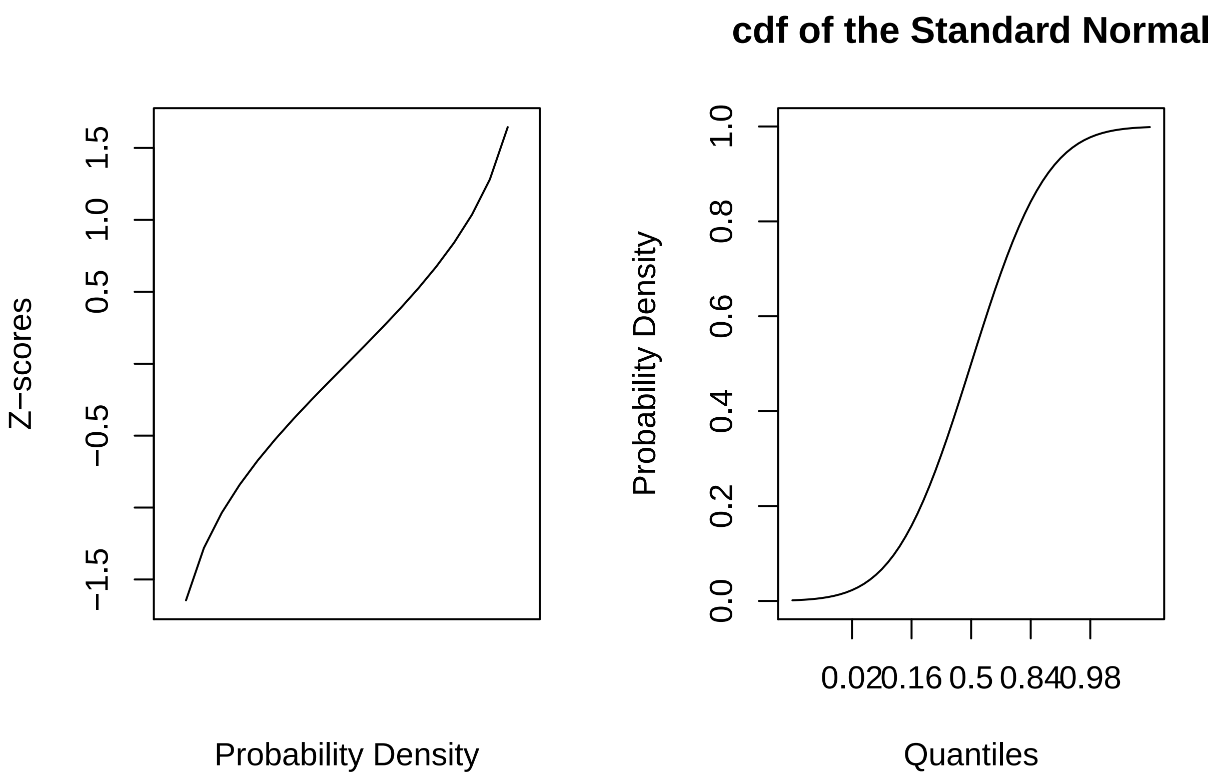
```r
# This is for getting two graphs next to each other
oldpar <- par()
par(mfrow=c(1,2))

# Let's make a vector of quantiles: from 0 to 1 by increments of .05
quantiles <- seq(0, 1, by = .05)
quantiles
```

```
##  [1] 0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70
## [16] 0.75 0.80 0.85 0.90 0.95 1.00
```

```r
# Now we'll find the Z-score at each quantile
qvalues <- qnorm(quantiles)
qvalues
```

```
##  [1]       -Inf -1.6448536 -1.2815516 -1.0364334 -0.8416212 -0.6744898
##  [7] -0.5244005 -0.3853205 -0.2533471 -0.1256613  0.0000000  0.1256613
## [13]  0.2533471  0.3853205  0.5244005  0.6744898  0.8416212  1.0364334
## [19]  1.2815516  1.6448536        Inf
```

```r
# Plot the z_scores
plot(qvalues,
     type = "l", # We want a line graph
     xaxt = "n", # No x-axis
     xlab="Probability Density",
     ylab="Z-scores")

# Same pnorm plot from before
plot(pvalues, # Plot where y = values and x = index of the value in the vector
     xaxt = "n", # Don't label the x-axis
     type = "l", # Make it a line plot
     main = "cdf of the Standard Normal",
     xlab= "Quantiles",
     ylab="Probability Density")

# These commands label the x-axis
axis(1, at=which(pvalues == pnorm(-2)), labels=round(pnorm(-2), 2))
axis(1, at=which(pvalues == pnorm(-1)), labels=round(pnorm(-1), 2))
axis(1, at=which(pvalues == pnorm(0)), labels=c(.5))
axis(1, at=which(pvalues == pnorm(1)), labels=round(pnorm(1), 2))
axis(1, at=which(pvalues == pnorm(2)), labels=round(pnorm(2), 2))
```



```
# Restore old plotting settings
par(oldpar)
```

`rnorm`

If you want to generate a vector of normally distributed random numbers, `rnorm` is the function you should use. The first argument `n` is the number of numbers you want to generate, followed by the standard `mean` and `sd` arguments. Let's illustrate the weak law of large numbers using `rnorm`.

```
# set.seed is a function that takes a number as an argument and sets a seed from
# which random numbers are generated. It's important to set a seed so that your
# code is reproduceable. If you wanted to you could always set your seed to the
# same number. I like to set seeds to the "date" which is really just
# the arithmetic equation "month minus day minus year". So today's seed
# is -2006.
set.seed(10-1-2015)
rnorm(5)
```

```
## [1] -0.7197035 -1.4442137 -1.0120381  1.4577066 -0.1212466
```

```r
# If I set the seed to the same seed again, I'll generate the same vector of
# numbers.
set.seed(10-1-2015)
rnorm(5)
```

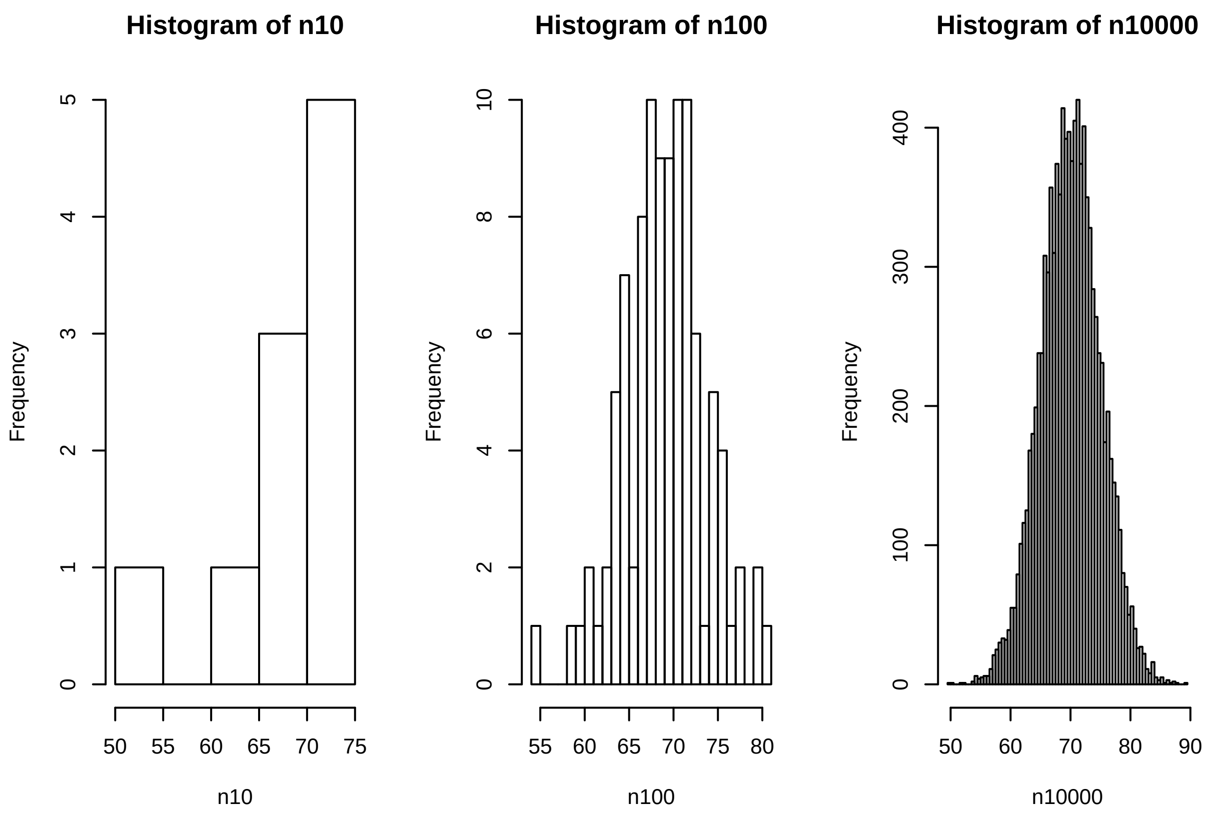
```
## [1] -0.7197035 -1.4442137 -1.0120381  1.4577066 -0.1212466
```

```r
# Now onto using rnorm

# Let's generate three different vectors of random numbers from a normal
# distribution
n10 <- rnorm(10, mean = 70, sd = 5)
n100 <- rnorm(100, mean = 70, sd = 5)
n10000 <-  rnorm(10000, mean = 70, sd = 5)

# Let's just look at one of the vectors
n10
```

```
##  [1] 54.70832 72.89000 70.27049 69.16508 72.97937 67.91004 67.77183 72.29231
##  [9] 74.33411 63.57151
```

Which historgram do you think will be most centered around the true mean of 70?

```r
# This is for getting two graphs next to each other
oldpar <- par()
par(mfrow=c(1,3))

# The breaks argument specifies how many bars are in the histogram
hist(n10, breaks = 5)
hist(n100, breaks = 20)
hist(n10000, breaks = 100)
```



```
# Restore old plotting settings
par(oldpar)
```

Closing thoughts

These concepts generally hold true for all the distribution functions built into R. You can learn more about all of the distribution functions by typing `help(Distributions)` into the R console. If you have any questions about this demonstration or about R programming please send me an email. If you'd like to change or contribute to this document I welcome pull requests on GitHub. This document and all code contained within is licensed CC0.