Introduction to dnorm, pnorm, qnorm, and rnorm

Contents

R has a family of built-in functions for a large number of common probability distributions. You can find a list of these functions by typing `help(Distributions)` at the R console.

To clarify how these work, this document provides illustrative examples focusing on the normal distribution. We will call these the *norm* family of functions. These are:

- **rnorm**: take a random sample from a normal distribution
- **dnorm**: find the height of the probability density function at one or more values of x
- **pnorm**: find the area under the PDF for a certain interval of x values; this is the CDF
- **qnorm**: find the value of x corresponding to the pth quantile (percentile); this is the inverse CDF

---

dnorm

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

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

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:

```
# The dnorm function takes three main arguments, as do all of the *norm functions in R.
# The following computes the PDF of the normal with x = 0, mu = 0 and sigma = 0.
dnorm(x = 0, mean = 0, sd = 1)
```

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

```
# Since mean = 0 and sd = 1 are the default arguments, the following is equivalent:
dnorm(0)
```

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

```
# Another exmaple of dnorm where parameters have been changed.
dnorm(2, mean = 5, sd = 3)
```

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

```
# Where not explicitly specified, arguments are expected in the same order (careful!)
dnorm(2,5,3)
```

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

Although x represents the independent variable of the PDF for the normal distribution, it can sometimes be useful to think of x as a Z-score. Let me show you what I mean by graphing the PDF of the standard normal distribution with `dnorm`.

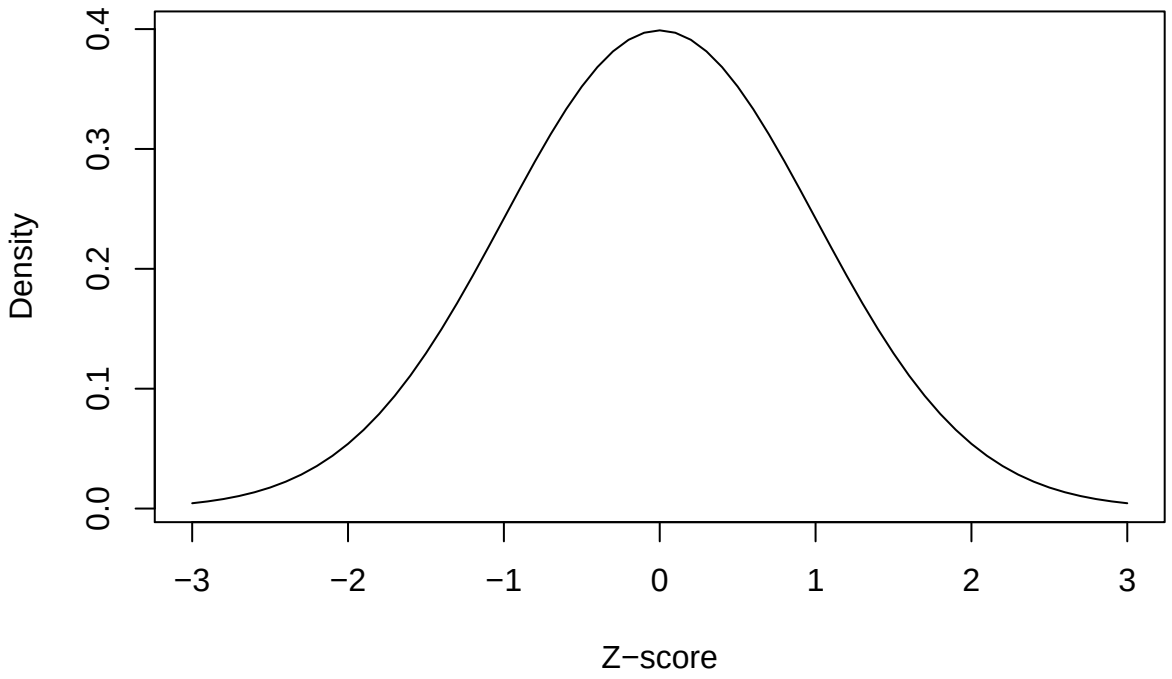
```
# First I'll make a vector of Z-scores
z_scores <- seq(-3, 3, by = .1)
# Let's print the vector
z_scores
```

```
##  [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
## [15] -1.6 -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
## [29] -0.2 -0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1
## [43]  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5
## [57]  2.6  2.7  2.8  2.9  3.0
```

```
# 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
##  [6] 0.017528300 0.022394530 0.028327038 0.035474593 0.043983596
## [11] 0.053990967 0.065615815 0.078950158 0.094049077 0.110920835
## [16] 0.129517596 0.149727466 0.171368592 0.194186055 0.217852177
## [21] 0.241970725 0.266085250 0.289691553 0.312253933 0.333224603
## [26] 0.352065327 0.368270140 0.381387815 0.391042694 0.396952547
## [31] 0.398942280 0.396952547 0.391042694 0.381387815 0.368270140
## [36] 0.352065327 0.333224603 0.312253933 0.289691553 0.266085250
## [41] 0.241970725 0.217852177 0.194186055 0.171368592 0.149727466
## [46] 0.129517596 0.110920835 0.094049077 0.078950158 0.065615815
## [51] 0.053990967 0.043983596 0.035474593 0.028327038 0.022394530
## [56] 0.017528300 0.013582969 0.010420935 0.007915452 0.005952532
## [61] 0.004431848
```

```
# Now we'll plot these values
plot(z_scores, # if omitted, the vector indices for dvalues are used instead
     dvalues,
     xlab= "Z-score",
     ylab="Density",
#     xaxt = "n", # Don't label the x-axis tick marks
     type = "l", # Make it a line plot
     main = "PDF of the Standard Normal")
```




```
# If you want to add your own labels, you can uncomment these and the 'xaxt' line above
# 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 standard 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. `pnorm` has the same default `mean` and `sd` arguments as `dnorm`. Try to guess the value of `pnorm(0)`.

```
# Arguments: q = 0, mean = 0, sd = 1
pnorm(0)  # same as pnorm(q=0, mean=0, sd=1)
```

```
## [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)                    # lower tail for the standard normal at Z-score=2
```

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

```
pnorm(2, mean = 5, sd = 3)  # lt for a broader distribution with mean shifted to 5
```

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

```r
# These are equivalent expressions for the probability that P(X > x)
# The upper-tail p value 1-pnorm(x) = rejection area
pnorm(2, mean = 5, sd = 3, lower.tail = FALSE)     # the upper tail
```

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

```r
1 - pnorm(2,5,3)                                   # also the upper tail
```

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

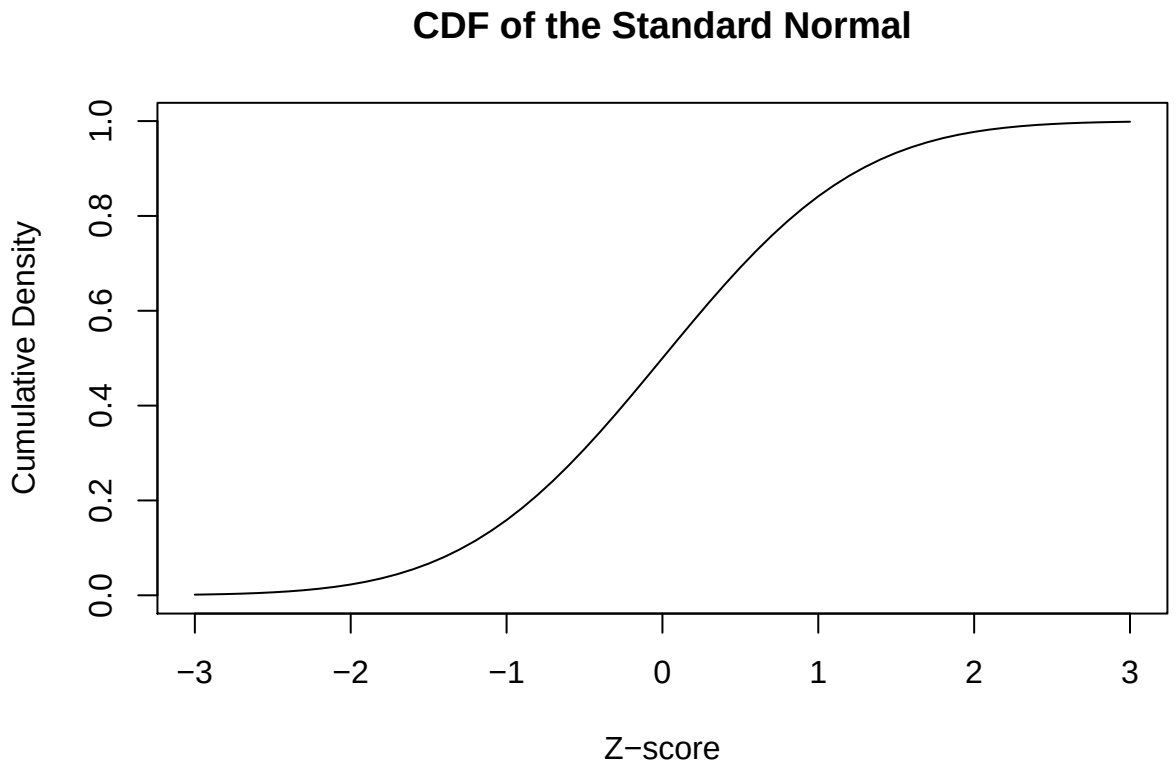
```r
#  The non-rejection region where P(X <= x)
1 - pnorm(2, mean = 5, sd = 3, lower.tail = FALSE) # back to the lower tail!
```

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

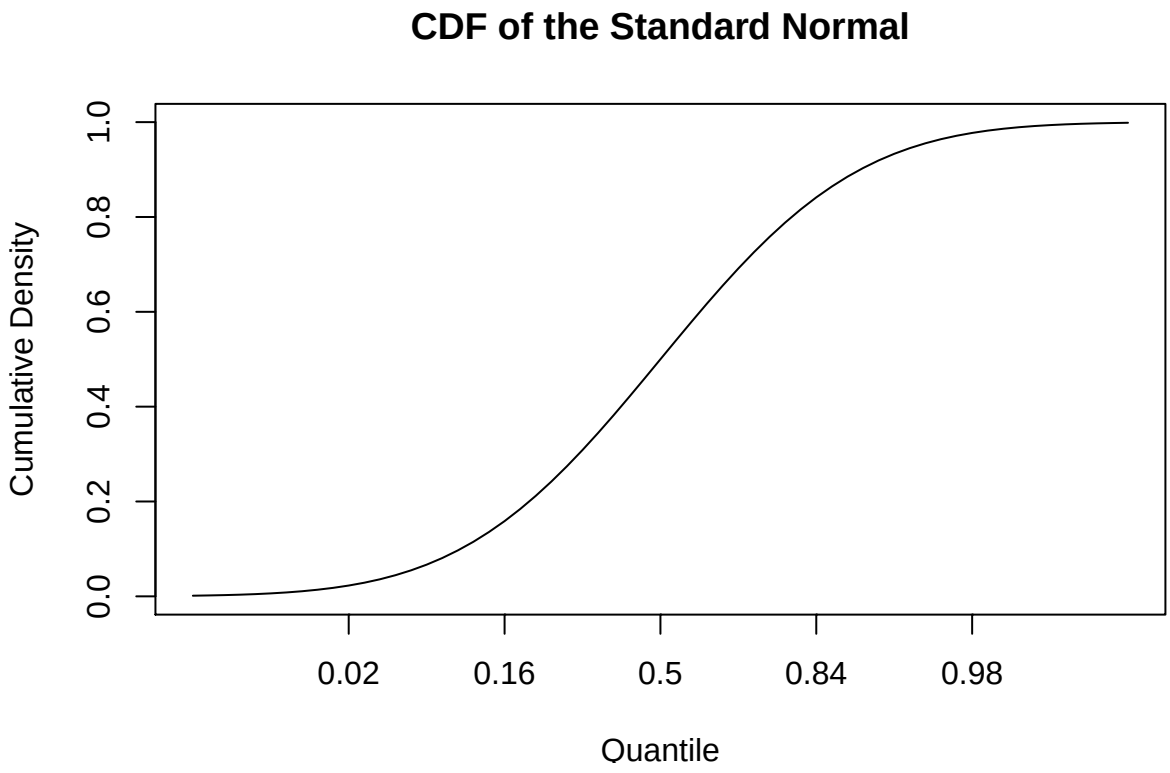
pnorm is the function that replaces the table of probabilites and Z-scores at the back of the statistics textbook qnd can be used to get p-values.

Let's take our vector of Z-scores from before (z_scores) and compute a new vector of "probability masses" using pnorm. A plot of these shows the cumulative distribution function of the normal distribution.

```r
z_scores <- seq(-3, 3, by = .1)
pvalues <- pnorm(z_scores)

# Now we'll make two different plots ... notice the different labels on the x-axis.

# This plot shows the **Z-scores** vs. the cumulative density
plot(z_scores, pvalues,
     type = "l", # Make it a line plot
     main = "CDF of the Standard Normal",
     xlab= "Z-score",
     ylab="Cumulative Density")
```



```r
# This plot shows the **quantiles** vs. the cumulative density
plot(pvalues,      # x is omitted, so y = values and x = vector indeces
     xaxt = "n",  # Don't label ticks on the x-axis
     type = "l",  # Make it a line plot
     main = "CDF of the Standard Normal",
     xlab= "Quantile",
     ylab="Cumulative 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))
```



---

qnorm

The `qnorm` function is simply the inverse of the CDF, i.e. it is 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
```

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

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

```
# around 2/3 (68% of the density falls between -1 and +1 sd of the mean
pnorm(1) - pnorm(-1)
```

```
## [1] 0.6826895
```

```
# around 95% of the density falls between -2 and +2 sd of the mean
pnorm(2) - pnorm(-2)
```

```
## [1] 0.9544997
```

```
# the central 95% quantile range spans roughly 4 sd (-2 to +2)
qnorm(0.975) - qnorm(0.025)
```

```
## [1] 3.919928
```

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

```
# Make a vector of Z-scores: from -3 to 3 by 0.1 # get the CDF for the standard normal
z_scores <- seq(-3, 3, by = .1)
pvalues <- pnorm(z_scores)

# Make a vector of quantiles: from 0 to 1 by increments of .05
# distribution is infinite so don't try to hit limits of probability
quantiles <- seq(0.001, 0.999, by = .01)
quantiles
```

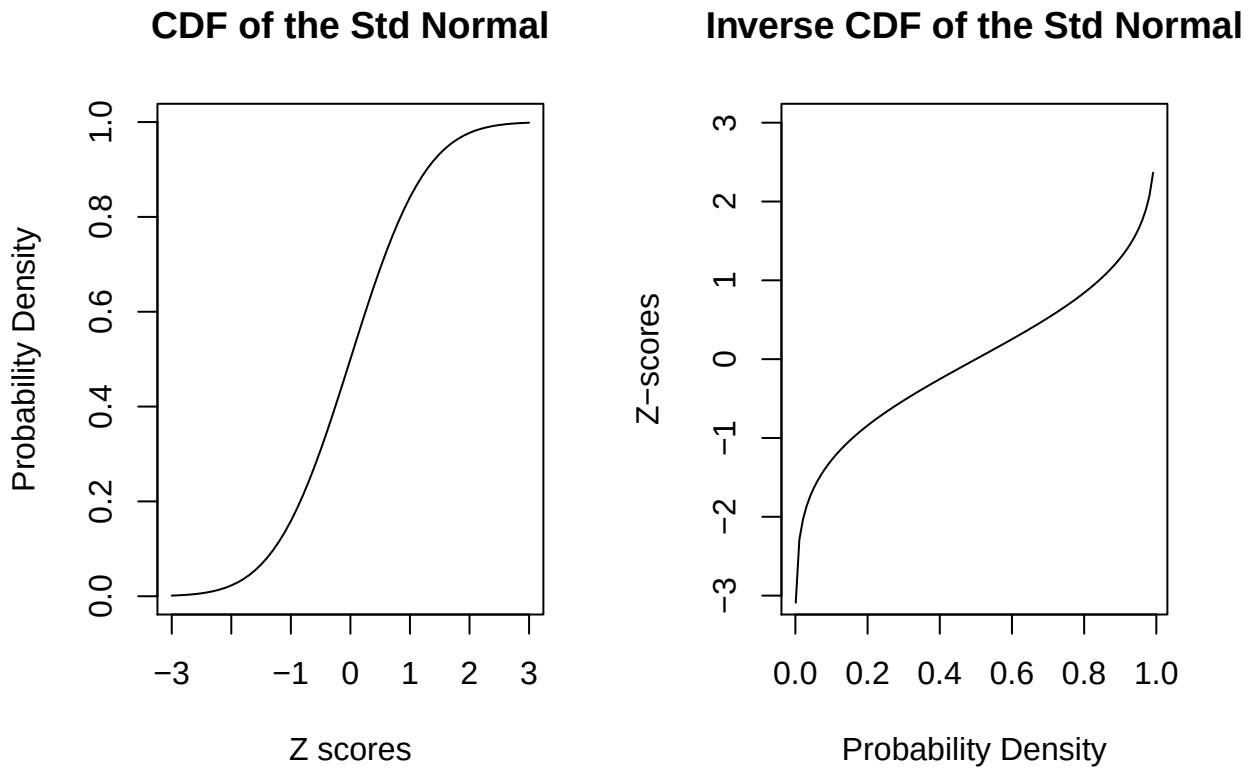
```
##   [1] 0.001 0.011 0.021 0.031 0.041 0.051 0.061 0.071 0.081 0.091 0.101
##  [12] 0.111 0.121 0.131 0.141 0.151 0.161 0.171 0.181 0.191 0.201 0.211
##  [23] 0.221 0.231 0.241 0.251 0.261 0.271 0.281 0.291 0.301 0.311 0.321
##  [34] 0.331 0.341 0.351 0.361 0.371 0.381 0.391 0.401 0.411 0.421 0.431
##  [45] 0.441 0.451 0.461 0.471 0.481 0.491 0.501 0.511 0.521 0.531 0.541
##  [56] 0.551 0.561 0.571 0.581 0.591 0.601 0.611 0.621 0.631 0.641 0.651
##  [67] 0.661 0.671 0.681 0.691 0.701 0.711 0.721 0.731 0.741 0.751 0.761
##  [78] 0.771 0.781 0.791 0.801 0.811 0.821 0.831 0.841 0.851 0.861 0.871
##  [89] 0.881 0.891 0.901 0.911 0.921 0.931 0.941 0.951 0.961 0.971 0.981
## [100] 0.991
```

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

```
##   [1] -3.090232306 -2.290367878 -2.033520149 -1.866295743 -1.739197665
##   [6] -1.635234015 -1.546433122 -1.468383798 -1.398376621 -1.334622287
##  [11] -1.275874179 -1.221227222 -1.170002408 -1.121676528 -1.075837361
##  [16] -1.032153958 -0.990356294 -0.950220942 -0.911560735 -0.874217165
##  [21] -0.838054670 -0.802956288 -0.768820293 -0.735557557 -0.703089460
##  [26] -0.671346215 -0.640265509 -0.609791399 -0.579873392 -0.550465695
##  [31] -0.521526572 -0.493017814 -0.464904288 -0.437153541 -0.409735480
##  [36] -0.382622075 -0.355787114 -0.329205984 -0.302855481 -0.276713637
##  [41] -0.250759572 -0.224973358 -0.199335898 -0.173828813 -0.148434341
##  [46] -0.123135248 -0.097914734 -0.072756358 -0.047643956 -0.022561568
##  [51]  0.002506631  0.027576406  0.052663527  0.077783842  0.102953344
```

```
##  [56]  0.128188248  0.153505060  0.178920660  0.204452382  0.230118101
##  [61]  0.255936332  0.281926330  0.308108202  0.334503036  0.361133034
##  [66]  0.388021666  0.415193851  0.442676144  0.470496968  0.498686864
##  [71]  0.527278791  0.556308467  0.585814766  0.615840189  0.646431416
##  [76]  0.677639965  0.709522974  0.742144154  0.775574943  0.809895915
##  [81]  0.845198535  0.881587347  0.919182735  0.958124465  0.998576271
##  [86]  1.040731886  1.084823128  1.131130901  1.180000540  1.231863709
##  [91]  1.287270563  1.346938626  1.411830078  1.483280127  1.563223647
##  [96]  1.654627902  1.762410298  1.895697924  2.074854734  2.365618127
```

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

# Same pnorm plot from before
plot(z_scores, pvalues, type = "l",
     main = "CDF of the Std Normal",
     xlab= "Z scores",
     ylab="Probability Density")
# Plot the quantiles
plot(quantiles, qvalues, ylim = c(-3,3),
     type = "l",
     main = "Inverse CDF of the Std Normal",
     xlab="Probability Density",
     ylab="Z-scores")
```



These plots show that qnorm (inverse CDF) really is the inverse function of pnorm (CDF). This can also be appreciated by applying the functions sequentially:

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

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

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

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

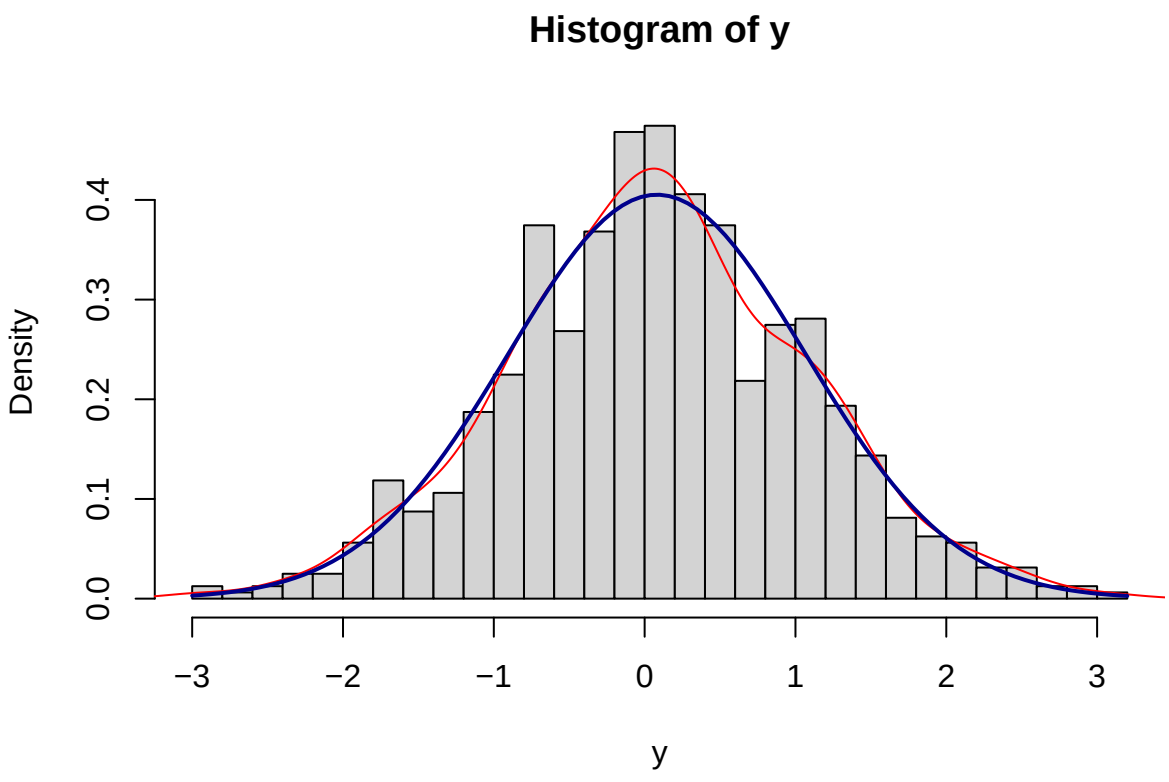
---

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.

You can plot a histogram of the samples along with the empirical density of the sample along with the theoretical curve:

```
par(mfrow=c(1,1), mar=c(5.1,4.1,4.1,2.1)) # making the plot window to default size
xseq<-seq(-4,4,.01)
y<-rnorm(length(xseq))
hist(y, prob=TRUE, breaks=40,col="lightgray")
# add density for the sample
sample_density <- density(y)
lines(sample_density, col="red")
# add curve() function to the plot using the mean and SD of y as the parameters:
curve(dnorm(x, mean(y), sd(y)), add=TRUE, col="darkblue", lwd=2)
```



The ***weak law of large numbers***, or *Bernoulli's theorem*, states that the sampling distribution of the mean converges on the true mean as the number of samples goes to infinity. Let's illustrate this using `rnorm`.

```
# The set.seed function 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.
```

```r
# Some people like to set seeds to the "date", which is just the arithmetic equation
# "month minus day minus year". The following evaluates to -2028:
set.seed(9-18-2019)  # it's my birthday!
rnorm(5)
```

```
## [1]  0.6850690  1.4422466 -0.6676201  1.3808275  0.6520382
```

```r
# Using the same seed again, I'll generate the same vector of numbers.
set.seed(9-18-2019)
rnorm(5)
```

```
## [1]  0.6850690  1.4422466 -0.6676201  1.3808275  0.6520382
```

```r
# Setting it to somethinge else gives a different random sample
set.seed(Sys.time())  # this date in the current time zone is coerced into an integer
rnorm(5)
```

```
## [1]  0.9152968 -0.9898384 -1.3963791 -0.9374331  0.2933826
```

```r
# The seed only sticks once!
rnorm(5)
```

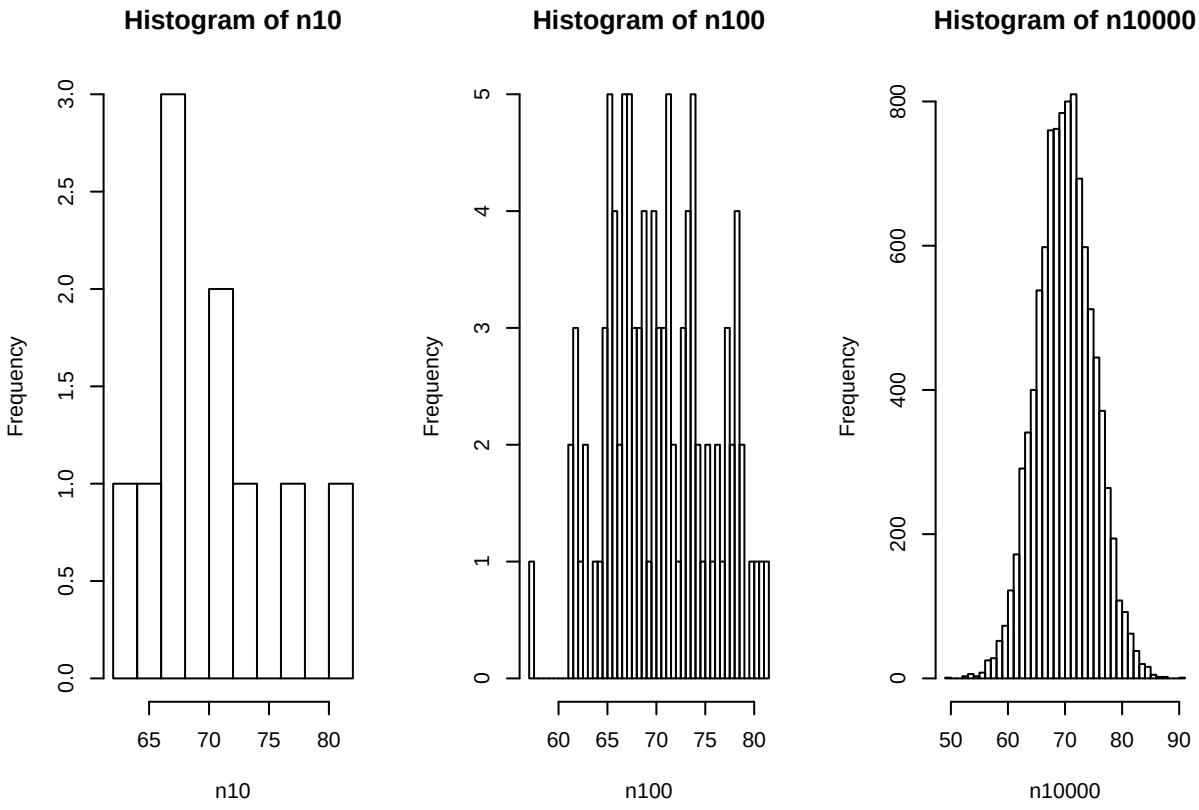
```
## [1]  0.9154941 -0.8949214 -1.0982440  1.0227845 -0.3881162
```

```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] 70.49996 64.62309 67.64532 62.14195 80.66368 73.95023 76.63410
##  [8] 71.00808 66.98691 67.43046
```

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

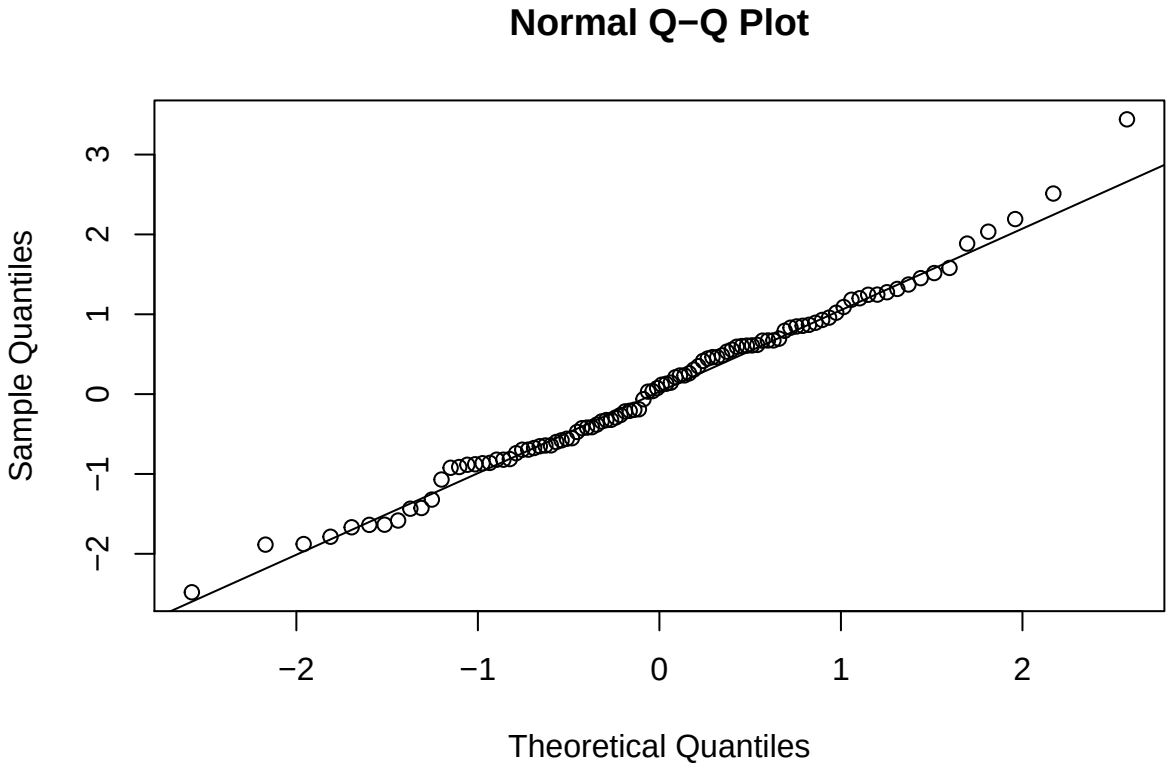
```r
# This is for getting two graphs next to each other
oldpar <- par()
par(mfrow=c(1,3))

# The breaks argument specifies how many bars are in the histogram
hist(n10, breaks = 10)
hist(n100, breaks = 50)
hist(n10000, breaks = 50)
```



QQ plots

You can compare a sample from a distribution to a theoretical distribution using a QQ plot.

```
y <- rnorm(100)
qqnorm(y)
qqline(y)
```

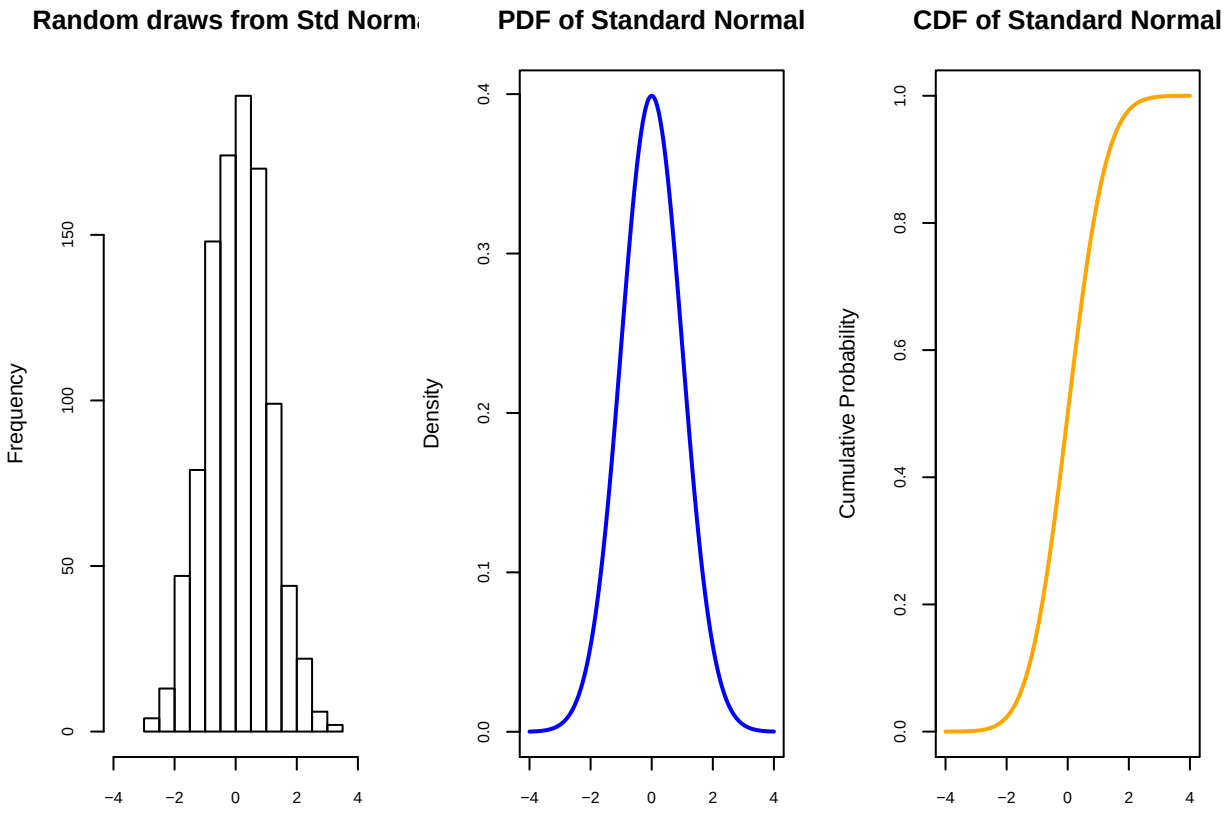


If a sample doesn't match the QQ line, then it's not very representative of the theoretical distribution. Even for a sample taken from the theoretical population, as above, the samples will usually deviate more from the line at the tails due to low sampling coverage.

Comparing the different functions

Now we can put all of these together:

```
set.seed(9-18-2019)
xseq<-seq(-4,4,.01)
densities<-dnorm(xseq, 0,1)
cumulative<-pnorm(xseq, 0, 1)
random.samples<-rnorm(1000,0,1)
par(mfrow=c(1,3), mar=c(3,4,4,2))
hist(random.samples, main="Random draws from Std Normal",
     cex.axis=.8, xlim=c(-4,4))
plot(xseq, densities, main="PDF of Standard Normal",
     col="blue", xlab="", ylab="Density",
     type="l", lwd=2, cex=2, cex.axis=.8)
plot(xseq, cumulative, main="CDF of Standard Normal",
     col="orange", xlab="", ylab="Cumulative Probability",
     type="l", lwd=2, cex=2, cex.axis=.8)
```



Sampling from distributions vs. using the sample function

When we use functions like `rnorm`, we sample much more often from regions where the distribution is more dense. For the normal distribution, this means that we get many more samples that are close to the mean than those that are far away from it.

In contrast, when we take a random sample from a set of numbers using the sample function, we are taking values from a uniform distribution across the range of values. So these two procedures produce very different results.

Other distributions

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.
