A Guide to dnorm, pnorm, qnorm, and rnorm in R

Authored by

Mohammed looti

November 9, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *A Guide to dnorm, pnorm, qnorm, and rnorm in R*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=14349

The Normal Distribution, often referred to as the Gaussian distribution, is arguably the most fundamental and widely utilized distribution in the field of statistics. Its familiar bell-shaped curve describes countless natural phenomena and serves as the backbone for much of modern statistical inference. This comprehensive tutorial is designed to explain how to effectively work with the **Normal Distribution** within the statistical programming environment R, specifically focusing on the core distributional functions: **dnorm**, **pnorm**, **qnorm**, and **rnorm**. Mastering these four functions is essential for anyone performing statistical analysis, simulation, or modeling in R.

dnorm: Calculating the Probability Density Function (PDF)

The **dnorm** function is crucial for determining the height of the probability distribution at a specific point. Technically, **dnorm** returns the value of the Probability Density Function (PDF) for the **Normal Distribution**, given a specific value of the random variable x, along with the population mean (mu) and the population standard deviation ($sigma$). While the PDF value itself is not a direct probability (since the probability of a continuous variable equaling an exact point is zero), it is indispensable for visualizing the shape of the distribution and performing advanced calculations like maximum likelihood estimation.

The fundamental syntax for utilizing **dnorm** is highly intuitive, requiring the input value x, the specified mean, and the standard deviation. If the mean and standard deviation are omitted, R defaults to using the parameters of the **Standard Normal Distribution** (mean=0, sd=1).

dnorm(x, mean, sd)

The following examples demonstrate how **dnorm** calculates the density for various scenarios, including the peak density point ($x=0$) for the standard normal curve:

```
# Find the value of the standard normal distribution PDF at x=0 (the mean)
dnorm(x=0, mean=0, sd=1)
# 0.3989423

# By default, R uses mean=0 and sd=1 if parameters are omitted
dnorm(x=0)
# 0.3989423

# Find the value of the normal distribution PDF at x=10 with mean=20 and sd=5.
# Note the much lower density since x=10 is 2 standard deviations below the mean.
dnorm(x=10, mean=20, sd=5)
# 0.01079819
```

While **dnorm** is less frequently used for solving direct probability questions--a task usually handled

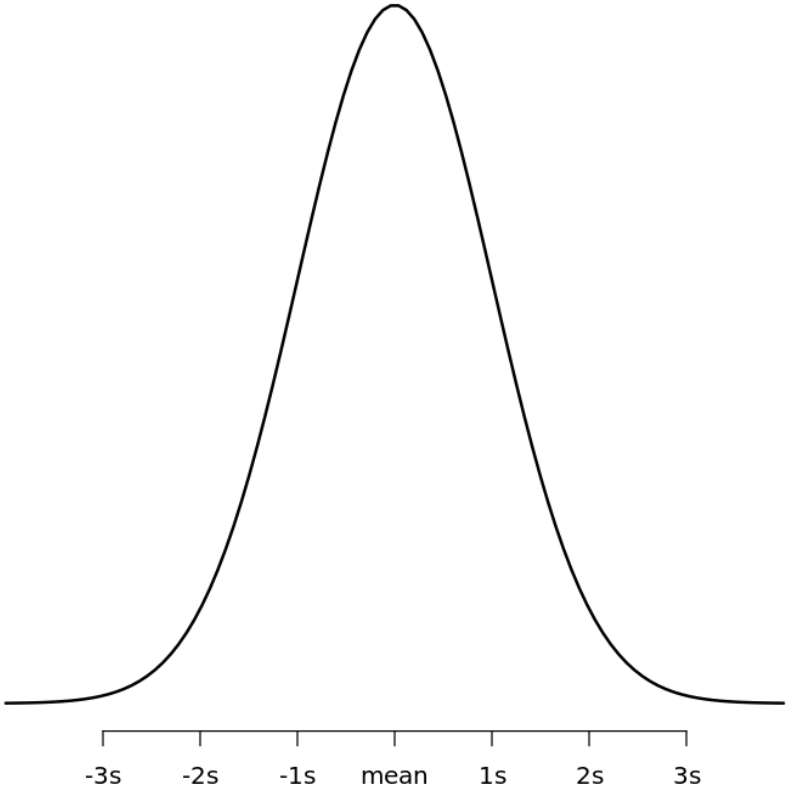
by **pnorm**--its main practical utility lies in visualization. It allows us to generate the characteristic bell curve shape by calculating density values across a range of x-inputs. The subsequent code block illustrates how to leverage **dnorm** to construct a high-quality plot of the **Normal Distribution** in R:

```
# Create a sequence of 100 equally spaced numbers between -4 and 4, covering most of the distribution
x <- seq(-4, 4, length=100)

# Create a vector (y) of corresponding density values for each x using dnorm
y <- dnorm(x)

# Plot x and y as a scatterplot with connected lines (type = "l") and customize the axes for clarity.
plot(x,y, type = "l", lwd = 2, axes = FALSE, xlab = "", ylab = "")
axis(1, at = -3:3, labels = c("-3s", "-2s", "-1s", "mean", "1s", "2s", "3s"))
```

Executing this visualization code generates the standard bell curve plot, where the density peaks precisely at the mean:



pnorm: Finding Cumulative Probabilities (CDF)

In contrast to **dnorm**, which provides the density at a single point, the **pnorm** function is designed to calculate actual probabilities. Specifically, **pnorm** returns the value of the Cumulative Distribution Function (CDF) for the **Normal Distribution**. Conceptually, the CDF gives the area under the PDF curve up to a specified value q. This area directly represents the probability that a randomly chosen observation will be less than or equal to q.

The primary syntax for **pnorm** is similar to **dnorm**, requiring the quantile q (the value of interest), the mean (mu), and the standard deviation ($sigma$). By default, **pnorm** calculates the probability for the left tail (i.e., $P(X le q)$).

pnorm(q, mean, sd)

A powerful feature of **pnorm** is the optional argument **lower.tail**. If you are interested in the probability that a value is greater than q (the area to the right of q), setting **lower.tail =**

FALSE allows R to calculate the right-tail probability directly, avoiding the need for manual subtraction from 1.

pnorm(q, mean, sd, lower.tail = FALSE)

We can apply **pnorm** to solve various real-world probability problems involving normally distributed data, as illustrated in the following scenarios:

Example 1: Right-Tail Probability (Taller than q)

*Consider the height of males at a school, which is normally distributed with a mean (mu) of 70 inches and a standard deviation ($sigma$) of 2 inches. We aim to find the percentage of males taller than 74 inches. Because we are looking for the probability above the value (the right tail), we use the **lower.tail = FALSE** argument.*

 # Find the percentage of males that are taller than 74 inches (right tail)
pnorm(74, mean=70, sd=2, lower.tail=FALSE)

0.02275013

The result indicates that approximately 2.275% of males at this school exceed 74 inches in height.

Example 2: Left-Tail Probability (Less than q)

*Suppose the weight of a species of otters is normally distributed with $mu = 30$ lbs and $sigma = 5$ lbs. We want to determine the percentage of otters weighing less than 22 lbs. Since we seek the probability below the value (the left tail), the default **pnorm** behavior is used.*

 # Find percentage of otters that weigh less than 22 lbs (left tail)
pnorm(22, mean=30, sd=5)

0.05479929

This calculation reveals that approximately 5.48% of this otter species weigh less than 22 lbs.

Example 3: Probability Between Two Values

If the height of plants in a region is normally distributed with $mu = 13$ inches and $sigma = 2$ inches, how do we find the percentage of plants between 10 and 14 inches tall? To find the area between two points (q_1 and q_2), we calculate the CDF for the upper bound ($P(X le q_2)$) and subtract the CDF for the lower bound ($P(X le q_1)$).

```
# Calculate P(X <= 14) and subtract P(X <= 10)
pnorm(14, mean=13, sd=2) - pnorm(10, mean=13, sd=2)

# 0.6246553
```

The analysis shows that approximately 62.47% of the plants in this region fall within the height range of 10 to 14 inches.

qnorm: Determining Quantiles (Inverse CDF)

The **qnorm** function operates as the inverse of **pnorm**. Instead of taking a value q and returning a probability p (area), **qnorm** takes a probability p (the area to the left) and returns the corresponding quantile value q on the x-axis. This process is formally known as calculating the value of the inverse Cumulative Distribution Function (CDF).

In practical statistical analysis, **qnorm** is primarily used to find critical values or cutoff points. For instance, if you want to identify the point below which 95% of the data falls, **qnorm** provides that exact boundary value. When **qnorm** is used with the **Standard Normal Distribution** (mean=0, sd=1), the output is the corresponding Z-score for that given probability.

qnorm(p, mean, sd)

The following examples illustrate how **qnorm** is used to locate specific quantiles within the **Normal Distribution**, providing crucial boundary values for hypothesis testing and confidence interval construction:

```
# Find the Z-score (the x-value) of the 99th quantile of the standard normal distribution (P=0.99)
qnorm(.99, mean=0, sd=1)
# 2.326348

# R defaults to mean=0 and sd=1 for the standard normal curve
qnorm(.99)
# 2.326348

# Find the Z-score corresponding to the 95th percentile (a common critical value)
qnorm(.95)
# 1.644854

# Find the Z-score of the 10th quantile (P=0.10, indicating a value far below the mean)
qnorm(.10)
# -1.281552
```

rnorm: Generating Random Normal Variables

The **rnorm** function is indispensable for statistical simulation, modeling, and bootstrapping techniques. Its purpose is to generate a specified number (n) of pseudo-random observations that follow a **Normal Distribution** characterized by a given mean (mu) and standard deviation ($sigma$). This capability allows researchers to create synthetic datasets that mirror real-world variability or to test the robustness of statistical models.

The syntax for **rnorm** requires the user to specify the number of random samples needed (n), followed by the desired population parameters (mean and standard deviation). If the mean and standard deviation are not explicitly provided, R defaults to generating random numbers from the **Standard Normal Distribution** (mean=0, sd=1).

rnorm(n, mean, sd)

The following examples demonstrate how to use **rnorm** to create vectors of random data, highlighting how varying the standard deviation impacts the generated data's spread:

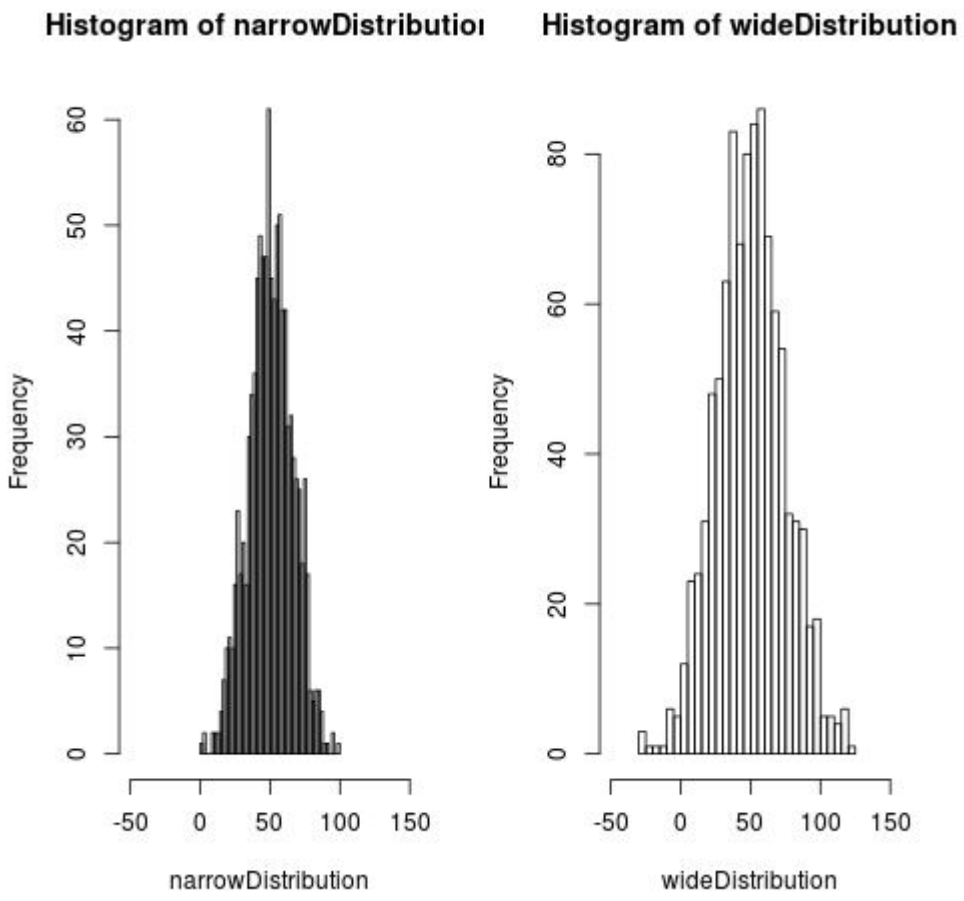
 # Generate a vector of 5 normally distributed random variables with mean=10 and sd=2
five <- rnorm(5, mean = 10, sd = 2)
five
 # 10.658117 8.613495 10.561760 11.123492 10.802768

# Generate a large sample (N=1000) with a relatively small standard deviation (SD=15)
narrowDistribution <- rnorm(1000, mean = 50, sd = 15)

# Generate a large sample (N=1000) with a large standard deviation (SD=25)
wideDistribution <- rnorm(1000, mean = 50, sd = 25)

Generate two histograms side-by-side to visually compare the effect of different standard deviations.
par(mfrow=c(1, 2)) # Set graphical parameter for one row, two columns
hist(narrowDistribution, breaks=50, xlim=c(-50, 150))
hist(wideDistribution, breaks=50, xlim=c(-50, 150))

Visualizing these two simulated datasets clearly demonstrates the impact of the standard deviation parameter:



As shown above, both synthetic distributions are centered around the specified mean of 50. However, the distribution generated with a standard deviation of 25 (the wide distribution) exhibits significantly greater spread and variability compared to the distribution generated with an SD of 15 (the narrow distribution). This highlights how **rnorm** faithfully incorporates the supplied parameters to model different levels of dispersion.

Summary of R's Normal Distribution Functions

The suite of distributional functions in R provides a powerful and consistent framework for working with the **Normal Distribution**. Whether you are generating data, calculating probabilities, finding critical values, or visualizing the density curve, these four functions--**dnorm**, **pnorm**, **qnorm**, and **rnorm**--cover nearly every scenario encountered in statistical practice.

To reinforce the primary distinctions between these commands, here is a quick reference guide detailing their inputs and outputs:

dnorm(x, mean, sd): Calculates the height of the curve (the **Probability Density Function**) at a specific point x. Output: Density.

pnorm(q, mean, sd): Calculates the cumulative area under the curve (the probability) up to a point q. Output: Probability/Area.
qnorm(p, mean, sd): Calculates the quantile value (the x-value or critical value) corresponding to a given cumulative probability p. Output: Quantile/Cutoff Value.
rnorm(n, mean, sd): Generates n number of random observations that adhere to the specified normal parameters. Output: Vector of Random Numbers.

By understanding these relationships, users of R can confidently manipulate and interpret data that follows the ubiquitous bell-shaped curve, paving the way for more complex statistical modeling and inference.