Learning the Bivariate Normal Distribution: Simulation and Plotting in R

Authored by

Mohammed looti

October 30, 2025

In modern **statistics** and advanced **data analysis**, the ability to model and interpret the joint behavior of multiple variables is fundamentally important. When dealing specifically with two continuous variables that exhibit a Gaussian joint behavior, the **bivariate normal distribution** (BND) stands out as a foundational concept. This distribution rigorously defines the joint probability of two random variables, ensuring that each variable, when considered individually, follows a standard **normal distribution**, and crucially, that any linear combination of these two variables also results in a normal distribution.

The BND is not merely a theoretical construct; its practical applications span diverse fields, ranging from modeling risk in finance and predicting biological traits to understanding measurement errors in engineering. By simulating such distributions, researchers and analysts gain the critical capability to generate realistic, synthetic datasets. These simulated data are essential for testing complex statistical hypotheses, deeply understanding specific statistical properties, and developing robust new analytical methodologies before applying them to costly or limited real-world data. Furthermore, effective visualization of these complex structures provides essential, intuitive insights into the distribution's core characteristics, including its central location, overall spread, and the inherent correlation structure connecting the two variables.

This expert tutorial is designed to provide a comprehensive, step-by-step methodology for both simulating and graphically visualizing a **bivariate normal distribution** using the highly acclaimed statistical programming environment, **R**. We will meticulously cover the necessary steps required to move from theoretical parameters to practical, visual representations. The key learning objectives we will achieve include:

Simulate a **bivariate normal distribution**, producing a reliable dataset that accurately reflects the defined statistical properties of the distribution.
Generate a 2-dimensional representation of the probability density using a **contour plot**, which visualizes lines of equal density.
Construct a high-impact 3-dimensional visualization using a **surface plot**, offering a complete view of the distribution's shape and height.

Defining the Parameters of Bivariate Normality

Before we can successfully dive into the practical implementation of simulation and plotting within **R**, it is absolutely vital to fully grasp the fundamental components that mathematically define a **bivariate normal distribution**. Unlike its simpler univariate counterpart, which is solely defined by a single mean and variance, the BND requires two specific, essential parameters to be fully characterized: the **mean vector** and the **covariance matrix**. These two elements collectively dictate the position, shape, and orientation of the resulting bell-shaped surface in 3D space.

The **mean vector**, conventionally symbolized as μ (mu), is structured as a 2x1 column vector. This

vector contains the arithmetic means of the two individual variables, typically labeled X and Y. Mathematically, this is expressed as μ = T. These values are incredibly significant because they precisely locate the center of the distribution in the two-dimensional coordinate system. Consequently, the maximum peak of the distribution's probability density function will always be centered exactly at these defined mean values, representing the most likely simultaneous outcome of X and Y.

The second defining parameter is the **covariance matrix**, denoted by Σ (Sigma). This critical parameter is a 2x2 symmetric matrix that encapsulates both the individual variances and the joint covariance between the variables. Its standard structure is represented as:

Σ = ,]

Within this structure, σX2 and σY2 are the variances of X and Y, respectively, indicating the spread along each axis. Crucially, the off-diagonal terms, σXY (which must equal σYX for a valid covariance matrix), define the covariance. The **covariance matrix** quantifies the linear dependency: a positive value shows a tendency for both variables to increase together, while a negative value implies an inverse relationship. If the covariance is zero, it signifies that there is no linear relationship between X and Y.

Example 1: Simulating Bivariate Data in R using MASS

The most efficient and widely accepted approach for generating random samples from a **bivariate normal distribution** within the **R** environment is through the use of the powerful `mvrnorm()` **function**. This function is housed within the widely utilized **MASS package**, which provides a robust and computationally sound method for generating samples from any multivariate normal distribution, including the specific bivariate case we are focusing on here.

To successfully execute the `mvrnorm()` **function**, three primary parameters must be accurately provided, as they define the statistical characteristics of the population from which the samples are to be drawn. Understanding these required arguments is key to successful simulation:

`n`: This specifies the required **sample size**, denoting the total number of random observations (rows) that the user wishes the simulation to generate from the defined distribution.
`mu`: This is the **mean** vector, representing the **mean vector** of the distribution. For bivariate simulation, it must be a numeric vector of length 2, containing the mean of the first variable (X) and the mean of the second variable (Y) in order.
`Sigma`: This defines the **covariance matrix**. It must be a 2x2 symmetric, positive semi-definite matrix. The values on the main diagonal are the variances, while the off-diagonal elements specify the covariance between the two variables.

The following R code block illustrates the practical application of the `mvrnorm()` **function**. Here, we simulate 100 observations using a specific mean vector and covariance matrix, ensuring reproducibility via the `set.seed()` function, and then inspect the resulting dataset structure using `head()`:

```
library(MASS)

#make this example reproducible
set.seed(0)

#simulate bivariate normal distribution
bivariate_data <- as.data.frame(mvrnorm(n=100,
mu=c(0, 0),
Sigma=matrix(c(5, 3, 4, 4), ncol=2)))

#view first six rows of bivariate dataset
head(bivariate_data)

V1 V2
1 -2.03600343 -2.9623059
2 0.07719131 1.2948982
3 -3.26729701 -1.7928069
4 -2.62985132 -2.3015471
5 -1.75126215 0.3056698
6 3.67698436 2.2020238
```

Upon execution, the resulting object `bivariate_data` is a **data frame** containing 100 observations, organized into two columns (V1 and V2). These data points are statistically sampled from a distribution centered at a **mean vector** of ``. The specific **covariance matrix** used here implies a variance of 5 for V1 and 4 for V2. The off-diagonal elements (3 and 4 in this non-symmetric input) define the correlation structure, which dictates the shape and tilt of the resulting data cloud. The `head()` output confirms the successful generation and structure of this simulated data, making it ready for further analysis or visualization.

Example 2: 2D Visualization using a Contour Plot

After successfully simulating or defining the parameters of a bivariate normal structure, visualizing its probability density function becomes the next critical step for comprehension. The **contour plot** is an exceptional tool for this purpose, providing an effective 2-dimensional map of the distribution. Conceptually, a contour plot operates much like a topographical map, where each line connects points in the (X, Y) plane that share an identical level of probability density. This visualization

method clearly illustrates the elliptical nature of the BND.

To generate this visualization in **R**, we utilize the `dmnorm()` **function** from the specialized **mnormt package**, which is dedicated to multivariate normal and t distributions. This function calculates the probability density values across a grid. We then pass this density grid to the base R `contour()` **function** to render the visualization. The preparation phase requires defining the coordinate range, setting the mean vector and covariance matrix, and generating the density matrix.

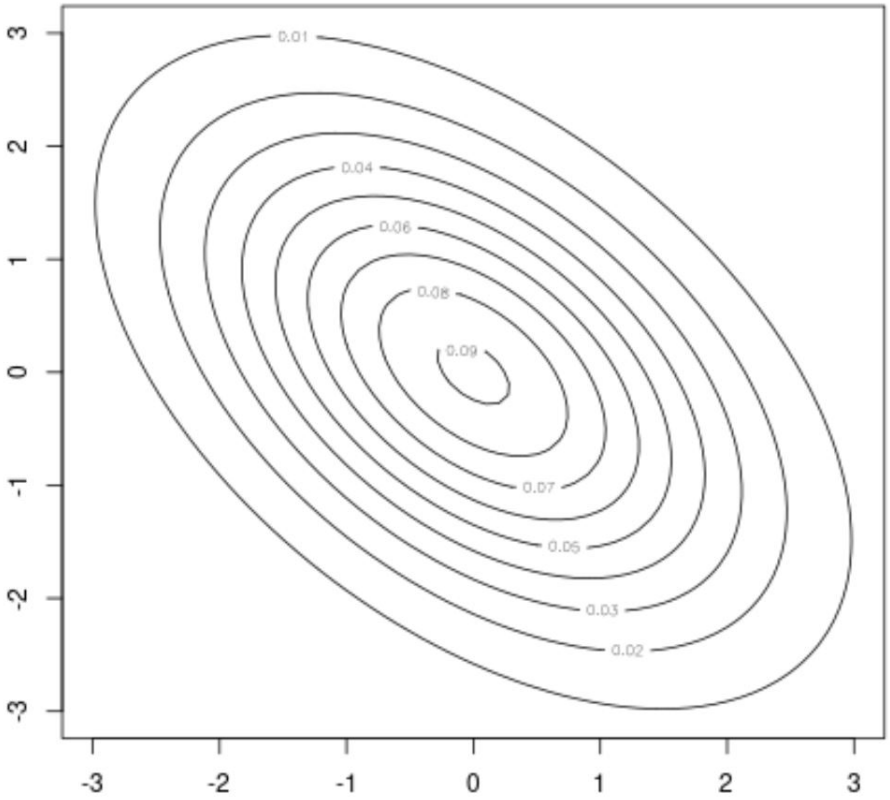
The following code snippet demonstrates how to set up the necessary grid (`x` and `y`), define the distribution parameters (`mu` and `sigma`), calculate the density matrix (`z`) using `dmnorm` and `outer`, and finally plot the results using `contour()`:

```
library(mnormt)

#make this example reproducible
set.seed(0)

#create bivariate normal distribution parameters
x <- seq(-3, 3, 0.1)
y <- seq(-3, 3, 0.1)
mu <- c(0, 0)
sigma <- matrix(c(2, -1, -1, 2), nrow=2)
f <- function(x, y) dmnorm(cbind(x, y), mu, sigma)
z <- outer(x, y, f)

#create contour plot
contour(x, y, z)
```



The resulting **contour plot** confirms the expected elliptical structure of the BND. The lines closest to the center represent the highest probability density, concentrated around the **mean vector** (0, 0). Moving outwards, the ellipses denote progressively lower densities. Critically, the orientation and elongation of these ellipses are directly determined by the covariance value specified in the `sigma` matrix. In this particular visualization, the negative covariance of -1 causes the ellipses to slant diagonally from top-left to bottom-right, visually confirming the inverse linear relationship between the two variables.

Example 3: 3D Visualization using a Surface Plot

While the **contour plot** effectively maps density in 2D, the **surface plot** provides the most direct and intuitive 3D visualization of the distribution's probability density function. This plot vividly displays the classic multi-dimensional "bell curve" shape, where the height of the surface at any coordinate pair (x, y) directly corresponds to the probability density value, f(x, y). This comprehensive visualization method allows users to appreciate the overall volumetric shape and the steepness of the density gradient.

To create this 3D representation in **R**, we reuse the initial setup steps from the contour example, as the underlying density calculations (`x`, `y`, and `z` matrices derived using the **mnormt package**) remain the same. The visualization itself is handled by the `persp()` **function**, a fundamental tool

within R's base graphics system. The strength of `persp()` lies in its extensive control over the viewing perspective and plot aesthetics, which are essential for presenting a clear 3D image.

The code below outlines the process, culminating in the call to `persp()`. Specific arguments are used to control the camera angle and appearance, ensuring the distribution's key features are optimally displayed:

library(mnormt)

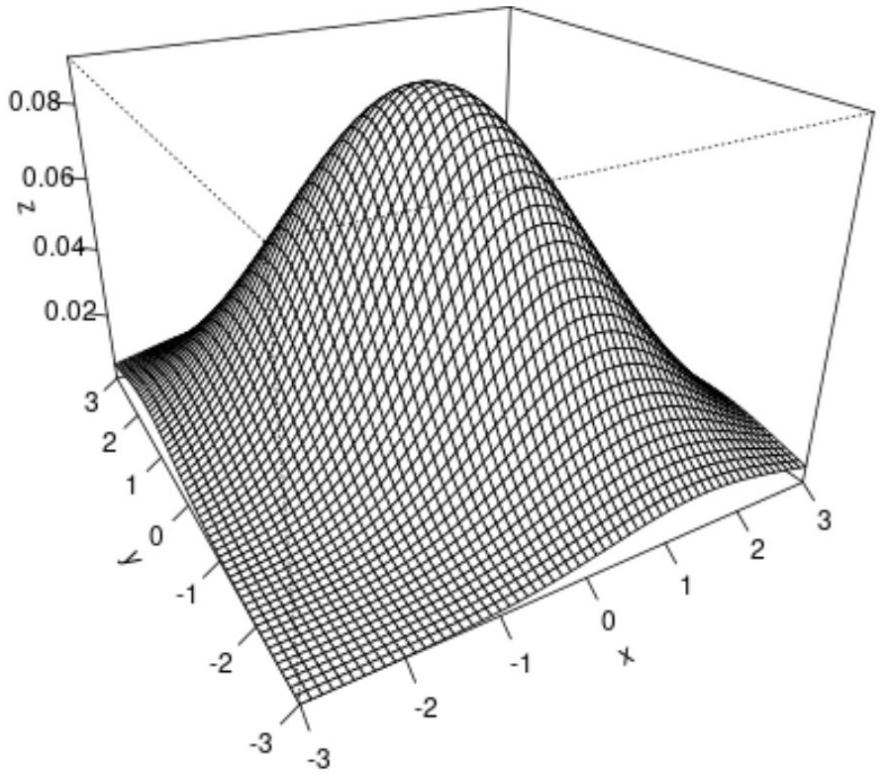
#make this example reproducible
set.seed(0)

#create bivariate normal distribution parameters
x <- seq(-3, 3, 0.1)
y <- seq(-3, 3, 0.1)
mu <- c(0, 0)
sigma <- matrix(c(2, -1, -1, 2), nrow=2)
f <- function(x, y) dmnorm(cbind(x, y), mu, sigma)
z <- outer(x, y, f)

#create surface plot
persp(x, y, z, theta=-30, phi=25, expand=0.6, ticktype='detailed')

The `persp()` **function** includes several critical arguments to define the perspective and scale of the 3D visualization:

`theta`: This controls the azimuthal viewing angle, specifying the horizontal rotation around the vertical z-axis.
`phi`: This controls the polar viewing angle, specifying the vertical tilt or angle of observation from above or below the horizontal plane.
`expand`: This argument dictates the scaling of the z-axis (height) relative to the x and y axes. Using a value less than 1, such as `0.6`, compresses the height, while larger values exaggerate it.
`ticktype`: Setting this to `'detailed'` ensures that the axes are labeled with informative tick marks, aiding in spatial interpretation.



The resultant **surface plot** offers a striking visual confirmation of the distribution's shape. The highest point, the peak of the bell, is correctly situated at the mean (0, 0). The base contours of this surface directly correspond to the ellipses visualized in the 2D **contour plot**, but here, the height provides the essential density information. By manipulating the `theta` and `phi` parameters, analysts can dynamically rotate the view, ensuring that the orientation imposed by the covariance structure is optimally communicated to an audience.

Conclusion: Mastering Bivariate Analysis in R

Through this focused tutorial, we have systematically navigated the fundamental methods required for effectively manipulating and interpreting **probability distributions** in R. We began by establishing a firm theoretical grounding, emphasizing the role of the **mean vector** and the **covariance matrix** as the defining pillars of the bivariate normal structure. Following this, we provided clear instructions on how to leverage the `mvrnorm()` function from the essential **MASS package** to generate synthetic, statistically sound samples, a crucial step for simulation and testing.

Furthermore, the visualization segment introduced two powerful and complementary graphical techniques. The 2D **contour plot** offered an insightful, flat projection that clearly defined regions of equal density and highlighted the distribution's elliptical correlation structure. Conversely, the 3D **surface plot**, constructed using the `persp()` function and density calculations from the **mnormt**

package, provided a full, intuitive representation of the bell-shaped probability density function. Both methods are indispensable for data exploration and sophisticated communication of statistical results.

Achieving proficiency in both the simulation and visualization of bivariate normal distributions constitutes a core competency for advanced statistical modeling and quantitative data analysis in R. These techniques serve as the essential foundation for tackling more intricate multivariate problems and are paramount for accurately interpreting the complex dependencies and relationships inherent between multiple variables encountered in real-world empirical datasets.

Additional Resources for Distribution Analysis

To continue strengthening your analytical capabilities and practical skills in working with various **probability distributions** in R, we highly recommend exploring the following related tutorials and documentation: