How to Calculate Manhattan Distance in R (With Examples)

Authored by

stats writer

December 17, 2025

The concept of Manhattan distance, also widely known as Taxicab Geometry, Rectilinear Distance, or L1-Norm, represents a fundamental measure of path length in a grid-like space. Unlike traditional direct-line measurements, this distance metric calculates the distance between two points based exclusively on movement along strictly horizontal and vertical paths. This strict adherence to orthogonal movement mirrors how a taxi might navigate the structured street grid of a city like Manhattan, hence its common name. This is a critical distinction from the more commonly known Euclidean distance, which measures the shortest straight-line path between two points in space. Understanding the core difference is vital for selecting the appropriate metric in statistical analysis and machine learning tasks, especially when feature measurements are non-commensurate or when the coordinate system imposes axis constraints.

For practitioners utilizing the R programming environment, calculating the Manhattan distance is straightforward, although a dedicated built-in function named simply `manhattan()` is not standard in the base installation. Instead, we typically rely on creating a custom function or leveraging the highly flexible **dist()** function found within the base distribution of R. When calculating this metric for two numeric vectors, the result is always a single non-negative numerical value representing the cumulative difference along each dimension. This metric's simplicity and robustness against outliers in specific contexts make it highly valuable across numerous high-dimensional applications, including specific forms of clustering algorithms and various tasks within data mining and feature selection.

The Mathematical Foundation of Manhattan Distance

To fully grasp the mechanism behind the Manhattan distance, it is essential to review its formal mathematical definition. For two multi-dimensional vectors, denoted as *A* and *B*, the distance is determined by summing the absolute differences of their corresponding coordinates. If $A = (a_1, a_2, dots, a_n)$ and $B = (b_1, b_2, dots, b_n)$, where n is the number of dimensions, the distance D is formally defined by the L1-Norm, which is expressed as follows:

Σ|ai - bi|

In this summation, *i* ranges from 1 to n, representing the ith element, or dimension, in each vector. The use of the absolute difference, indicated by the vertical bars $| cdot |$, ensures that regardless of which vector's coordinate is larger, the resulting distance contribution for that dimension remains positive and contributes to the total distance measure. This metric effectively quantifies the total path length traveled along each axis to move from point *A* to point *B*, maintaining the strict requirement for orthogonal movement, which makes it particularly suitable for scenarios involving grid systems or feature spaces where features are naturally interpreted additively.

This measure of dissimilarity between any two vectors is a foundational component in many statistical and algorithmic processes. Because the Manhattan distance considers only the axis-aligned differences, it tends to be less influenced by single, large deviations along one dimension compared to Euclidean distance (L2-Norm), which squares these differences, thereby exaggerating outliers. This property is highly beneficial when developing robust models and algorithms that are less sensitive to noise or specific scaling issues in the data. The following examples will demonstrate the practical implementation of this calculation using the power and flexibility of the R environment, beginning with a simple, tailored approach.

Advantages in High-Dimensional Data Analysis

When working with large datasets, especially those encountered in modern clustering and pattern recognition tasks, the choice of distance metric can significantly impact the final results. The Manhattan distance often shines in high-dimensional spaces where the concept of "straight-line" distance can become less intuitive or less effective due to the curse of dimensionality. As the number of dimensions increases, the differences between various distance metrics become pronounced. Because the Manhattan distance sums the contributions from each dimension linearly, it provides a stable and interpretable measure of dissimilarity, ensuring that no single dimension dominates the distance calculation disproportionately, assuming the data has been appropriately scaled.

In practical applications such as image processing, where data points represent pixel color intensities across numerous channels, or in recommender systems dealing with large feature vectors, the L1-Norm offers computational advantages and robustness. The calculation involves only absolute differences and summation, avoiding the complex squaring and square root operations required by the Euclidean distance. Furthermore, when dealing with certain types of data, such as binary data or count data, the Manhattan metric naturally aligns with the idea of counting the number of differences or discrepancies between two items, offering a highly intuitive interpretation of dissimilarity.

The consistent use of the absolute difference also makes Manhattan distance a popular choice in certain specialized algorithms, particularly those related to sparse signal processing or specific forms of optimization, where minimizing the L1-Norm is a standard practice. Selecting this metric confirms an underlying assumption about the structure of the data space--namely, that the features are independent and their differences should be aggregated linearly. This contrasts sharply with metrics that assume a smooth, isotropic space, reinforcing why its proper application is a hallmark of sophisticated data mining techniques.

Example 1: Calculating Manhattan Distance Between Two Vectors Using a

Custom Function in R

While base R provides extensive statistical capabilities, it is often beneficial to define custom functions for specialized metrics like the Manhattan distance, especially when seeking maximum transparency and control over the calculation process. The following example demonstrates how to construct a simple, vectorized function that accepts two numeric vectors, computes the absolute difference element-wise, and then sums these differences to yield the final distance value. This method is particularly instructive as it directly translates the mathematical definition into executable code, providing clarity for users exploring the foundations of distance metrics in data science.

```
#create function to calculate Manhattan distance
manhattan_dist <- function(a, b){
dist <- abs(a-b)
dist <- sum(dist)
return(dist)
}

#define two vectors
a <- c(2, 4, 4, 6)

b <- c(5, 5, 7, 8)

#calculate Manhattan distance between vectors
manhattan_dist(a, b)

9
```

The result, as seen in the code output, confirms that the Manhattan distance between vector *a* and vector *b* is precisely **9**. This simple implementation leverages R's powerful vectorization capabilities, meaning the subtraction operation `a-b` and the subsequent absolute value calculation `abs(a-b)` are performed simultaneously across all corresponding elements of the vectors without the need for explicit loops, which greatly enhances computational efficiency and code readability. This custom function provides a direct and elegant solution for calculating the distance between any two numeric vectors of equal length.

Detailed Breakdown of the Custom R Function Logic

Understanding the internal workings of the custom function **manhattan_dist** requires a look at the two core steps executed within the function body. First, the line `dist <- abs(a-b)` performs the element-wise difference and immediately takes the absolute value. For our example vectors $a=(2, 4, 4, 6)$ and $b=(5, 5, 7, 8)$, the intermediate difference vector $(a-b)$ would be $(-3, -1, -3, -2)$.

Applying the base R function **abs()** transforms this into the vector of absolute differences $(3, 1, 3, 2)$. This step directly addresses the requirement of the mathematical L1-Norm formula to only consider the magnitude of the difference along each axis, discarding the direction.

The second critical step is `dist <- sum(dist)`, which completes the calculation by aggregating all the absolute differences. This summation is the final operation that collapses the vector of path segment lengths into a single scalar value representing the total Manhattan distance. Following our calculation, summing $(3, 1, 3, 2)$ yields $3 + 1 + 3 + 2 = 9$. This process rigorously validates the automated result generated by the R code, ensuring conceptual alignment with the mathematical definition. This manual validation is often crucial when developing new algorithms or verifying the results of customized statistical tools.

To further solidify the understanding, we can confirm this result through a quick, step-by-step manual calculation, demonstrating how the components of the L1-Norm formula are executed:

Σ|ai - bi| = |2-5| + |4-5| + |4-7| + |6-8| = 3 + 1 + 3 + 2 = **9**.

Example 2: Calculating Manhattan Distance Between Vectors in a Matrix Using the Built-in dist() Function

While defining a custom function is excellent for conceptual clarity and calculating the distance between just two vectors, real-world data science tasks often require calculating the pairwise distance between every vector (or row) within a large dataset matrix. For this purpose, R provides the highly optimized and versatile built-in function, **dist()**. The `dist()` function is specifically designed to calculate distance matrices for data objects, offering support for several standard distance metrics, including Euclidean, Maximum, Binary, and importantly, Manhattan.

When utilizing `dist()`, the input data should be structured as a matrix where each row represents a distinct data point (vector) and each column represents a dimension or feature. This structure allows the function to efficiently compute the dissimilarity between all unique pairs of rows in the matrix. The key to calculating the Manhattan distance using this built-in function is specifying the `method` argument as `"manhattan"`, which instructs R to perform the L1-Norm calculation iteratively across all combinations of rows. This approach dramatically simplifies the code required for complex pairwise comparisons, as shown in the following example:

#create four vectors
a <- c(2, 4, 4, 6)

b <- c(5, 5, 7, 8)

c <- c(9, 9, 9, 8)

```
d <- c(1, 2, 3, 3)

#bind vectors into one matrix
mat <- rbind(a, b, c, d)

#calculate Manhattan distance between each vector in the matrix
dist(mat, method = "manhattan")
```

```
a b c
b 9
c 19 10
d 7 16 26
```

The preparatory steps involve defining four distinct vectors (a, b, c, and d), all of which must be of the same length to ensure the resulting matrix is well-formed. The `rbind()` function then stacks these vectors row-wise into a single matrix object named **mat**. Finally, calling `dist(mat, method = "manhattan")` executes the pairwise distance calculation, producing a highly concise output structure known as a distance matrix object. This matrix is essential for algorithms like hierarchical clustering, where the similarity (or dissimilarity) between all data points must be known prior to grouping.

Interpreting the Output Distance Matrix

The output from the `dist()` function is a specialized matrix structure designed to save space by displaying only the lower triangle of the distance matrix, as the distance from A to B is identical to the distance from B to A (the matrix is symmetric), and the distance from a point to itself is always zero (the diagonal elements). Understanding how to read this structure is fundamental to utilizing the results effectively in subsequent analytical steps. The rows correspond to the subsequent data points (starting from the second point), and the columns correspond to the preceding data points.

For our example output, the interpretation is straightforward once the structure is recognized. The matrix summarizes the total Manhattan distance for every unique pairing among the four vectors (a, b, c, d). We can extract the following specific pairwise distances from the triangular matrix:

The Manhattan distance between vector *a* and vector *b* is found in the row labeled 'b' under the column 'a', which is **9**.
The Manhattan distance between vector *a* and vector *c* is found in the row labeled 'c' under the column 'a', which is **19**.
The Manhattan distance between vector *a* and vector *d* is found in the row labeled 'd' under the column 'a', which is **7**.
The Manhattan distance between vector *b* and vector *c* is found in the row labeled 'c' under the

column 'b', which is **10**.
The Manhattan distance between vector *b* and vector *d* is found in the row labeled 'd' under the column 'b', which is **16**.
The Manhattan distance between vector *c* and vector *d* is found in the row labeled 'd' under the column 'c', which is **26**.

It is paramount to observe that each vector in the matrix should be the same length. If the vectors were of unequal lengths, the `rbind()` operation might produce unexpected behavior or errors, and the subsequent distance calculation would certainly fail, as the mathematical premise of pairwise distance calculation requires perfect alignment across all features.

Further Resources on Distance Metrics in R

The Manhattan distance is just one of several critical metrics used in data science. Depending on the data structure and analytical goals, other metrics might be more appropriate. For those interested in exploring the full spectrum of distance measurements available in R, especially those often contrasted with the L1-Norm, the following resources provide excellent introductory material. These complementary metrics--Euclidean, Mahalanobis, and Minkowski--each offer a unique perspective on defining dissimilarity, crucial knowledge for any statistician or data scientist.

The Euclidean Distance, or L2-Norm, focuses on the direct spatial distance, often preferred when feature correlations are low and geometric intuition is paramount. The Mahalanobis distance accounts for the covariance structure of the data, adjusting for correlations and scaling issues, making it a powerful tool in multivariate analysis. Finally, the Minkowski distance is a generalization of both Manhattan and Euclidean distances, allowing analysts to tune the sensitivity to individual dimensional differences via a parameter p. Mastering these different measures ensures flexibility and accuracy in interpreting complex data structures.

How to Calculate Euclidean Distance in R
How to Calculate Mahalanobis Distance in R
How to Calculate Minkowski Distance in R