Understanding Quantiles: A Comprehensive Guide to the quantile() Function in R

Authored by

Mohammed looti

November 3, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Understanding Quantiles: A Comprehensive Guide to the quantile() Function in R*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=9248

In the field of statistics and data science, accurately understanding the shape, spread, and central tendency of a dataset is paramount. **Quantiles** serve as crucial descriptive statistics, dividing a probability distribution or a sorted dataset into continuous intervals that possess equal probability. These divisions are fundamental for identifying data spread, detecting skewness, and flagging potential outliers, making them indispensable tools for any rigorous analysis.

The primary utility for computing these vital statistical measures within the R programming language is the built-in `quantile()` function. This powerful function efficiently calculates sample quantiles for any provided numeric vector or specific column within a larger dataset. Its flexibility allows data analysts to define precise divisions necessary for deep, insightful data inspection.

Mastering the `quantile()` Function Syntax in R

While the basic structure of the `quantile()` function is intuitive, understanding its optional parameters is key to leveraging its full statistical power and ensuring the results are appropriate for the data type being analyzed. Grasping these arguments allows both novice and expert users to tailor quantile calculation to meet complex analytical requirements.

The comprehensive syntax for calling the function is:

quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE, type = 7)

Below is a detailed breakdown of the essential arguments that govern the function's execution and output:

x: This is the mandatory argument. It specifies the numeric vector or the identified column within a data structure for which the quantile values must be calculated.
probs: This critical parameter requires a numeric vector containing probabilities, which must range inclusively from 0 to 1. By default, R calculates the standard quartiles (0%, 25%, 50%, 75%, 100%). Users can easily customize this vector to calculate other divisions, such as deciles (tenths), quintiles (fifths), or any specific percentile of interest.
na.rm: A logical parameter (either **TRUE** or **FALSE**) that dictates whether missing values (NA) should be excluded from the computation. Setting this argument to **TRUE** is frequently necessary to prevent the entire function from returning a missing value result if any NA is present in the input vector.
type: This argument specifies the underlying algorithm used to calculate the sample quantiles. While often left at the default setting, it is statistically significant. R provides nine distinct methods (types 1 through 9). Type 7, the default method, is generally recommended for analyzing continuous data distributions due to its statistical properties.

The following practical examples illustrate how to effectively apply the `quantile()` function across

common data structures, providing immediate insights into various analytical requirements in R.

Example 1: Calculating Quantiles for a Single Numeric Vector

The most straightforward and common usage of the **`quantile()`** function involves analyzing a single, ordered collection of numerical data. This initial analysis provides a rapid and robust assessment of the data's central location and overall spread, forming a core component of exploratory data analysis (EDA).

To begin, we define a sample dataset. We then demonstrate how to calculate standard data divisions--quartiles, quintiles, and deciles--which yield immediate insights into the underlying distribution. Quartiles divide the data into four equal parts, quintiles into five, and deciles into ten. The `seq(0, 1, step)` function is highly useful here, simplifying the creation of the required probability vectors for these standardized divisions.

#define vector of data
data = c(1, 3, 3, 4, 5, 7, 8, 9, 12, 13, 13, 15, 18, 20, 22, 23, 24, 28)

#calculate quartiles (0%, 25%, 50%, 75%, 100%)
quantile(data, probs = seq(0, 1, 1/4))

0% 25% 50% 75% 100%
1.0 5.5 12.5 19.5 28.0

#calculate quintiles (dividing the data into five equal parts)
quantile(data, probs = seq(0, 1, 1/5))

0% 20% 40% 60% 80% 100%
1.0 4.4 8.8 13.4 21.2 28.0

#calculate deciles (dividing the data into ten equal parts)
quantile(data, probs = seq(0, 1, 1/10))

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
1.0 3.0 4.4 7.1 8.8 12.5 13.4 17.7 21.2 23.3 28.0

#calculate random quantiles of interest (e.g., 20th, 50th, and 90th percentiles)
quantile(data, probs = c(.2, .5, .9))

20% 50% 90%
4.4 12.5 23.3

The output clearly presents the exact value corresponding to each specified probability level. For instance, the median (50th percentile) of this sample dataset is 12.5, which signifies that exactly 50% of the data points fall at or below this value. The capacity to select arbitrary probabilities, as demonstrated in the final calculation block, underscores the function's flexibility in highly focused analytical tasks.

Example 2: Analyzing Quantiles within an R Data Frame

While analyzing a standalone vector is a fundamental skill, most complex, real-world data is structured within a data frame in R. When working with this structure, it becomes necessary to explicitly specify the column targeted for quantile calculation, typically accomplished using the dollar sign ($) operator to subset the data.

We first establish a sample data frame consisting of three different numeric variables--`var1`, `var2`, and `var3`--to showcase the method for targeting a single, specific column for quartile computation. This is a common requirement when examining variable distributions individually.

```
#create data frame
df <- data.frame(var1=c(1, 3, 3, 4, 5, 7, 7, 8, 12, 14, 18),
var2=c(7, 7, 8, 3, 2, 6, 8, 9, 11, 11, 16),
var3=c(3, 3, 6, 6, 8, 4, 4, 7, 10, 10, 11))

#calculate quartiles of column 'var2'
quantile(df$var2, probs = seq(0, 1, 1/4))

0% 25% 50% 75% 100%
2.0 6.5 8.0 10.0 16.0
```

The resulting output provides the quartile summary exclusively for `var2`. The 50th percentile (median) is 8.0. Furthermore, the Interquartile Range (IQR), a vital measure of variability, can be quickly calculated by subtracting the 25th percentile (6.5) from the 75th percentile (10.0), offering immediate insight into the spread of values for that specific variable.

For scenarios requiring simultaneous analysis of all numeric columns in the data frame, the **`sapply()`** function offers a highly efficient solution. `sapply()` iteratively applies a specified function--in this case, `quantile()`--across every relevant element of the list or data frame, returning the results in a simplified, easy-to-read matrix format for comparative analysis.

```
#calculate quartiles of every column
sapply(df, function(x) quantile(x, probs = seq(0, 1, 1/4)))
```

var1 var2 var3
0% 1.0 2.0 3
25% 3.5 6.5 4
50% 7.0 8.0 6
75% 10.0 10.0 9
100% 18.0 16.0 11

This streamlined method produces a concise summary table, enabling direct comparison of data spread across `var1`, `var2`, and `var3`. We can instantly deduce that `var1` exhibits the largest overall range and holds the highest 75th percentile value among the three variables, prompting further investigation into its distribution.

Example 3: Grouped Quantile Calculation using the `dplyr` Package

In advanced data analysis, it is frequently necessary to calculate descriptive statistics, such as quantiles, separately for defined subsets of the data based on a categorical grouping variable. While base R provides capabilities for this, the modern R environment heavily favors the powerful **dplyr** package for highly efficient data manipulation and aggregation tasks, particularly within tidyverse workflows.

By utilizing `dplyr`, we implement the pipe operator (`%>%`) to sequentially apply functions. This workflow significantly enhances the readability and robustness of group-wise quantile calculation. We establish a new data frame containing a categorical grouping variable, `team`, alongside a numeric variable, `points`, which we intend to summarize by group.

library(dplyr)

#define data frame
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'C', 'C', 'C'),
points=c(1, 3, 3, 4, 5, 7, 7, 8, 12, 14, 18))

#define quantiles of interest (25th, 50th, 75th percentiles)
q = c(.25, .5, .75)

#calculate quantiles by grouping variable using dplyr workflow
df %>%
group_by(team) %>%
summarize(quant25 = quantile(points, probs = q),
quant50 = quantile(points, probs = q),
quant75 = quantile(points, probs = q))

A tibble: 3 x 4
team quant25 quant50 quant75

1 A 2.5 3 3.25
2 B 6.5 7 7.25
3 C 13 14 16

The provided code snippet first loads the `dplyr` package, then groups the entire data frame based on the `team` variable. Subsequently, the `summarize()` function computes the specified percentiles of the `points` column exclusively within each distinct team group. The final result, presented as a clean tibble, offers distinct quantile summaries for Team A, Team B, and Team C, proving invaluable for comparative statistical analysis across different categories.

Conclusion: Leveraging Quantiles for Robust Data Understanding

The **`quantile()`** function stands as an indispensable utility in R, forming the bedrock for performing robust exploratory data analysis across diverse applications. Whether your task involves examining a simple numeric vector, targeting specific variables within a complex data frame, or executing sophisticated aggregated analysis using modern packages like `dplyr`, proficiency in this function unlocks profound insights into the structure and distribution characteristics of your data.

Accurate quantile calculation facilitates crucial analytical steps, such as understanding where the majority of your data points reside relative to the median (50th percentile) and precisely identifying the boundaries that encompass the middle 50% of your observations (the Interquartile Range). Always remember the importance of considering the `type` argument when aligning your calculations with specific statistical methodologies or requirements for inference.

Additional Resources for R Quantiles

The following resources offer further guidance on using the **`quantile()`** function to calculate other common quantile values and explore advanced applications: