How Do I Print All Rows of a Tibble in R?

Authored by

stats writer

December 23, 2025

When working within the statistical programming environment of R, data manipulation often involves handling complex and extensive datasets. The modern standard for organizing this information, particularly within the tidyverse ecosystem, is the **tibble**. Unlike traditional R data frames, tibbles are designed with a sophisticated, refined print method engineered specifically for efficiency and user experience when dealing with immense volumes of data. This design choice is fundamental: by default, simply calling a tibble's name in the console will only display a limited subset of the data, typically the first 10 rows and columns that fit the screen width, alongside metadata indicating the full dimensions.

This default behavior serves a critical purpose: it prevents the R console from being overwhelmed and frozen by the attempt to render hundreds of thousands of rows simultaneously. However, there are scenarios in advanced data analysis, debugging, or presentation preparation where a user might genuinely need to inspect or output the complete structure, including every single observation. While the standard approach to viewing data in R involves using the print() function, accessing all rows of a tibble requires explicitly overriding its restrictive default settings. Understanding how to customize the display parameters of the **print() function** is essential for full control over your data visualization workflow in R.

Understanding the Tibble Philosophy: Why Default Printing Is Limited

A **tibble** is essentially a modern, streamlined version of the R data frame, packaged primarily within the dplyr library, though it is foundational to the entire tidyverse suite. The most defining characteristic of the tibble is its refined print method, which is implemented precisely to enhance performance and readability. When a standard data frame is printed, R attempts to output every single row, which is highly problematic for large datasets, often leading to slow operation or filling the console output with unnecessary information that requires extensive scrolling.

The tibble solves this problem by enforcing a principle of "lazy" evaluation for printing. It defaults to showing only the first ten rows, along with a summary of the remaining dimensions (e.g., "70 more rows"), and crucially, it displays the data types of the columns immediately below the header. This metadata is extremely helpful during the iterative process of data cleaning and transformation. The limited display is not a hindrance but a feature designed to keep the analytical workflow fast and focused on the variables and their structure, rather than on every individual observation.

Demonstrating the Default Tibble Output

To illustrate the standard printing behavior of a **tibble**, let us construct a sample dataset containing 80 rows and 2 columns. We utilize the dplyr package to create the tibble and employ the set.seed function to ensure that the random data generation is reproducible across different sessions. Note how simply calling the object name, `data`, results in truncated output.

For example, consider the following tibble with 80 rows and 2 columns:

```
#load dplyr
library(dplyr)

#make this example reproducible
set.seed(1)

#create tibble
data <- tibble(a = rnorm(80),
b = rnorm(80))

#view tibble
data
```

```
# A tibble: 80 x 2
a b

1 -0.626 -0.569
2 0.184 -0.135
3 -0.836 1.18
4 1.60 -1.52
5 0.330 0.594
6 -0.820 0.333
7 0.487 1.06
8 0.738 -0.304
9 0.576 0.370
10 -0.305 0.267
# ... with 70 more rows
```

When we type in the name of the tibble in R, it will only show the first 10 rows by default. However, it does clearly alert the user that there are **70 more rows** that are not being displayed, which is an improvement over traditional data frames that might silently overwhelm the console. This concise summary allows the analyst to quickly verify the structure and initial contents without scrolling.

While this feature is beneficial for large-scale exploratory analysis, specific use cases, such as running small regression checks or verifying data integrity where all 80 rows fit comfortably within the console, necessitate overriding this truncation. This leads us to the primary methods for adjusting the printing behavior.

Print a Specific Number of Rows of a Tibble

The most straightforward approach to gaining more control over the output is by leveraging the versatile print() function directly. Although R automatically calls this function when an object name is typed, using it explicitly allows us to pass specific arguments that modify the default behavior. To print a customized number of rows, we utilize the mandatory `n` argument within the function call.

The `n` argument dictates the maximum number of rows that should be displayed in the console output. By specifying a numerical value greater than the default 10, we can expand the view to match our analytical requirements. For instance, if our tibble has 80 rows and we decide that viewing the top 20 rows provides sufficient context for a specific task, we simply set `n = 20` when invoking the **print() function**.

You can print a specific number of rows of a tibble by specifying a number in the **print()** function:

#print first 20 rows of tibble
print(data, n=20)

 # A tibble: 80 x 2
a b

1 -0.626 -0.569
2 0.184 -0.135
3 -0.836 1.18
4 1.60 -1.52
5 0.330 0.594
6 -0.820 0.333
7 0.487 1.06
8 0.738 -0.304
9 0.576 0.370
10 -0.305 0.267
11 1.51 -0.543
12 0.390 1.21
13 -0.621 1.16
14 -2.21 0.700
15 1.12 1.59
16 -0.0449 0.558
17 -0.0162 -1.28
18 0.944 -0.573
19 0.821 -1.22
20 0.594 -0.473

... with 60 more rows

Upon executing the code above, the console output immediately expands to display 20 rows of the **tibble**. Crucially, the tibble retains its smart printing summary, confirming the total dimensions (80x2) and indicating that 60 rows remain truncated. This method offers a flexible middle ground between the default conservative display and the eventual requirement of printing the entire dataset.

Leveraging the Pipe Operator for Concise Printing

A common practice in modern R programming, especially within the tidyverse, is the use of the pipe operator (`%>%`). This operator allows for chaining multiple functions together, passing the output of one function as the first argument (or data input) to the next function. Integrating the **pipe operator** into the printing routine often results in cleaner and more readable code, particularly when the printing step follows a sequence of data manipulation steps.

The ability to pipe the data object directly into the **print() function** is highly advantageous as it aligns with the functional programming paradigm favored in R. Instead of calling `print(data, n=20)`, which places the function call first, we write `data %>% print(n=20)`. This sequence visually represents the flow of data: the `data` object flows into the `print()` function, which then executes the display command using the specified parameter `n`.

You can also use the pipe operator to achieve the same result:

#print first 20 rows of tibble
data %>% print(n=20)

 # A tibble: 80 x 2
a b

1 -0.626 -0.569
2 0.184 -0.135
3 -0.836 1.18
4 1.60 -1.52
5 0.330 0.594
6 -0.820 0.333
7 0.487 1.06
8 0.738 -0.304
9 0.576 0.370
10 -0.305 0.267

11 1.51 -0.543
12 0.390 1.21
13 -0.621 1.16
14 -2.21 0.700
15 1.12 1.59
16 -0.0449 0.558
17 -0.0162 -1.28
18 0.944 -0.573
19 0.821 -1.22
20 0.594 -0.473
... with 60 more rows

This approach is highly recommended when embedding print statements within complex data pipelines, ensuring that the code remains cohesive and easily understandable by other developers. Whether you use the direct function call or the **pipe operator**, the result--displaying a specific number of rows (20 in this case)--remains identical.

Print All Rows of a Tibble

When the analytical need truly demands viewing every single observation, irrespective of the performance implications or console length, the dedicated setting for the **print() function** must be engaged. To instruct R to display the entire contents of the **tibble**, including all rows and columns, the `n` argument must be set to the value representing infinity (`Inf`).

Setting `n = Inf` effectively removes the truncation limit imposed by the tibble's default print method. This command tells R to keep printing rows until the end of the data frame is reached. While this is acceptable for datasets like our 80-row example, caution should be exercised when applying this method to extremely large datasets (e.g., millions of rows), as it may consume significant memory resources and result in an unmanageably long console output.

You can print every row of a tibble by specifying **n = Inf**. We continue to use the **pipe operator** for idiomatic R programming:

#print all rows of tibble
data %>% print(n=Inf)

 # A tibble: 80 x 2
a b

1 -0.626 -0.569

2 0.184 -0.135
3 -0.836 1.18
4 1.60 -1.52
5 0.330 0.594
6 -0.820 0.333
7 0.487 1.06
8 0.738 -0.304
9 0.576 0.370
10 -0.305 0.267
11 1.51 -0.543
12 0.390 1.21
13 -0.621 1.16
14 -2.21 0.700
15 1.12 1.59
16 -0.0449 0.558
17 -0.0162 -1.28
18 0.944 -0.573
19 0.821 -1.22
20 0.594 -0.473
21 0.919 -0.620
22 0.782 0.0421
23 0.0746 -0.911
24 -1.99 0.158
25 0.620 -0.655
26 -0.0561 1.77
27 -0.156 0.717
28 -1.47 0.910
29 -0.478 0.384
30 0.418 1.68
31 1.36 -0.636
32 -0.103 -0.462
33 0.388 1.43
34 -0.0538 -0.651
35 -1.38 -0.207
36 -0.415 -0.393
37 -0.394 -0.320
38 -0.0593 -0.279
39 1.10 0.494
40 0.763 -0.177

41 -0.165 -0.506
42 -0.253 1.34
43 0.697 -0.215
44 0.557 -0.180
45 -0.689 -0.100
46 -0.707 0.713
47 0.365 -0.0736
48 0.769 -0.0376
49 -0.112 -0.682
50 0.881 -0.324
51 0.398 0.0602
52 -0.612 -0.589
53 0.341 0.531
54 -1.13 -1.52
55 1.43 0.307
56 1.98 -1.54
57 -0.367 -0.301
58 -1.04 -0.528
59 0.570 -0.652
60 -0.135 -0.0569
61 2.40 -1.91
62 -0.0392 1.18
63 0.690 -1.66
64 0.0280 -0.464
65 -0.743 -1.12
66 0.189 -0.751
67 -1.80 2.09
68 1.47 0.0174
69 0.153 -1.29
70 2.17 -1.64
71 0.476 0.450
72 -0.710 -0.0186
73 0.611 -0.318
74 -0.934 -0.929
75 -1.25 -1.49
76 0.291 -1.08
77 -0.443 1.00
78 0.00111 -0.621
79 0.0743 -1.38

80 -0.590 1.87

After running this command, the summary line indicating truncated rows disappears, and the full 80 rows are printed consecutively to the console. This method is the definitive solution for printing every observation in a tibble directly to the standard output.

Alternative Methods for Data Inspection

While using `print(n = Inf)` addresses the console output requirement, it is often not the most efficient or practical way to inspect large datasets. R provides several graphical and interactive alternatives that offer better visualization and ease of navigation for comprehensive data review.

One popular alternative is the `View()` function, which is distinct from the **print() function**. The `View()` function opens the **tibble** in a separate, dedicated spreadsheet-style window within the RStudio environment or R's default graphical user interface (GUI). This window is fully scrollable and searchable, allowing for quick inspection of all data elements without flooding the console log. This method is highly recommended for reviewing datasets that are too large for comfortable console printing but still require comprehensive row-by-row visibility.

Another powerful technique involves using functions designed for detailed structural review, such as `glimpse()` from the dplyr package. While `glimpse()` does not print all rows, it transposes the output, displaying columns vertically and revealing more data type information and initial values than the default tibble print method, making it superior for checking structural integrity and variable types.

Configuration Options for Global Printing Defaults

For analysts who frequently work with smaller tibbles and find the default 10-row limit restrictive, it is possible to change the global printing defaults for **tibble** objects. This modification allows the user to set a higher default row count without having to repeatedly specify the `n` parameter in every **print() function** call.

This is achieved by modifying R's options. Specifically, the option `tibble.print_max` controls the maximum number of rows that will be printed before truncation occurs. By setting this value higher, you can adjust the default behavior. However, it is generally recommended to keep this setting at a moderate level (e.g., 20 or 30 rows) to retain the core performance benefit of the tibble. Setting it to an extremely high number globally defeats the purpose of the tibble's design.

Conversely, if you want to limit the width of the display (how many columns are shown), you can adjust the `tibble.width` option. By default, tibbles attempt to smartly determine the width based on the console size, but manual adjustment provides granular control over how the output is

wrapped and displayed, further ensuring clean visualization of the data frame structure.

Summary of Printing Strategies

Managing the display of data is a crucial skill in R programming, especially when transitioning from traditional **data frame**s to the modern tibble structure. The ability to control the print output ensures both efficiency and clarity in the analytical process.

Default Print: Typing the object name displays the first 10 rows, prioritizing speed and console management.
Specific Row Count: Use `print(data, n = X)` or `data %>% print(n = X)` to view X number of rows.
Print All Rows: Use `print(data, n = Inf)` to override the truncation limit and display the entire dataset in the console.
Interactive View: For large datasets that should not clog the console, use `View(data)` to open the tibble in a dedicated, searchable GUI window.

Mastering these methods ensures that whether you are performing quick data checks or outputting final verified data structures, you maintain full control over your R environment. For those looking to deepen their understanding of data manipulation in R, continued study of the tibbles chapter in *R for Data Science* provides excellent foundational knowledge.

You can find more R tutorials here.