# Tibbles

Source: [vignettes/tibble.Rmd](https://github.com/tidyverse/tibble/blob/master/vignettes/tibble.Rmd)

Tibbles are a modern take on data frames. They keep the features that have stood the test of time, and drop the features that used to be convenient but are now frustrating (i.e. converting character vectors to factors).

## Creating

[tibble()](https://tibble.tidyverse.org/reference/tibble.html) is a nice way to create data frames. It encapsulates best practices for data frames:

* It never changes an input’s type (i.e., no more stringsAsFactors = FALSE!).

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(x = letters)

#> # A tibble: 26 x 1

#> **x**

#> *<chr>*

#> 1 a

#> 2 b

#> 3 c

#> 4 d

#> 5 e

#> 6 f

#> 7 g

#> 8 h

#> 9 i

#> 10 j

#> # … with 16 more rows

This makes it easier to use with list-columns:

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(x = 1:3, y = [list](https://www.rdocumentation.org/packages/base/topics/list)(1:5, 1:10, 1:20))

#> # A tibble: 3 x 2

#> **x** **y**

#> <int> <list>

#> 1 1 <int [5]>

#> 2 2 <int [10]>

#> 3 3 <int [20]>

List-columns are most commonly created by do(), but they can be useful to create by hand.

* It never adjusts the names of variables:

[names](https://www.rdocumentation.org/packages/base/topics/names)([data.frame](https://www.rdocumentation.org/packages/base/topics/data.frame)(`crazy name` = 1))

#> [1] "crazy.name"

[names](https://www.rdocumentation.org/packages/base/topics/names)([tibble](https://tibble.tidyverse.org/reference/tibble.html)(`crazy name` = 1))

#> [1] "crazy name"

* It evaluates its arguments lazily and sequentially:

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(x = 1:5, y = x ^ 2)

#> # A tibble: 5 x 2

#> **x** **y**

#> <int> <dbl>

#> 1 1 1

#> 2 2 4

#> 3 3 9

#> 4 4 16

#> 5 5 25

* It never uses [row.names()](https://www.rdocumentation.org/packages/base/topics/row.names). The whole point of tidy data is to store variables in a consistent way. So it never stores a variable as special attribute.
* It only recycles vectors of length 1. This is because recycling vectors of greater lengths is a frequent source of bugs.

## Coercion

To complement [tibble()](https://tibble.tidyverse.org/reference/tibble.html), tibble provides [as\_tibble()](https://tibble.tidyverse.org/reference/as_tibble.html) to coerce objects into tibbles. Generally, [as\_tibble()](https://tibble.tidyverse.org/reference/as_tibble.html) methods are much simpler than [as.data.frame()](https://www.rdocumentation.org/packages/base/topics/as.data.frame) methods, and in fact, it’s precisely what [as.data.frame()](https://www.rdocumentation.org/packages/base/topics/as.data.frame) does, but it’s similar to [do.call(cbind, lapply(x, data.frame))](https://www.rdocumentation.org/packages/base/topics/do.call) - i.e. it coerces each component to a data frame and then cbinds() them all together.

[as\_tibble()](https://tibble.tidyverse.org/reference/as_tibble.html) has been written with an eye for performance:

l <- [replicate](https://www.rdocumentation.org/packages/base/topics/lapply)(26, [sample](https://www.rdocumentation.org/packages/base/topics/sample)(100), simplify = FALSE)

[names](https://www.rdocumentation.org/packages/base/topics/names)(l) <- letters

timing <- bench::[mark](https://www.rdocumentation.org/packages/bench/topics/mark)(

[as\_tibble](https://tibble.tidyverse.org/reference/as_tibble.html)(l),

[as.data.frame](https://www.rdocumentation.org/packages/base/topics/as.data.frame)(l),

check = FALSE

)

timing

#> # A tibble: 2 x 14

#> expression min mean median max `itr/sec` mem\_alloc n\_gc

#> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>

#> 1 as\_tibble… 2.88e-4 6.25e-4 3.27e-4 0.00451 1600. 1840 5

#> 2 as.data.f… 7.92e-4 1.66e-3 1.10e-3 0.00765 601. 34584 5

#> # … with 6 more variables: **n\_itr** *<int>*, **total\_time** *<dbl>*, **result** *<list>*,

#> # **memory** *<list>*, **time** *<list>*, **gc** *<list>*

The speed of [as.data.frame()](https://www.rdocumentation.org/packages/base/topics/as.data.frame) is not usually a bottleneck when used interactively, but can be a problem when combining thousands of messy inputs into one tidy data frame.

## Tibbles vs data frames

There are three key differences between tibbles and data frames: printing, subsetting, and recycling rules.

## Printing

When you print a tibble, it only shows the first ten rows and all the columns that fit on one screen. It also prints an abbreviated description of the column type, and uses font styles and color for highlighting:

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(x = -5:1000)

#> # A tibble: 1,006 x 1

#> **x**

#> *<int>*

#> 1 -5

#> 2 -4

#> 3 -3

#> 4 -2

#> 5 -1

#> 6 0

#> 7 1

#> 8 2

#> 9 3

#> 10 4

#> # … with 996 more rows

You can control the default appearance with options:

* [options(tibble.print\_max = n, tibble.print\_min = m)](https://www.rdocumentation.org/packages/base/topics/options): if there are more than n rows, print only the first mrows. Use [options(tibble.print\_max = Inf)](https://www.rdocumentation.org/packages/base/topics/options) to always show all rows.
* [options(tibble.width = Inf)](https://www.rdocumentation.org/packages/base/topics/options) will always print all columns, regardless of the width of the screen.

## Subsetting

Tibbles are quite strict about subsetting. [ always returns another tibble. Contrast this with a data frame: sometimes [returns a data frame and sometimes it just returns a vector:

df1 <- [data.frame](https://www.rdocumentation.org/packages/base/topics/data.frame)(x = 1:3, y = 3:1)

[class](https://www.rdocumentation.org/packages/base/topics/class)(df1[, 1:2])

#> [1] "data.frame"

[class](https://www.rdocumentation.org/packages/base/topics/class)(df1[, 1])

#> [1] "integer"

df2 <- [tibble](https://tibble.tidyverse.org/reference/tibble.html)(x = 1:3, y = 3:1)

[class](https://www.rdocumentation.org/packages/base/topics/class)(df2[, 1:2])

#> [1] "tbl\_df" "tbl" "data.frame"

[class](https://www.rdocumentation.org/packages/base/topics/class)(df2[, 1])

#> [1] "tbl\_df" "tbl" "data.frame"

To extract a single column use [[ or $:

[class](https://www.rdocumentation.org/packages/base/topics/class)(df2[[1]])

#> [1] "integer"

[class](https://www.rdocumentation.org/packages/base/topics/class)(df2$x)

#> [1] "integer"

Tibbles are also stricter with $. Tibbles never do partial matching, and will throw a warning and return NULL if the column does not exist:

df <- [data.frame](https://www.rdocumentation.org/packages/base/topics/data.frame)(abc = 1)

df$a

#> [1] 1

df2 <- [tibble](https://tibble.tidyverse.org/reference/tibble.html)(abc = 1)

df2$a

#> Warning: Unknown or uninitialised column: 'a'.

#> NULL

As of version 1.4.1, tibbles no longer ignore the drop argument:

[data.frame](https://www.rdocumentation.org/packages/base/topics/data.frame)(a = 1:3)[, "a", drop = TRUE]

#> [1] 1 2 3

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(a = 1:3)[, "a", drop = TRUE]

#> [1] 1 2 3

## Recycling

When constructing a tibble, only values of length 1 are recycled. The first column with length different to one determines the number of rows in the tibble, conflicts lead to an error. This also extends to tibbles with *zero* rows, which is sometimes important for programming:

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(a = 1, b = 1:3)

#> # A tibble: 3 x 2

#> **a** **b**

#> <dbl> <int>

#> 1 1 1

#> 2 1 2

#> 3 1 3

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(a = 1:3, b = 1)

#> # A tibble: 3 x 2

#> **a** **b**

#> <int> <dbl>

#> 1 1 1

#> 2 2 1

#> 3 3 1

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(a = 1:3, c = 1:2)

#> Error: Tibble columns must have consistent lengths, only values of length one are recycled:

#> \* Length 2: Column `c`

#> \* Length 3: Column `a`

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(a = 1, b = [integer](https://www.rdocumentation.org/packages/base/topics/integer)())

#> # A tibble: 0 x 2

#> # … with 2 variables: **a** *<dbl>*, **b** *<int>*

[tibble](https://tibble.tidyverse.org/reference/tibble.html)(a = [integer](https://www.rdocumentation.org/packages/base/topics/integer)(), b = 1)

#> # A tibble: 0 x 2

#> # … with 2 variables: **a** *<int>*, **b** *<dbl>*