

# Data Frames

*David Gerard*

*2019-01-23*

Based on a bootcamp originally written by Sohail Nizam.

## Learning Objectives

- Understand the fundamental type for storing data in R.
- Create and manipulate data frames (Tibbles), extract variables.
- Chapter 10 in R for Data Science

## Data Frames

- Usually we have more than one vector (variable) in our data set.
- So how can we store several vectors together?
- We'll use something called a data frame.
- It's important to think of a data frame as a collection of columns, not a collection of rows because *that's how R thinks of it*.
- When you have a data frame, you can easily refer to specific columns.
- But referring to rows becomes more complicated.
- Just like vectors are created with the `c()` function taking a collection of elements of the same type as input, data frames are created with the `data.frame()` function taking a collection of vectors as input.
- The vectors can be of differing data types.
- Let's play around with a dataframe from the `mtcars` dataset. To see a description of these data, type  
`help(mtcars)`
- We can load in these data with the `data` function  
`data("mtcars")`
- If you have a relatively small data set, and you just want a cursory look at the data, printing the data frame in the Console (my just typing `mtcars`) may suffice. However, if you have many columns and many rows, viewing your data in the console will be very difficult.
- Instead, we can take a look at the data in a nice table in a new RStudio tab using the `View()` function.
- `View()` takes the name of a data frame as an argument.
- Please note, `view()` is incorrect. The V must be capitalized.  
`View(mtcars)`
- More than likely, when you receive data to work with, it will be in the form of a data frame.
- So once you have a data frame, how can you examine individual columns?
- The syntax is very simple. To refer to one column simply type the name of the data frame and the name of the column separated by a `$`. For example:

```
mtcars$mpg #calls the mpg column of our data frame
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
```

- One nice thing about Rstudio is that it has a suggestion feature.
- If you've saved a data frame, when you type its name and the dollar sign, a dropdown with all of the possible columns should appear for you.
- If the dropdown does not appear, try pressing tab.
- Maybe you just want all of the column names displayed for you.
- For that you can use the `names()` function.
- `names()` takes a data frame as input and outputs a vector comprised of that data frame's column names.

```
names(mtcars)
```

```
## [1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"
## [11] "carb"
```

- Now let's use what we know about indexing to rename the first column.
- We know that `names(mtcars)` represents a vector.
- So let's refer to the first element of that vector and set it equal to something new.

```
names(mtcars)[1] <- "mpg2" #rename the first column
names(mtcars) #display the new names vector
```

```
## [1] "mpg2" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"
## [11] "carb"
```

- Here are some more useful functions for data frames:

```
head(mtcars, 15) #see the first 15 rows of the data frame
```

```
##           mpg2 cyl  disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46 0  1    4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02 0  1    4    4
## Datsun 710      22.8   4 108.0  93 3.85 2.320 18.61 1  1    4    1
## Hornet 4 Drive  21.4   6 258.0 110 3.08 3.215 19.44 1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02 0  0    3    2
## Valiant         18.1   6 225.0 105 2.76 3.460 20.22 1  0    3    1
## Duster 360      14.3   8 360.0 245 3.21 3.570 15.84 0  0    3    4
## Merc 240D        24.4   4 146.7  62 3.69 3.190 20.00 1  0    4    2
## Merc 230         22.8   4 140.8  95 3.92 3.150 22.90 1  0    4    2
## Merc 280         19.2   6 167.6 123 3.92 3.440 18.30 1  0    4    4
## Merc 280C        17.8   6 167.6 123 3.92 3.440 18.90 1  0    4    4
## Merc 450SE       16.4   8 275.8 180 3.07 4.070 17.40 0  0    3    3
## Merc 450SL       17.3   8 275.8 180 3.07 3.730 17.60 0  0    3    3
## Merc 450SLC      15.2   8 275.8 180 3.07 3.780 18.00 0  0    3    3
## Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98 0  0    3    4
```

```
tail(mtcars, 9) #see the last 9 rows of the data frames
```

```
##           mpg2 cyl  disp  hp drat   wt  qsec vs am gear carb
## Camaro Z28     13.3   8 350.0 245 3.73 3.840 15.41 0  0    3    4
```

```
## Pontiac Firebird 19.2  8 400.0 175 3.08 3.845 17.05 0 0  3  2
## Fiat X1-9        27.3  4  79.0  66 4.08 1.935 18.90 1 1  4  1
## Porsche 914-2    26.0  4 120.3  91 4.43 2.140 16.70 0 1  5  2
## Lotus Europa     30.4  4  95.1 113 3.77 1.513 16.90 1 1  5  2
## Ford Pantera L   15.8  8 351.0 264 4.22 3.170 14.50 0 1  5  4
## Ferrari Dino     19.7  6 145.0 175 3.62 2.770 15.50 0 1  5  6
## Maserati Bora    15.0  8 301.0 335 3.54 3.570 14.60 0 1  5  8
## Volvo 142E       21.4  4 121.0 109 4.11 2.780 18.60 1 1  4  2
```

## Tibbles

- The tidyverse uses tibbles more often than data frames.
- **Tibbles are mostly the same as data frames** with a few small exceptions:
  1. Better printing to console.
  2. Better interactions with strings.
- You can convert a data frame to a tibble with

```
suppressPackageStartupMessages(library(tidyverse))
mtcars <- as_tibble(mtcars)
mtcars
```

```
## # A tibble: 32 x 11
##   mpg2    cyl  disp    hp  drat    wt   qsec    vs    am  gear  carb
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  21      6  160    110  3.9   2.62  16.5    0     1     4     4
## 2  21      6  160    110  3.9   2.88  17.0    0     1     4     4
## 3 22.8     4  108     93  3.85  2.32  18.6    1     1     4     1
## 4 21.4     6  258    110  3.08  3.22  19.4    1     0     3     1
## 5 18.7     8  360    175  3.15  3.44  17.0    0     0     3     2
## 6 18.1     6  225    105  2.76  3.46  20.2    1     0     3     1
## 7 14.3     8  360    245  3.21  3.57  15.8    0     0     3     4
## 8 24.4     4  147.     62  3.69  3.19  20      1     0     4     2
## 9 22.8     4  141.     95  3.92  3.15  22.9    1     0     4     2
##10 19.2     6  168.    123  3.92  3.44  18.3    1     0     4     4
## # ... with 22 more rows
```

- **Exercise:** Extract the 8th to 28th elements of the `am` variable from the `mtcars` data frame.