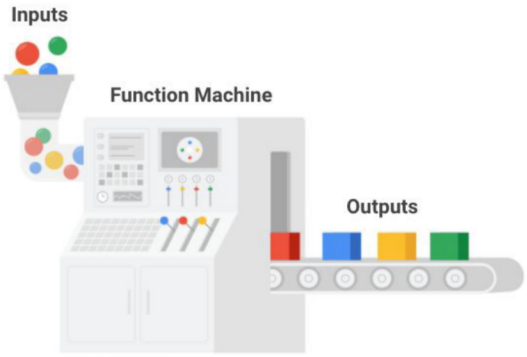
Data import

In this reading, you will learn the basics of importing data into R. It comes with built-in datasets that are great tools for learning how to use R and practice analyzing data. You will explore the **data()** function and learn how to load sample datasets into RStudio. Then you will go through how to use two tidyverse packages—readr and readxl—to import files from other sources into R. You will learn how to use readr to read a .csv file, and how to use readxl to read a .xlsx file.

The data() function

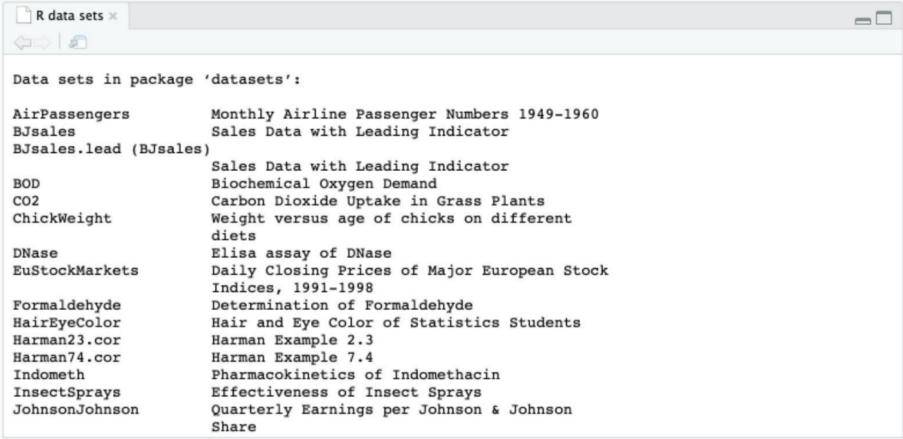


The default installation of R comes with a number of preloaded datasets that you can practice with. This is a great way to develop your R skills and learn about some important data analysis functions. Plus, many online resources and tutorials use these sample datasets to teach coding concepts in R.

You can use the **data()** function to load these datasets in R. If you run the data function without an argument, R will display a list of the available datasets.

```
data()
```

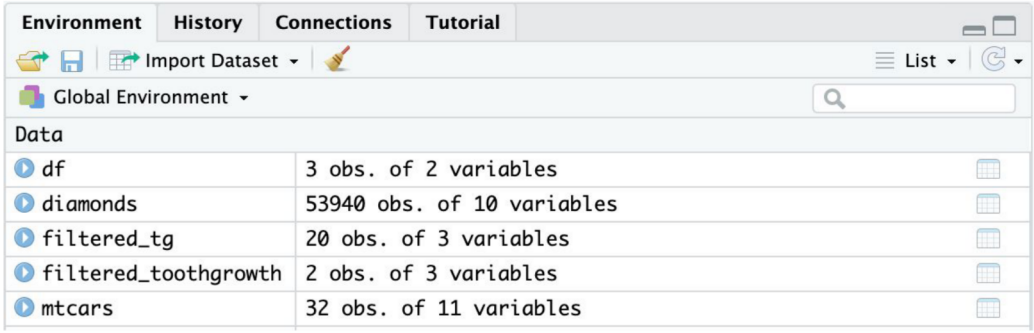
This includes the list of preloaded datasets from the “datasets” package.



If you want to load a specific dataset, just enter its name in the parentheses of the **data()** function. For example, let’s load the “mtcars” dataset, which has information about cars that have been featured in past issues of *Motor Trend* magazine.

```
data(mtcars)
```

When you run the function, R will load the dataset. The dataset will also appear in the “environment” pane of your RStudio. The environment pane displays the names of the data objects, like data frames and variables, that you have in your current workspace. In this image, “mtcars” appears in the fifth row of the pane. R tells us that it contains 32 observations and 11 variables.



Now that the dataset is loaded, you can get a preview of it in the R console pane. Just type its name...

```
mtcars
```

...and then press ctrl (or cmnd) and enter.

	mpg	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	18.62	1	0	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	0	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 250D	24.4	4	146.7	62	3.69	3.190	20.00	0	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	0	0	4	2
Merc 280	19.2	6	167.6	123	3.2	3.440	18.30	0	0	4	4

You can also display the dataset by clicking directly on the name of the dataset in the environment pane. So, if you click on **mtcars** in the environment pane, R automatically runs the View() function and displays the dataset in the RStudio data viewer.

mtcars

Filter

	mpg	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

Try experimenting with other datasets in the list if you want some more practice.

readr

In addition to using R's built-in datasets, it is also helpful to import data from other sources to use for practice or analysis. The readr package in R is a great tool for reading rectangular data. Rectangular data is data that fits nicely inside a rectangle of rows and columns, with each column referring to a single variable and each row referring to a single observation.

Here are some examples of file types that store rectangular data:

- **.csv (comma separated values)**: a .csv file is a plain text file that contains a list of data. They mostly use commas to separate (or delimit) data, but sometimes they use other characters, like semicolons.
- **.tsv (tab separated values)**: a .tsv file stores a data table in which the columns of data are separated by tabs. For example, a database table or spreadsheet data.
- **.fwf (fixed width files)**: a .fwf file has a specific format that allows for the saving of textual data in an organized fashion.
- **.log:** a .log file is a computer-generated file that records events from operating systems and other software programs.

Base R also has functions for reading files, but the equivalent functions in readr are typically *much* faster. They also produce tibbles, which are easy to use and read.

The readr package is part of the core tidyverse. So, if you've already installed the tidyverse, you have what you need to start

working with readr. If not, you can install the tidyverse now.

readr functions

The goal of readr is to provide a fast and friendly way to read rectangular data. readr supports several `read_` functions. Each function refers to a specific file format.

- `read_csv()`: comma separated (CSV) files
- `read_tsv()`: tab separated files
- `read_delim()`: general delimited files
- `read_fwf()`: fixed width files
- `read_table()`: tabular files where columns are separated by white-space
- `read_log()`: web log files

These functions all have similar syntax, so once you learn how to use one of them, you can apply your knowledge to the others. This reading will focus on the `read_csv()` function, since .csv files are one of the most common forms of data storage and you will work with them frequently.

In most cases, these functions will work automatically: you supply the path to a file, run the function, and you get a tibble that displays the data in the file. Behind the scenes, readr parses the overall file and specifies how each column should be converted from a character vector to the most appropriate data type.

Read a csv file with readr

The readr package comes with some sample files from built-in datasets that you can use for example code. To list the sample files, you can run the readr_example() function with no arguments.

```
readr_example()
```

```
[1] "challenge.csv"     "epa78.txt"         "example.log"
[4] "fwf-sample.txt"    "massey-rating.txt" "mtcars.csv"
[7] "mtcars.csv.bz2"    "mtcars.csv.zip"
```

The “mtcars.csv” file refers to the “mtcars” dataset that was mentioned earlier. Let’s use the **read_csv()** function to read the “mtcars.csv” file, as an example. In the parentheses, you need to supply the path to the file. In this case, it’s “readr_example(“mtcars.csv”).

```
read_csv(readr_example("mtcars.csv"))
```

When you run the function, R prints out a column specification that gives the name and type of each column.

```
── ♦■✓♦●○ ◆□♦◆➔○➔◆◆❖➔■○ ────────────────────────────────────────────────────────
cols(
  mpg = col_double(),
  cyl = col_double(),
  disp = col_double(),
  hp = col_double(),
  drat = col_double(),
  wt = col_double(),
  qsec = col_double(),
  vs = col_double(),
  am = col_double(),
  gear = col_double(),
  carb = col_double()
)
```

R also prints a tibble.

```
# A tibble: 32 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦ ♦♦◆○♦
 1  21       6  160    110  3.9   2.62  16.5     0     1     4     4
```

```
 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
```

readxl

To import spreadsheet data into R, you can use the readxl package. The readxl package makes it easy to transfer data from Excel into R. Readxl supports both the legacy `.xls` file format and the modern xml-based `.xlsx` file format.

The readxl package is part of the tidyverse but is not a *core* tidyverse package, so you need to load readxl in R by using the library() function.

```
library(readxl)
```

Read a .csv file with readxl

Like the readr package, readxl comes with some sample files from built-in datasets that you can use for practice. You can run the code `readxl_example()` to see the list.

You can use the **read_excel()** function to read a spreadsheet file just like you used read_csv() function to read a .csv file. The code for reading the example file "type-me.xlsx" includes the path to the file in the parentheses of the function.

```
read_excel(readxl_example("type-me.xlsx"))
```

You can use the `excel_sheets()` function to list the names of the individual sheets.

```
excel_sheets(readxl_example("type-me.xlsx"))
```

```
[1] "logical_coercion" "numeric_coercion" "date_coercion" "text_coercion"
```

You can use the excel_sheets() function to list the names of the individual sheets.

```
excel_sheets(readxl_example("type-me.xlsx"))
```

```
[1] "logical_coercion" "numeric_coercion" "date_coercion" "text_coercion"
```

You can also specify a sheet by name or number. Just type "sheet =" followed by the name or number of the sheet. For example, you can use the sheet named "numeric_coercion" from the list above.

```
read_excel(readxl_example("type-me.xlsx"), sheet = "numeric_coercion")
```

When you run the function, R returns a tibble of the sheet.

```
# A tibble: 7 x 2
  `maybe numeric?` explanation
  <chr>            <chr>
1 NA               "empty"
2 TRUE             "boolean true"
3 FALSE            "boolean false"
4 40534            "datetime"
5 123456           "the string \"123456\""
6 123456           "the number 123456"
7 cabbage          "\"cabbage\""
```

Resources

- If you want to learn how to use readr functions to work with more complex files, check out the "Data Import" chapter in R for Data Science. It explores some of the common issues you might encounter when reading files, and how to use readr to manage those issues.
- The readxl entry in the tidyverse documentation gives a good overview of the basic functions in readxl, provides a detailed explanation of how the package operates and the coding concepts behind them, and offers links to other useful resources.

- The R “datasets” package contains lots of useful preloaded datasets. Check out the list of datasets on the R Datasets Package website. The list includes links to detailed descriptions of each dataset.