babynames

✓ Loading babynames

Before we begin, let's learn a little about our data. The `babynames` dataset comes in the **babynames** package. The package is pre-installed for you, just as ggplot2 was pre-installed in the last tutorial. But unlike in the last tutorial, I have not pre-loaded babynames, or any other package.

What does this mean? In R, whenever you want to use a package that is not part of base R, you need to load the package with the command `library()`. Until you load a package, R will not be able to find the datasets and functions contained in the package. For example, if we asked R to display the `babynames` dataset, which comes in the babynames package, right now, we'd get the message below. R cannot find the dataset because we haven't loaded the babynames package.

```
## Error in eval(expr, envir, enclos): object 'babynames' not found
```

To load the `babynames` package, you would run the command `library(babynames)`. After you load a package, R will be able to find its contents *until you close R*. The next time you open R, you will need to reload the package if you wish to use it again.

This might sound like an inconvenience, but choosing which packages to load keeps your R experience simple and orderly.

In the chunk below, load babynames (the package) and then open the help page for `babynames` (the data set). Be sure to read the help page before going on.

Code | Start Over | Solution | Run Code

```
1 library(babynames)
2 ?babynames
3
4
```

```
babynames              package:babynames              R Documentation

_B_a_b_y _n_a_m_e_s.

_D_e_s_c_r_i_p_t_i_o_n:

     Full baby name data provided by the SSA. This includes all names
     with at least 5 uses.

_U_s_a_g_e:

     babynames

_F_o_r_m_a_t:

     A data frame with five variables: 'year', 'sex', 'name', 'n' and
     'prop' ('n' divided by total number of applicants in that year,
     which means proportions are of people of that gender with that
     name born in that year).
```

The data

Now that you know a little about the dataset, let's examine its contents. If you were to run `babynames` at your R console, you would get output that looks like this:

```
babynames

#> 187     1880   F       Christina    65 6.659495e-04
#> 188     1880   F           Lelia    65 6.659495e-04
#> 189     1880   F           Nelle    65 6.659495e-04
#> 190     1880   F             Sue    65 6.659495e-04
#> 191     1880   F         Johanna    64 6.557041e-04
#> 192     1880   F           Lilly    64 6.557041e-04
#> 193     1880   F         Lucinda    63 6.454587e-04
#> 194     1880   F         Minerva    63 6.454587e-04
#> 195     1880   F          Lettie    62 6.352134e-04
#> 196     1880   F           Roxie    62 6.352134e-04
#> 197     1880   F         Cynthia    61 6.249680e-04
#> 198     1880   F          Helena    60 6.147226e-04
#> 199     1880   F           Hilda    60 6.147226e-04
#> 200     1880   F           Hulda    60 6.147226e-04
#>  [ reached getOption("max.print") -- omitted 1825233 rows ]
```

Yikes. What is happening?

Displaying large data

`babynames` is a large data frame, and R is not well equiped to display the contents of large data frames. R shows as many rows as possible before your memory buffer is overwhelmed. At that point, R stops, leaving you to look at an arbitrary section of your data.

You can avoid this behaviour by transforming your data frame to a tibble.

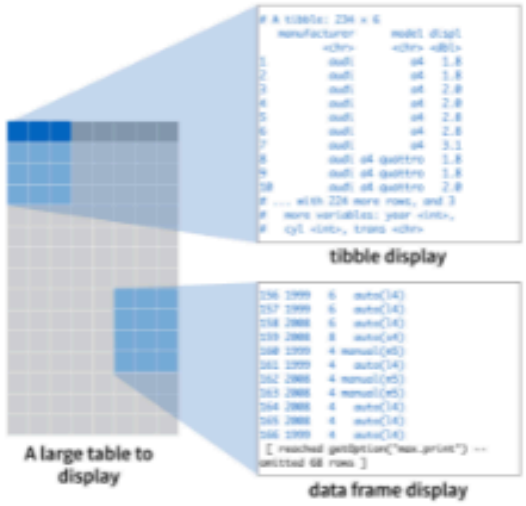
tibbles

✓ What is a tibble?

A tibble is a special type of table. R displays tibbles in a refined way whenever you have the **tibble** package loaded: R will print only the first ten rows of a tibble as well as all of the columns that fit into your console window. R also adds useful summary information about the tibble, such as the data types of each column and the size of the data set.

Whenever you do not have the tibble packages loaded, R will display the tibble as if it were a data frame. In fact, tibbles *are* data frames, an enhanced type of data frame.

You can think of the difference between the data frame display and the tibble display like this:



✓ as_tibble()

You can transform a data frame to a tibble with the `as_tibble()` function in the tibble package, e.g. `as_tibble(cars)`. However, `babynames` is already a tibble. To display it nicely, you just need to load the tibble package.

To see what I mean, use `library()` to load the tibble package in the chunk below and then call `babynames`.

Code | Start Over | Solution | Run Code | Submit Answer

```
1 library(tibble)
2 babynames
3
4
```

```
# A tibble: 1,924,665 x 5
    year sex   name          n   prop
   <dbl> <chr> <chr>     <int>  <dbl>
 1  1880 F     Mary       7065 0.0724
 2  1880 F     Anna       2604 0.0267
 3  1880 F     Emma       2003 0.0205
 4  1880 F     Elizabeth  1939 0.0199
 5  1880 F     Minnie     1746 0.0179
 6  1880 F     Margaret   1578 0.0162
 7  1880 F     Ida        1472 0.0151
 8  1880 F     Alice      1414 0.0145
 9  1880 F     Bertha     1320 0.0135
10  1880 F     Sarah      1288 0.0132
# … with 1,924,655 more rows
```

You do not need to worry much about tibbles in these tutorials; in future tutorials, I'll automatically convert each data frame into an interactive table. However, you should consider making tibbles an important part of your work in R.

tidyverse

✓ The tidyverse

The tibble package is one of several packages that are known collectively as "the tidyverse". Tidyverse packages share a common philosophy and are designed to work well together. For example, in this tutorial you will use the **tibble** package, the **ggplot2** package, and the **dplyr** package, all of which belong to the tidyverse.

✓ The tidyverse package

When you use tidyverse packages, you can make your life easier by using the **tidyverse** package. The tidyverse package provides a shortcut for installing and loading the entire suite of packages in "the tidyverse", e.g.

✓ Installing the tidyverse

Think of the **tidyverse** package as a placeholder for the packages that are in the "tidyverse". By itself, tidyverse does not do much, but when you install the tidyverse package it instructs R to install every other package in the tidyverse at the same time. In other words, when you run `install.packages("tidyverse")`, R installs the following packages for you in one simple step:

- ggplot2
- dplyr
- tidyr
- readr
- purrr
- tibble
- hms
- stringr
- lubridate
- forcats
- DBI
- haven
- jsonlite
- readxl
- rvest
- xml2
- modelr
- broom

✓ loading the tidyverse

When you load tidyverse with `library("tidyverse")`, it instructs R to load *the most commonly used* tidyverse packages. These are:

- ggplot2
- dplyr
- tidyr
- readr
- purrr
- tibble

You can load the less commonly used tidyverse packages in the normal way, by running `library(<PACKAGE NAME>)` for each of them.

Let's give this a try. We will use the ggplot2 and dplyr packages later in this tutorial. Let's use the tidyverse package to load them in the chunk below:

Let's give this a try. We will use the ggplot2 and dplyr packages later in this tutorial. Let's use the tidyverse package to load them in the chunk below:

Code | Start Over | Solution | Run Code

```
1 library("tidyverse")
2
3
```

✓ Quiz

Which package is not loaded by `library("tidyverse")`

- ggplot2 ✗
- dplyr ✗
- tibble ✗
- babynames ✓

Correct!

Now that you are familiar with the data set, and have loaded the necessary packages, let's explore the data.

Recap

Tibbles and the tidyverse package are two tools that make life with R easier. Ironically, you may not come to appreciate their value right away: these tutorials pre-load packages for you, and they wrap data frames into an interactive table for display (at least the tutorials in the primers that follow will). However, you will want to utilize tibbles and the tidyverse package when you move out of the tutorials and begin doing your own work with R inside of the RStudio IDE.

This tutorial also introduced the babynames dataset. In the next tutorial, you will use this data set to plot the popularity of *your* name over time. Along the way, you will learn how to filter and subset data sets in R.