Reading Data

July 16, 2023 V4

Dataframes and Tibbles are frequently employed data structures in R for storing and manipulating data. They facilitate the organization, exploration, and analysis of data.

Dataframes

1. A dataframe is a two-dimensional table-like data structure in R that stores data in rows and columns, with distinct data types for each column.
2. Similar to a spreadsheet or a SQL table, it is one of the most frequently employed data structures in R. Each column in a data.frame is a constant-length vector, and each row represents an observation or case.
3. Using the `data.frame()` function or by importing data from external sources such as CSV files, Excel spreadsheets, or databases, dataframe objects can be created in R.
4. dataframe objects have many useful built-in methods and functions for manipulating and summarizing data, including subsetting, merging, filtering, and aggregation. [1]

Creating a dataframe using raw data

5. The following code generates a data.frame named `df` containing three columns - `names`, `ages`, and `heights`, and four rows of data for each individual.

```
# Create input data as vectors
names <- c("Ashok", "Bullu", "Charu", "Divya")
ages <- c(72, 49, 46, 42)
heights <- c(170, 167, 160, 166)

# Combine input data into a data.frame
people <- data.frame(Name = names, Age = ages, Height = heights)

# Print the resulting dataframe
```

```r
print(people)
```

```
   Name Age Height
1 Ashok  72    170
2 Bullu  49    167
3 Charu  46    160
4 Divya  42    166
```

Reading Inbuilt datasets in R

1. R contains a number of built-in datasets that can be accessed without downloading or integrating from external sources. Here are some of the most frequently used built-in datasets in R:

- `women`: This dataset includes the heights and weights of a sample of 15,000 women.
- `mtcars`: This dataset contains information on 32 distinct automobile models, including the number of cylinders, engine displacement, horsepower, and weight.
- `diamonds`: This dataset includes the prices and characteristics of approximately 54,000 diamonds, including carat weight, cut, color, and clarity.
- `iris`: This data set measures the sepal length, sepal width, petal length, and petal breadth of 150 iris flowers from three distinct species.

The `women` dataset

As an illustration, consider the `women` dataset inbuilt in R, which contains information about the heights and weights of women. It has just two variables:

1. `height`: Height of each woman in inches
2. `weight`: Weight of each woman in pounds
3. The `data()` function is used to import any inbuilt dataset into R. The `data(women)` command in R loads the `women` dataset

```r
data(women)
```

4. The `str()` function gives the dimensions and data types and also previews the data.

```r
str(women)
```

```
'data.frame':   15 obs. of  2 variables:
 $ height: num  58 59 60 61 62 63 64 65 66 67 ...
 $ weight: num  115 117 120 123 126 129 132 135 139 142 ...
```

5. The `summary()` function gives some summary statistics.

```r
summary(women)
```

```
     height          weight
 Min.   :58.0   Min.   :115.0
 1st Qu.:61.5   1st Qu.:124.5
 Median :65.0   Median :135.0
 Mean   :65.0   Mean   :136.7
 3rd Qu.:68.5   3rd Qu.:148.0
 Max.   :72.0   Max.   :164.0
```

The `mtcars` dataset

The `mtcars` dataset inbuilt in R comprises data on the fuel consumption and other characteristics of 32 different automobile models. Here is a concise description of the 11 `mtcars` data columns:

1. `mpg`: Miles per gallon (fuel efficiency)
2. `cyl`: Number of cylinders
3. `disp`: Displacement of the engine (in cubic inches)
4. `hp`: gross horsepower
5. `drat`: Back axle ratio wt: Weight (in thousands of pounds)
6. `wt`: Weight (in thousands of pounds)
7. `qsec`: 1/4 mile speed (in seconds)
8. `vs`: Type of engine (0 = V-shaped, 1 = straight)
9. `am`: Type of transmission (0 for automatic, 1 for manual)
10. `gear`: the number of forward gears
11. `carb`: the number of carburetors

```r
data(mtcars)
str(mtcars)
```

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Reading different file formats into a dataframe

1. We examine how to read data into a dataframe in R when the original data is stored in prominent file formats such as CSV, Excel, and Google Sheets.
2. Before learning how to accomplish this, it is necessary to comprehend how to configure the Working Directory in R.

Working Directory

1. The working directory is the location where R searches for and saves files by default.
2. By default, when we execute a script or import data into R, R will search the working directory for files.
3. Using R's `getwd()` function, we can examine our current working directory:

```r
getwd()
```

```
[1] "/cloud/project"
```

4. We are running R in the Cloud and hence we are seeing that the working directory is specified as `/cloud/project/DataAnalyticsBook101`. If we are doing R programming on a local computer, and if our working directory is the Desktop, then we may see a different response such as `C:/Users/YourUserName/Desktop`.
5. Using R's `setwd()` function, we can change our current working directory. For example, the following code will set our working directory to the Desktop:

```
setwd("C:/Users/YourUserName/Desktop")
```

6. We should choose an easily-remembered and accessible working directory to store our R scripts and data files. Additionally, we should avoid using spaces, special characters, and non-ASCII characters in file paths, as these can cause file handling issues in R. [2]

Reading a CSV file into a dataframe

1. CSV is the abbreviation for "Comma-Separated Values." A CSV file is a plain text file that stores structured tabular data.
2. Each entry in a CSV file represents a record, whereas each column represents a field. The elements in each record are separated by commas (hence the name Comma-Separated Values), semicolons, or tabs.
3. Before proceeding ahead, it is imperative that the file that we wish to read is located in the Working Directory.
4. Suppose we wish to import a CSV file named `mtcars.csv`, located in the Working Directory. We can use the `read.csv()` function, illustrated as follows.

```
df_csv <- read.csv("mtcars.csv")
```

4. In this example, the `read.csv()` function reads the mtcars.csv file into a data frame named `df_csv`.
5. If the file is not in the current working directory, the complete file path must be specified in the `read.csv()` function argument; otherwise, an error will occur.

Reading an Excel (xlsx) file into a dataframe

1. Suppose we wish to import a Microsoft Excel file named `mtcars.xlsx`, located in the Working Directory.
2. We can use the `read_excel` function in the R package `readxl`, illustrated as follows.

```
library(readxl)
df_xlsx <- read_excel("mtcars.xlsx")
```

Reading a Google Sheet into a dataframe

1. Google Sheets is a ubiquitous cloud-based spreadsheet application developed by Google. It is a web-based application that enables collaborative online creation and modification of spreadsheets.
2. We can import data from a Google Sheet into a R dataframe, as follows.

- Consider a Google Sheet whose preferences have been set such that anyone can view it using its URL. If this is not done, then some authentication would become necessary.
- Every Google Sheet is characterized by a unique Sheet ID, embedded within the URL. For example, consider a Google Sheet containing some financial data concerning S&P500 index shares.
- Suppose the Sheet ID is: `11ahk9uWxBkDqrhNm7qYmiTwrlSC53N1zvXYfv7ttOCM`
- We can use the function `gsheet2tbl` in package `gsheet` to read the Google Sheet into a dataframe, as demonstrated in the following code.

```
# Read S&P500 stock data present in a Google Sheet.
library(gsheet)

prefix <- "https://docs.google.com/spreadsheets/d/"
sheetID <- "11ahk9uWxBkDqrhNm7qYmiTwrlSC53N1zvXYfv7ttOCM"

# Form the URL to connect to
url500 <- paste(prefix, sheetID)

# Read the Google Sheet located at the URL into a dataframe called sp500
sp500 <- gsheet2tbl(url500)
```

```
No encoding supplied: defaulting to UTF-8.
```

- The first line imports the `gsheet` package required to access Google Sheets into R.
- The following three lines define URL variables for Google Sheets. The `prefix` variable contains the base URL for accessing Google Sheets, the `sheetID` variable contains the ID of the desired Google Sheet.
- The `paste()` function is used to combine the `prefix`, `sheetID` variables into a complete URL for accessing the Google Sheet.
- The `gsheet2tbl()` function from the `gsheet` package is then used to read the specified Google Sheet into a dataframe called `sp500`, which can then be analyzed further in R.

Joining or Merging two dataframes

- Suppose we have a second S&P 500 data located in a second Google Sheet and suppose that we would like to join or merge the data in this dataframe with the above dataframe sp500.
- The ID of this second sheet is: `1F5KvFATcehrdJuGjYVqppNYC9hEKSww9rXYHCk2g6OA`
- We can read the data present in this Google Sheet using the following code, similar to the one discussed above, using the following code.

```
# Read additional S&P500 data that is posted in a Google Sheet.
library(gsheet)

prefix <- "https://docs.google.com/spreadsheets/d/"
sheetID <- "1nm688a3GsPM5cadJIwu6zj336WBaduglY9TSTUaM9jk"

# Form the URL to connect to
url <- paste(prefix, sheetID)

# Read the Google Sheet located at the URL into a dataframe called gf
gf <- gsheet2tbl(url)
```

```
No encoding supplied: defaulting to UTF-8.
```

- We now have two dataframes named `sp500` and `gf` that we wish to merge or join.
- The two dataframes have a column named `Stock` in common, which will serve as the key, while doing the join.
- The following code illusrates how to merge two dataframes:

```
# merging dataframes
df <- merge(sp500, gf , id = "Stock")
```

- We now have a new dataframe named `df`, which contains the data got from merging the two dataframes `sp500` and `gf`.

Tibbles

1. A tibble is a contemporary and enhanced variant of a R data frame that is part of the tidyverse package collection.
2. Tibbles are created and manipulated using the dplyr package, which provides a suite of functions optimized for data manipulation.
3. The following characteristics distinguish a tibble from a conventional data frame:
4. Tibbles must always have unique, non-empty column names. Tibbles do not permit the creation or modification of columns using partial matching of column names. Tibbles improve the output of large datasets by displaying by default only a few rows and columns.
5. Tibbles have a more consistent behavior for subsetting, with the use of [[always returning a vector or NULL, and [] always returning a tibble.
6. Here is an example of using the tibble() function in dplyr to construct a tibble:

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
# Create a tibble
my_tibble <- tibble(
```

```r
  name = c("Alice", "Bob", "Charlie"),
  age = c(25, 30, 35),
  gender = c("F", "M", "M")
)

# Print the tibble
my_tibble
```

```
# A tibble: 3 x 3
  name      age gender
  <chr>   <dbl> <chr>
1 Alice      25 F
2 Bob        30 M
3 Charlie    35 M
```

7. This will generate a tibble consisting of three columns (name, age, and gender) and three rows of data. Note that the column names are preserved and the tibble is printed in a compact and legible manner.

Converting a dataframe into a tibble

```r
# Create a data frame
my_df <- data.frame(
  name = c("Alice", "Bob", "Charlie"),
  age = c(25, 30, 35),
  gender = c("F", "M", "M")
)

# Convert the data frame to a tibble
my_tibble <- as_tibble(my_df)

# Print the tibble
my_tibble
```

```
# A tibble: 3 x 3
  name      age gender
  <chr>   <dbl> <chr>
1 Alice      25 F
2 Bob        30 M
3 Charlie    35 M
```

8. This assigns the tibble representation of the data frame my_df to the variable my_tibble.
9. Note that the resulting tibble has the same column names and data as the original data frame, but has the additional characteristics and behaviors of a tibble.

Converting a tibble into a dataframe

```
library(dplyr)

# Convert the tibble to a data frame
my_df <- as.data.frame(my_tibble)

# Print the data frame
my_df
```

```
     name age gender
1   Alice  25      F
2     Bob  30      M
3 Charlie  35      M
```

10. A tibble offers several advantages over a data frame in R:

- Large datasets can be printed with greater clarity and precision using Tibbles. By default, they only print the first few rows and columns, making it simpler to read and comprehend the data structure.
- Better subsetting behavior: With [[always returning a vector or NULL and [] always returning a tibble, Tibbles have a more consistent subsetting behavior. This facilitates the subset and manipulation of data without unintended consequences.
- Consistent naming: Tibbles always have column names that are distinct and non-empty. This makes it simpler to refer to specific columns and prevents errors caused by duplicate or unnamed column names.
- More informative errors: Tibbles provides more informative error messages that make it simpler to diagnose and resolve data-related problems.
- Fewer surprises: Tibbles have more stringent constraints than data frames, resulting in fewer surprises and unexpected behavior when manipulating data.

References

[1]

R Core Team. (2021). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. **https://www.R-project.org/**

R Core Team. (2022). Vectors, Lists, and Arrays. R Documentation. https://cran.r-project.org/doc/manuals/r-release/R-intro.html#vectors-lists-and-arrays

Wickham, H., & Grolemund, G. (2016). R for data science: Import, tidy, transform, visualize, and model data. O'Reilly Media, Inc.

R Core Team. (2022, March 2). Data Frames. R Documentation. https://www.rdocumentation.org/packages/ba

[2]

OpenIntro. (2022). 1.3 RStudio and working directory. In Introductory Statistics with Randomization and Simulation (1st ed.). https://www.openintro.org/book/isrs/

R Core Team. (2021). getwd(): working directory; setwd(dir): change working directory. In R: A language and environment for statistical computing. R Foundation for Statistical Computing. https://stat.ethz.ch/R-manual/R-devel/library/base/html/getwd.html