Importing Data in R

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Import csv and excel file
- Access rows and columns from dataset
- Access R built-in dataset

Table of Contents

About the Dataset

Movies dataset

Here we have a dataset that includes one row for each movie, with several columns for each movie characteristic:

- **name** - Name of the movie
- **year** - Year the movie was released
- **length_min** - Length of the movie (minutes)
- **genre** - Genre of the movie
- **average_rating** - Average rating on IMDB
- **cost_millions** - Movie's production cost (millions in USD)

- **foreign** - Is the movie foreign (1) or domestic (0)?
- **age_restriction** - Age restriction for the movie

name	year	length_min	genre	average_rating	cost_millions	foreign	age_restriction
Toy Story	1995	81	Animation	8.3	30	0	0
Akira	1998	125	Animation	8.1	10.4	1	14
The Breakfast Club	1985	97	Drama	7.9	1	0	14
The Artist	2011	100	Romance	8	15	1	12
Modern Times	1936	87	Comedy	8.6	1.5	0	10
Fight Club	1999	139	Drama	8.9	63	0	18
City of God	2002	130	Crime	8.7	3.3	1	18
The Untouchables	1987	119	Drama	7.9	25	0	14
Star Wars	1977	121	Action	8.7	11	0	10
American Beauty	1999	122	Drama	8.4	15	0	14
Room	2015	118	Drama	8.3	13	1	14
Dr. Strangelove	1964	94	Comedy	8.5	1.8	1	10
The Ring	1998	95	Horror	7.3	1.2	1	18
Monty Python and the Holy Grail	1975	91	Comedy	8.3	0.4	1	18
High School Musical	2006	98	Comedy	5.2	4.2	0	0
Shaun of the Dead	2004	99	Horror	8	6.1	1	18
Taxi Driver	1976	113	Crime	8.3	1.3	1	14
The Shawshank Redemption	1994	142	Crime	9.3	25	0	16
Interstellar	2014	169	Adventure	8.6	165	0	10
Casino	1995	178	Biography	8.2	50	0	18
The Goodfellas	1990	145	Biography	8.7	25	0	14
Blue is the Warmest Colour	2013	179	Romance	7.8	4.5	1	18
Black Swan	2010	108	Thriller	8	13	0	16
Back to the Future	1985	116	Sci-fi	8.5	19	0	0
The Wave	2008	107	Thriller	7.6	5.5	1	16
Whiplash	2014	106	Drama	8.5	3.3	1	12
The Grand Hotel Budapest	2014	100	Crime	8.1	25.5	0	14
Jumanji	1995	104	Fantasy	6.9	65	0	12
The Eternal Sunshine of the Spotless Mind	2004	108	Drama	8.3	20	0	14
Chicago	2002	113	Comedy	7.2	45	0	12

Let's learn how to **import and read data** from two common types of files used to store tabular data (when data is stored in a table or a spreadsheet.)

- **CSV files** (.csv)
- **Excel files** (.xls or .xlsx)

To begin, we'll need to **download the data**!

Download the Data

We've made it easy for you to get the data, which we've hosted online. Simply run the code cell below (Shift + Enter) to download the data to your current folder.

In [2]:
```
# Download datasets

# CSV file
download.file("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/I
              destfile="movies-db.csv")

# XLS file
download.file("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/I
              destfile="movies-db.xls")
```

If you ran the cell above, you have now downloaded the following files to your current folder:

> movies-db.csv
> movies-db.xls

Reading CSV Files

What are CSV files?

Let's read data from a CSV file. CSV (Comma Separated Values) is one of the most common formats of structured data you will find. These files contain data in a table format, where in each row, columns are separated by a delimiter -- traditionally, a comma (hence comma-separated values).

Usually, the first line in a CSV file contains the column names for the table itself. CSV files are popular because you do not need a particular program to open it.

Reading CSV files in R

In the **`movies-db.csv`** file, the first line of text is the header (names of each of the columns), followed by rows of movie information.

To read CSV files into R, we use the core function **`read.csv`**.

`read.csv` easy to use. All you need is the filepath to the CSV file. Let's try loading the file using the filepath to the `movies-db.csv` file we downloaded earlier:

In [3]:
```
# Load the CSV table into the my_data variable.
my_data <- read.csv("movies-db.csv")
my_data
```

A data.frame: 30 × 8

name	year	length_min	genre	average_rating	cost_millions	foreign	age_restri
<fct>	<int>	<int>	<fct>	<dbl>	<dbl>	<int>	
Toy Story	1995	81	Animation	8.3	30.0	0	
Akira	1998	125	Animation	8.1	10.4	1	
The Breakfast Club	1985	97	Drama	7.9	1.0	0	
The Artist	2011	100	Romance	8.0	15.0	1	
Modern Times	1936	87	Comedy	8.6	1.5	0	
Fight Club	1999	139	Drama	8.9	63.0	0	
City of God	2002	130	Crime	8.7	3.3	1	
The Untouchables	1987	119	Drama	7.9	25.0	0	
Star Wars Episode IV	1977	121	Action	8.7	11.0	0	
American Beauty	1999	122	Drama	8.4	15.0	0	
Room	2015	118	Drama	8.3	13.0	1	
Dr. Strangelove	1964	94	Comedy	8.5	1.8	1	
The Ring	1998	95	Horror	7.3	1.2	1	
Monty Python and the Holy Grail	1975	91	Comedy	8.3	0.4	1	
High School Musical	2006	98	Comedy	5.2	4.2	0	
Shaun of the Dead	2004	99	Horror	8.0	6.1	1	
Taxi Driver	1976	113	Crime	8.3	1.3	1	
The Shawshank Redemption	1994	142	Crime	9.3	25.0	0	
Interstellar	2014	169	Adventure	8.6	165.0	0	
Casino	1995	178	Biography	8.2	50.0	0	
The Goodfellas	1990	145	Biography	8.7	25.0	0	

name	year	length_min	genre	average_rating	cost_millions	foreign	age_restri
<fct>	<int>	<int>	<fct>	<dbl>	<dbl>	<int>	
Blue is the Warmest Colour	2013	179	Romance	7.8	4.5	1	
Black Swan	2010	108	Thriller	8.0	13.0	0	
Back to the Future	1985	116	Sci-fi	8.5	19.0	0	
The Wave	2008	107	Thriller	7.6	5.5	1	
Whiplash	2014	106	Drama	8.5	3.3	1	
The Grand Hotel Budapest	2014	100	Crime	8.1	25.5	0	
Jumanji	1995	104	Fantasy	6.9	65.0	0	
The Eternal Sunshine of the Spotless Mind	2004	108	Drama	8.3	20.0	0	
Chicago	2002	113	Comedy	7.2	45.0	0	

The data was loaded into the `my_data` variable. But instead of viewing all the data at once, we can use the `head` function to take a look at only the top six rows of our table, like so:

In [4]:
```
# Print out the first six rows of my_data
head(my_data)
```

A data.frame: 6 × 8

	name	year	length_min	genre	average_rating	cost_millions	foreign	age_restric
	<fct>	<int>	<int>	<fct>	<dbl>	<dbl>	<int>	<
1	Toy Story	1995	81	Animation	8.3	30.0	0	
2	Akira	1998	125	Animation	8.1	10.4	1	
3	The Breakfast Club	1985	97	Drama	7.9	1.0	0	
4	The Artist	2011	100	Romance	8.0	15.0	1	
5	Modern Times	1936	87	Comedy	8.6	1.5	0	
6	Fight Club	1999	139	Drama	8.9	63.0	0	

Additionally, you may want to take a look at the **structure** of your newly created table. R provides us with a function that summarizes an entire table's properties, called `str`. Let's try it out.

In [5]:
```
# Prints out the structure of your table.
str(my_data)
```

```
'data.frame':   30 obs. of  8 variables:
 $ name           : Factor w/ 30 levels "Akira","American Beauty",..: 29 1 21 20 14
10 8 27 18 2 ...
 $ year           : int  1995 1998 1985 2011 1936 1999 2002 1987 1977 1999 ...
 $ length_min     : int  81 125 97 100 87 139 130 119 121 122 ...
 $ genre          : Factor w/ 12 levels "Action","Adventure",..: 3 3 7 10 5 7 6 7 1
7 ...
 $ average_rating : num  8.3 8.1 7.9 8 8.6 8.9 8.7 7.9 8.7 8.4 ...
 $ cost_millions  : num  30 10.4 1 15 1.5 63 3.3 25 11 15 ...
 $ foreign        : int  0 1 0 1 0 0 1 0 0 0 ...
 $ age_restriction: int  0 14 14 12 10 18 18 14 10 14 ...
```

When we loaded the file with the `read.csv` function, we had to only pass it one parameter -- the **path** to our desired file.

Coding Exercise: in the code cell below, get the summary of my_data data frame

In [6]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
summary(my_data)
```

```
                        name          year        length_min           genre
Akira                     : 1   Min.   :1936   Min.   : 81.00   Drama    :7
American Beauty           : 1   1st Qu.:1988   1st Qu.: 99.25   Comedy   :5
Back to the Future        : 1   Median :1998   Median :110.50   Crime    :4
Black Swan                : 1   Mean   :1996   Mean   :116.80   Animation:2
Blue is the Warmest Colour: 1   3rd Qu.:2008   3rd Qu.:124.25   Biography:2
Casino                    : 1   Max.   :2015   Max.   :179.00   Horror   :2
(Other)                   :24                                   (Other)  :8
average_rating  cost_millions        foreign    age_restriction
Min.   :5.200   Min.   :  0.400   Min.   :0.0   Min.   : 0.00
1st Qu.:7.925   1st Qu.:  3.525   1st Qu.:0.0   1st Qu.:12.00
Median :8.300   Median : 13.000   Median :0.0   Median :14.00
Mean   :8.103   Mean   : 22.300   Mean   :0.4   Mean   :12.93
3rd Qu.:8.500   3rd Qu.: 25.000   3rd Qu.:1.0   3rd Qu.:16.00
Max.   :9.300   Max.   :165.000   Max.   :1.0   Max.   :18.00
```

▶ Click here for the solution

Reading Excel Files

Reading XLS (Excel Spreadsheet) files is similar to reading CSV files, but there's one catch -- R does not have a native function to read them. However, thankfully, R has an extremely large repository of user-created functions, called *CRAN*. From there, we can download a library package to make us able to read XLS files.

To download a package, we use the `install.packages` function (may take minutes because it is a big library). Once installed, you do not need to install that same library ever again, unless, of course, you uninstall it.

Whenever you are going to use a library that is not native to R, you have to load it into the R environment after you install it. In other words, you need to install once only, but to use it, you must load it into R for every new session. To do so, use the `library` function, which loads up everything we can use in that library into R.

In [7]:
```r
# Load the "readxl" library into the R environment.
library(readxl)
```

Now that we have our library and its functions ready, we can move on to actually reading the file. In `readxl`, there is a function called `read_excel`, which does all the work for us. You can use it like this:

In [8]:
```r
# Read data from the XLS file and attribute the table to the my_excel_data variable
my_excel_data <- read_excel("movies-db.xls")
```

Since `my_excel_data` is now a dataframe in R, much like the one we created out of the CSV file, all of the native R functions can be applied to it, like `head` and `str`.

In [9]:
```
# Prints out the structure of your table.
# Tells you how many rows and columns there are, and the names and type of each col
# This should be the very same as the other table we created, as they are the same
str(my_excel_data)
```

```
tibble [30 × 8] (S3: tbl_df/tbl/data.frame)
 $ name           : chr [1:30] "Toy Story" "Akira" "The Breakfast Club" "The Artist"
...
 $ year           : num [1:30] 1995 1998 1985 2011 1936 ...
 $ length_min     : num [1:30] 81 125 97 100 87 139 130 119 121 122 ...
 $ genre          : chr [1:30] "Animation" "Animation" "Drama" "Romance" ...
 $ average_rating : num [1:30] 8.3 8.1 7.9 8 8.6 8.9 8.7 7.9 8.7 8.4 ...
 $ cost_millions  : num [1:30] 30 10.4 1 15 1.5 63 3.3 25 11 15 ...
 $ foreign        : num [1:30] 0 1 0 1 0 0 1 0 0 0 ...
 $ age_restriction: num [1:30] 0 14 14 12 10 18 18 14 10 14 ...
```

Much like the `read.csv` function, `read_excel` takes as its main parameter the **path** to the desired file.

> **[Tip]** A library is basically a collection of different classes and functions which are used to perform some specific operations. You can install and use libraries to add more functions that are not included on the core R files. For example, the readxl library adds functions to read data from excel files.
>
> It's important to know that there are many other libraries too which can be used for a variety of things. There are also plenty of other libraries to read Excel files -- readxl is just one of them.

Accessing Rows and Columns

Whenever we use functions to read tabular data in R, the default method of structuring this data in the R environment is using Data Frames -- R's primary data structure. Data Frames are extremely versatile, and R presents us many options to manipulate them.

Suppose we want to access the "name" column of our dataset. We can directly reference the column name on our data frame to retrieve this data, like this:

In [10]:
```
# Retrieve a subset of the data frame consisting of the "name" columns
my_data['name']
```

A data.frame: 30 × 1

name <fct>
Toy Story
Akira
The Breakfast Club
The Artist
Modern Times
Fight Club
City of God
The Untouchables
Star Wars Episode IV
American Beauty
Room
Dr. Strangelove
The Ring
Monty Python and the Holy Grail
High School Musical
Shaun of the Dead
Taxi Driver
The Shawshank Redemption
Interstellar
Casino
The Goodfellas
Blue is the Warmest Colour
Black Swan
Back to the Future
The Wave
Whiplash
The Grand Hotel Budapest
Jumanji
The Eternal Sunshine of the Spotless Mind

name
<fct>
Chicago

Another way to do this is by using the `$` notation which at the output will provide a vector:

In [11]:
```
# Retrieve the data for the "name" column in the data frame.
my_data$name
```

Toy Story · Akira · The Breakfast Club · The Artist · Modern Times · Fight Club · City of God · The Untouchables · Star Wars Episode IV · American Beauty · Room · Dr. Strangelove · The Ring · Monty Python and the Holy Grail · High School Musical · Shaun of the Dead · Taxi Driver · The Shawshank Redemption · Interstellar · Casino · The Goodfellas · Blue is the Warmest Colour · Black Swan · Back to the Future · The Wave · Whiplash · The Grand Hotel Budapest · Jumanji · The Eternal Sunshine of the Spotless Mind · Chicago

▶ **Levels**:

You can also do the same thing using **double square brackets**, to get a vector of `names` column.

In [12]:
```
my_data[["name"]]
```

Toy Story · Akira · The Breakfast Club · The Artist · Modern Times · Fight Club · City of God · The Untouchables · Star Wars Episode IV · American Beauty · Room · Dr. Strangelove · The Ring · Monty Python and the Holy Grail · High School Musical · Shaun of the Dead · Taxi Driver · The Shawshank Redemption · Interstellar · Casino · The Goodfellas · Blue is the Warmest Colour · Black Swan · Back to the Future · The Wave · Whiplash · The Grand Hotel Budapest · Jumanji · The Eternal Sunshine of the Spotless Mind · Chicago

▶ **Levels**:

Similarly, any particular row of the dataset can also be accessed. For example, to get the first row of the dataset with all column values, we can use:

In [13]:
```
# Retrieve the first row of the data frame.
my_data[1,]
```

A data.frame: 1 × 8

	name	year	length_min	genre	average_rating	cost_millions	foreign	age_restrictio
	<fct>	<int>	<int>	<fct>	<dbl>	<dbl>	<int>	<int
1	Toy Story	1995	81	Animation	8.3	30	0	

The first value before the comma represents the **row** of the dataset and the second value (which is blank in the above example) represents the **column** of the dataset to be retrieved. By setting the first number as 1 we say we want data from row 1. By leaving the column blank we say we want all the columns in that row.

We can specify more than one column or row by using `c`, the **concatenate** function. By using `c` to concatenate a list of elements, we tell R that we want these observations out of the data frame. Let's try it out.

In [14]:
```
# Retrieve the first row of the data frame, but only the "name" and "length_min" co
my_data[1, c("name","length_min")]
```

A data.frame: 1 × 2

	name	length_min
	<fct>	<int>
1	Toy Story	81

Accessing Built-in Datasets in R

R provides various built-in datasets for users to utilize for different purposes. To know which datasets are available, R provides a simple function -- `data` -- that returns all of the present datasets' names with a small description beside them. The ones in the `datasets` package are all inbuilt.

In [15]:
```
# Displays a list of the inbuilt datasets. Opens in a new "window".
data()
```

Data sets

A data.frame: 104 × 3

Package	Item	Title
<chr>	<chr>	<chr>
datasets	AirPassengers	Monthly Airline Passenger Numbers 1949-1960
datasets	BJsales	Sales Data with Leading Indicator
datasets	BJsales.lead (BJsales)	Sales Data with Leading Indicator
datasets	BOD	Biochemical Oxygen Demand
datasets	CO2	Carbon Dioxide Uptake in Grass Plants
datasets	ChickWeight	Weight versus age of chicks on different diets
datasets	DNase	Elisa assay of DNase
datasets	EuStockMarkets	Daily Closing Prices of Major European Stock Indices, 1991-1998
datasets	Formaldehyde	Determination of Formaldehyde
datasets	HairEyeColor	Hair and Eye Color of Statistics Students
datasets	Harman23.cor	Harman Example 2.3
datasets	Harman74.cor	Harman Example 7.4
datasets	Indometh	Pharmacokinetics of Indomethacin
datasets	InsectSprays	Effectiveness of Insect Sprays
datasets	JohnsonJohnson	Quarterly Earnings per Johnson & Johnson Share
datasets	LakeHuron	Level of Lake Huron 1875-1972
datasets	LifeCycleSavings	Intercountry Life-Cycle Savings Data
datasets	Loblolly	Growth of Loblolly pine trees
datasets	Nile	Flow of the River Nile
datasets	Orange	Growth of Orange Trees
datasets	OrchardSprays	Potency of Orchard Sprays
datasets	PlantGrowth	Results from an Experiment on Plant Growth
datasets	Puromycin	Reaction Velocity of an Enzymatic Reaction
datasets	Seatbelts	Road Casualties in Great Britain 1969-84
datasets	Theoph	Pharmacokinetics of Theophylline
datasets	Titanic	Survival of passengers on the Titanic
datasets	ToothGrowth	The Effect of Vitamin C on Tooth Growth in Guinea Pigs

Package	Item	Title
<chr>	<chr>	<chr>
datasets	UCBAdmissions	Student Admissions at UC Berkeley
datasets	UKDriverDeaths	Road Casualties in Great Britain 1969-84
datasets	UKgas	UK Quarterly Gas Consumption
datasets	USAccDeaths	Accidental Deaths in the US 1973-1978
datasets	USArrests	Violent Crime Rates by US State
datasets	USJudgeRatings	Lawyers' Ratings of State Judges in the US Superior Court
datasets	USPersonalExpenditure	Personal Expenditure Data
datasets	UScitiesD	Distances Between European Cities and Between US Cities
datasets	VADeaths	Death Rates in Virginia (1940)
datasets	WWWusage	Internet Usage per Minute
datasets	WorldPhones	The World's Telephones
datasets	ability.cov	Ability and Intelligence Tests
datasets	airmiles	Passenger Miles on Commercial US Airlines, 1937-1960
datasets	airquality	New York Air Quality Measurements
datasets	anscombe	Anscombe's Quartet of 'Identical' Simple Linear Regressions
datasets	attenu	The Joyner-Boore Attenuation Data
datasets	attitude	The Chatterjee-Price Attitude Data
datasets	austres	Quarterly Time Series of the Number of Australian Residents
datasets	beaver1 (beavers)	Body Temperature Series of Two Beavers
datasets	beaver2 (beavers)	Body Temperature Series of Two Beavers
datasets	cars	Speed and Stopping Distances of Cars
datasets	chickwts	Chicken Weights by Feed Type
datasets	co2	Mauna Loa Atmospheric CO2 Concentration
datasets	crimtab	Student's 3000 Criminals Data
datasets	discoveries	Yearly Numbers of Important Discoveries
datasets	esoph	Smoking, Alcohol and (O)esophageal Cancer
datasets	euro	Conversion Rates of Euro Currencies
datasets	euro.cross (euro)	Conversion Rates of Euro Currencies
datasets	eurodist	Distances Between European Cities and Between US Cities

Package	Item	Title
<chr>	<chr>	<chr>
datasets	faithful	Old Faithful Geyser Data
datasets	fdeaths (UKLungDeaths)	Monthly Deaths from Lung Diseases in the UK
datasets	freeny	Freeny's Revenue Data
datasets	freeny.x (freeny)	Freeny's Revenue Data
datasets	freeny.y (freeny)	Freeny's Revenue Data
datasets	infert	Infertility after Spontaneous and Induced Abortion
datasets	iris	Edgar Anderson's Iris Data
datasets	iris3	Edgar Anderson's Iris Data
datasets	islands	Areas of the World's Major Landmasses
datasets	ldeaths (UKLungDeaths)	Monthly Deaths from Lung Diseases in the UK
datasets	lh	Luteinizing Hormone in Blood Samples
datasets	longley	Longley's Economic Regression Data
datasets	lynx	Annual Canadian Lynx trappings 1821-1934
datasets	mdeaths (UKLungDeaths)	Monthly Deaths from Lung Diseases in the UK
datasets	morley	Michelson Speed of Light Data
datasets	mtcars	Motor Trend Car Road Tests
datasets	nhtemp	Average Yearly Temperatures in New Haven
datasets	nottem	Average Monthly Temperatures at Nottingham, 1920-1939
datasets	npk	Classical N, P, K Factorial Experiment
datasets	occupationalStatus	Occupational Status of Fathers and their Sons
datasets	precip	Annual Precipitation in US Cities
datasets	presidents	Quarterly Approval Ratings of US Presidents
datasets	pressure	Vapor Pressure of Mercury as a Function of Temperature
datasets	quakes	Locations of Earthquakes off Fiji
datasets	randu	Random Numbers from Congruential Generator RANDU
datasets	rivers	Lengths of Major North American Rivers
datasets	rock	Measurements on Petroleum Rock Samples
datasets	sleep	Student's Sleep Data
datasets	stack.loss (stackloss)	Brownlee's Stack Loss Plant Data

Package	Item	Title
<chr>	<chr>	<chr>
datasets	stack.x (stackloss)	Brownlee's Stack Loss Plant Data
datasets	stackloss	Brownlee's Stack Loss Plant Data
datasets	state.abb (state)	US State Facts and Figures
datasets	state.area (state)	US State Facts and Figures
datasets	state.center (state)	US State Facts and Figures
datasets	state.division (state)	US State Facts and Figures
datasets	state.name (state)	US State Facts and Figures
datasets	state.region (state)	US State Facts and Figures
datasets	state.x77 (state)	US State Facts and Figures
datasets	sunspot.month	Monthly Sunspot Data, from 1749 to "Present"
datasets	sunspot.year	Yearly Sunspot Data, 1700-1988
datasets	sunspots	Monthly Sunspot Numbers, 1749-1983
datasets	swiss	Swiss Fertility and Socioeconomic Indicators (1888) Data
datasets	treering	Yearly Treering Data, -6000-1979
datasets	trees	Girth, Height and Volume for Black Cherry Trees
datasets	uspop	Populations Recorded by the US Census
datasets	volcano	Topographic Information on Auckland's Maunga Whau Volcano
datasets	warpbreaks	The Number of Breaks in Yarn during Weaving
datasets	women	Average Heights and Weights for American Women

Use 'data(package = .packages(all.available = TRUE))' to list the data sets in all *available* packages.

As you can see, there are many different datasets already inbuilt in the R environment. Having to go through each of them to take a look at their structure and try to find out what they represent might be very tiring. Thankfully, R has documentation present for each inbuilt dataset. You can take a look at that by using the `help` function.

For example, if we want to know more about the `women` dataset, we can use the following function:

In [16]:
```
# Opens up the documentation for the inbuilt "women" dataset.
help(women)
```

Average Heights and Weights for American Women

Description

This data set gives the average heights and weights for American women aged 30–39.

Usage

```
women
```

Format

A data frame with 15 observations on 2 variables.

`[,1]`	`height`	numeric	Height (in)
`[,2]`	`weight`	numeric	Weight (lbs)

Details

The data set appears to have been taken from the American Society of Actuaries *Build and Blood Pressure Study* for some (unknown to us) earlier year.

The World Almanac notes: "The figures represent weights in ordinary indoor clothing and shoes, and heights with shoes".

Source

The World Almanac and Book of Facts, 1975.

References

McNeil, D. R. (1977) *Interactive Data Analysis*. Wiley.

Examples

```
require(graphics)
plot(women, xlab = "Height (in)", ylab = "Weight (lb)",
    main = "women data: American women aged 30-39")
```

[Package *datasets* version 3.5.1]

Since the datasets listed are inbuilt, you do not need to import or load them to use them. If you reference them by their name, R already has the data frame ready.

In [17]:
```
women
```

A data.frame: 15 × 2

height <dbl>	weight <dbl>
58	115
59	117
60	120
61	123
62	126
63	129
64	132
65	135
66	139
67	142
68	146
69	150
70	154
71	159
72	164

Coding Exercise: in the code cell below, get the CO2 dataset

In [18]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
CO2
```

A nfnGroupedData: 84 × 5

	Plant	Type	Treatment	conc	uptake
	<ord>	<fct>	<fct>	<dbl>	<dbl>
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	nonchilled	175	30.4
3	Qn1	Quebec	nonchilled	250	34.8
4	Qn1	Quebec	nonchilled	350	37.2
5	Qn1	Quebec	nonchilled	500	35.3
6	Qn1	Quebec	nonchilled	675	39.2
7	Qn1	Quebec	nonchilled	1000	39.7
8	Qn2	Quebec	nonchilled	95	13.6
9	Qn2	Quebec	nonchilled	175	27.3
10	Qn2	Quebec	nonchilled	250	37.1
11	Qn2	Quebec	nonchilled	350	41.8
12	Qn2	Quebec	nonchilled	500	40.6
13	Qn2	Quebec	nonchilled	675	41.4
14	Qn2	Quebec	nonchilled	1000	44.3
15	Qn3	Quebec	nonchilled	95	16.2
16	Qn3	Quebec	nonchilled	175	32.4
17	Qn3	Quebec	nonchilled	250	40.3
18	Qn3	Quebec	nonchilled	350	42.1
19	Qn3	Quebec	nonchilled	500	42.9
20	Qn3	Quebec	nonchilled	675	43.9
21	Qn3	Quebec	nonchilled	1000	45.5
22	Qc1	Quebec	chilled	95	14.2
23	Qc1	Quebec	chilled	175	24.1
24	Qc1	Quebec	chilled	250	30.3
25	Qc1	Quebec	chilled	350	34.6
26	Qc1	Quebec	chilled	500	32.5
27	Qc1	Quebec	chilled	675	35.4
28	Qc1	Quebec	chilled	1000	38.7
29	Qc2	Quebec	chilled	95	9.3

	Plant	Type	Treatment	conc	uptake
	<ord>	<fct>	<fct>	<dbl>	<dbl>
30	Qc2	Quebec	chilled	175	27.3
⋮	⋮	⋮	⋮	⋮	⋮
55	Mn2	Mississippi	nonchilled	675	31.1
56	Mn2	Mississippi	nonchilled	1000	31.5
57	Mn3	Mississippi	nonchilled	95	11.3
58	Mn3	Mississippi	nonchilled	175	19.4
59	Mn3	Mississippi	nonchilled	250	25.8
60	Mn3	Mississippi	nonchilled	350	27.9
61	Mn3	Mississippi	nonchilled	500	28.5
62	Mn3	Mississippi	nonchilled	675	28.1
63	Mn3	Mississippi	nonchilled	1000	27.8
64	Mc1	Mississippi	chilled	95	10.5
65	Mc1	Mississippi	chilled	175	14.9
66	Mc1	Mississippi	chilled	250	18.1
67	Mc1	Mississippi	chilled	350	18.9
68	Mc1	Mississippi	chilled	500	19.5
69	Mc1	Mississippi	chilled	675	22.2
70	Mc1	Mississippi	chilled	1000	21.9
71	Mc2	Mississippi	chilled	95	7.7
72	Mc2	Mississippi	chilled	175	11.4
73	Mc2	Mississippi	chilled	250	12.3
74	Mc2	Mississippi	chilled	350	13.0
75	Mc2	Mississippi	chilled	500	12.5
76	Mc2	Mississippi	chilled	675	13.7
77	Mc2	Mississippi	chilled	1000	14.4
78	Mc3	Mississippi	chilled	95	10.6
79	Mc3	Mississippi	chilled	175	18.0
80	Mc3	Mississippi	chilled	250	17.9
81	Mc3	Mississippi	chilled	350	17.9

	Plant	Type	Treatment	conc	uptake
	<ord>	<fct>	<fct>	<dbl>	<dbl>
82	Mc3	Mississippi	chilled	500	17.9
83	Mc3	Mississippi	chilled	675	18.9
84	Mc3	Mississippi	chilled	1000	19.9

▶ Click here for the solution

Scaling R with big data

As you learn more about R, if you are interested in exploring platforms that can help you run analyses at scale, you might want to sign up for a free account on IBM Watson Studio, which allows you to run analyses in R with two Spark executors for free.

Authors

Hi! It's Iqbal Singh and Walter Gomes, the authors of this notebook. I hope you found it easy to learn how to import data into R! Feel free to connect with us if you have any questions.

Other Contributors

Yan Luo

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-03-04	2.0	Yan	Added coding tasks