Importing Tables (part 1)

Input / Output

Gaston Sanchez

STAT 33B, Fall 2025

About

In this slides we describe various "standard" ways to import tabular data. By standard we mean using base functions such as `read.table()` and friends.

Keep in mind that there are functions from external packages such as `"readr"`, `"readxl"`, `"rvest"`, etc, that can also be used to import tables in R.

Motivation

Leia

Luke

Han

Motivation

Some "data" of three individuals from a galaxy far, far away:

- Leia is a force-sensitive woman, 150 centimeters tall, who doesn't have any robots.
- Luke is a force-sensitive man, 172 centimeters tall, who has 2 robots.
- Han is a non-force-sensitive man, 180 centimeters tall, who doesn't have any robots.

Data Table

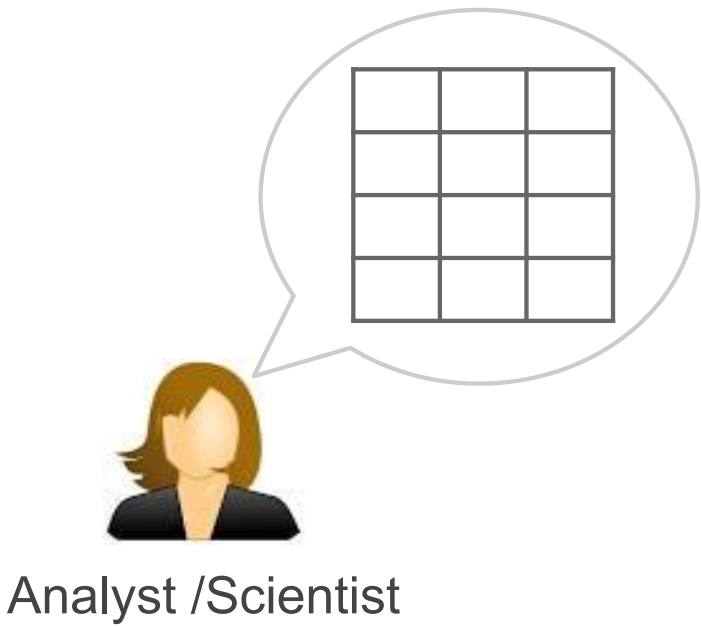
Assuming that the variables or features in our data consist of Name, Sex, Force (sensitivity), Height, and (number of) Robots, we can present this information in some sort of rectangular or tabular layout, like the one below:

Name	Sex	Force	Height	Robots
Leia	female	true	150.0	0
Luke	male	false	172.0	2
Han	male	false	180.0	0

We can say that this data set is now in tabular form, with five columns and three rows (or four rows if you include the row of column names).

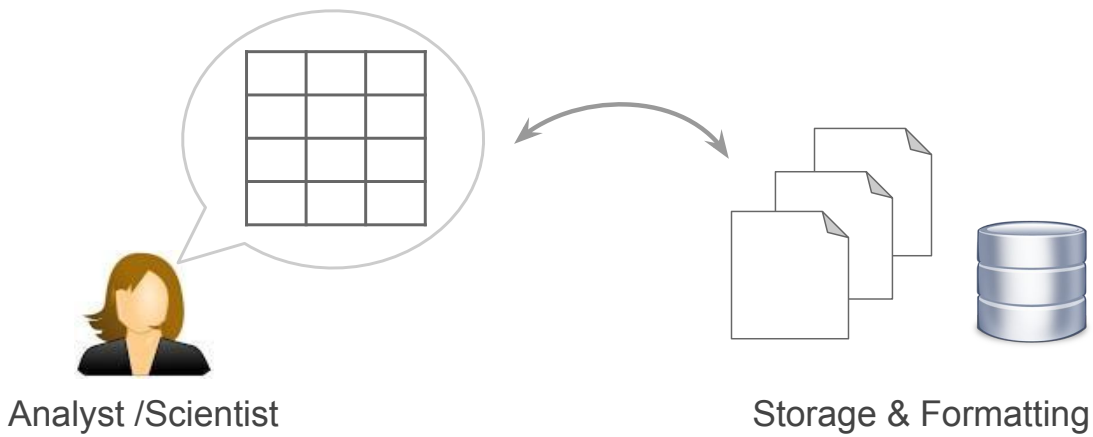
Motivation

From a statistical standpoint, perhaps the conventional thing to do is to **think** of a data table. That is, to think of the variables observed on the individuals as values organized in a table.



Data Table

To do computations with data, we need a place for the data to live in. Which means that we have to store it in one or more files, that could potentially be part of a relational data base.



Files and Formats

Files and Formats

A file format:

- is a way of interpreting the bytes in a file
- specifies how bits are used to encode information in a digital storage medium
- For example, in the simplest case, a **plain text** format means that each byte is used to represent a single character

Some Confusing Terms

- Text files
- Plain text files
- Formatted text files
- Enriched text files

Some Confusing Terms

*"Let's take the term **text files** to mean a file that consists mainly of ASCII characters . . . and that uses newline characters to give humans the perception of lines"*

Norman Matloff (2011) The Art of R Programming

Plain Text Files

- By text files we mean plain text files
- Plain text as an umbrella term for any file that is in a human-readable form (`.txt`, `.csv`, `.xml`, `.html`)
- Text files stored as a sequence of characters
- Each character stored as a single byte of data
- Data is arranged in rows, with several values stored on each row
- Text files that can be read and manipulated with a text editor

Tabular Datasets

Data Tables

A common storage option is to find a way in which we could organize the data in a form that resembles a table or rectangular format. The most common storage option to do this is by using the so-called field-delimiter formats such as comma separated values or CSV.

Character Delimited Text

- A common way to store data in tabular form is via text files
- To store the data we need a way to separate data values
- Each line represents a "row"
- The idea of "columns" is conveyed with delimiters
- In summary, fields within each line are separated by the **delimiter**
- Quotation marks are used when the delimiter character occurs within one of the fields

Plain Text Formats

There are two main subtypes of plain text format, depending on how the separated values are identified in a row

- Delimited formats
- Fixed-width formats

Delimited Formats

In a delimited format, values within a row are separated by a special character, or **delimiter**

Delimiter	Description
`" "`	white space
`","`	comma
`"\t"`	tab
`";"`	semicolon

Space Delimited

Example of a **space** delimited file (common file extension `.txt`)

```
Name Sex Force Height Robots
Leia female true 150.0 0
Luke male false 172.0 2
Han male false 180.0 0
```

Tab Delimited

Example of a **tab** delimited file (common file extensions `.txt` or `.tsv`)

```
Name    Sex Force   Height  Robots
Leia    female  true    150.0   0
Luke    male    false   172.0   2
Han male    false   180.0   0
```

Comma Delimited

Example of a **comma** delimited file (common file extension `.csv`)

```
Name,Sex,Force,Height,Robots
Leia,female,true,150.0,0
Luke,male,false,172.0,2
Han,male,false,180.0,0
```

Fixed-Width Format

Example of a **fixed width** delimited file (common file extension `.txt`)

```
Name Sex    Force Height Robots
Leia female true  150.0  0
Luke male   false 172.0  2
Han  male   false 180.0  0
```

In a fixed-width format, each value is allocated a **fixed number of characters** within every row

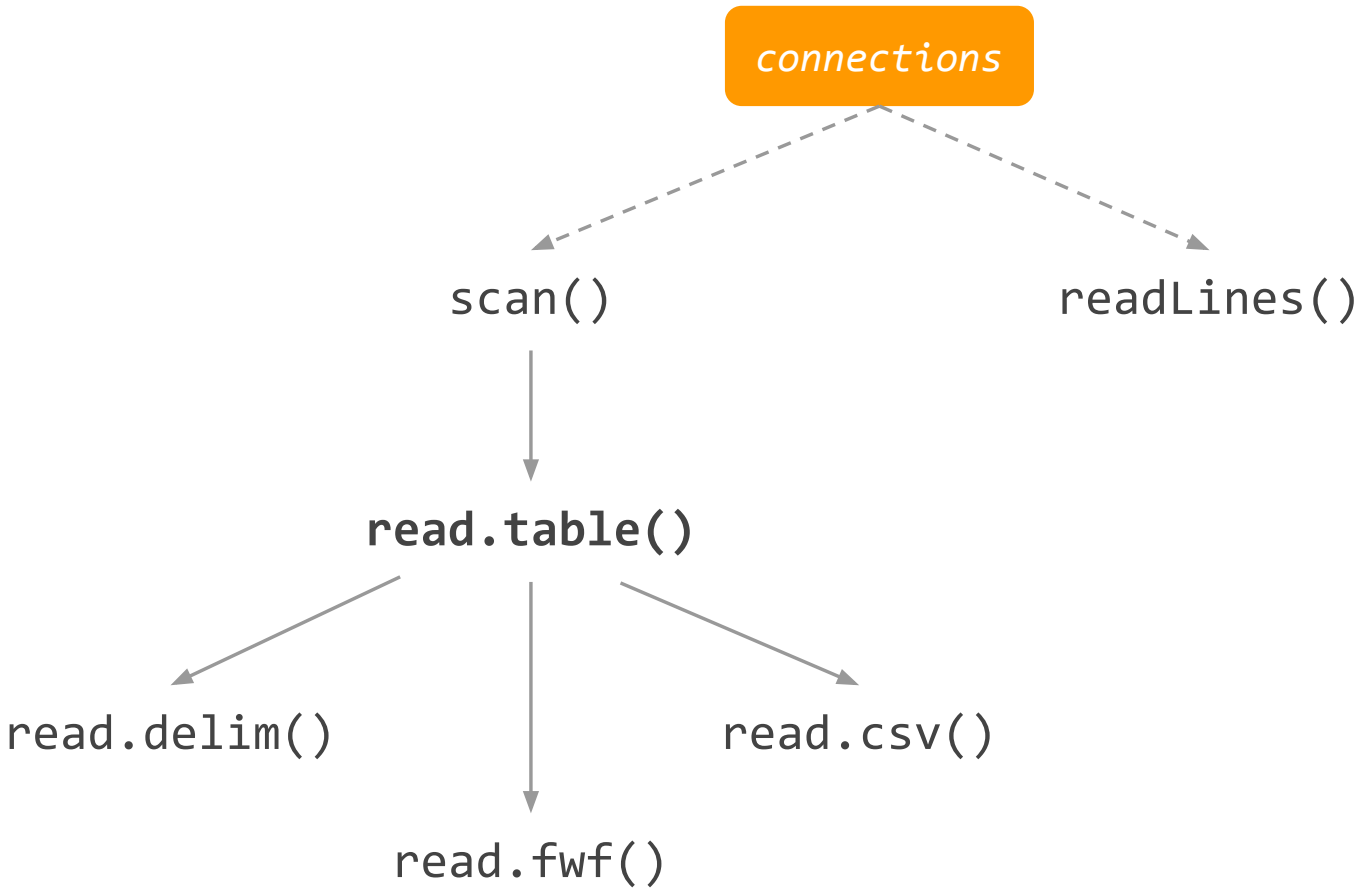
Importing Tables in R

Functions to import tables

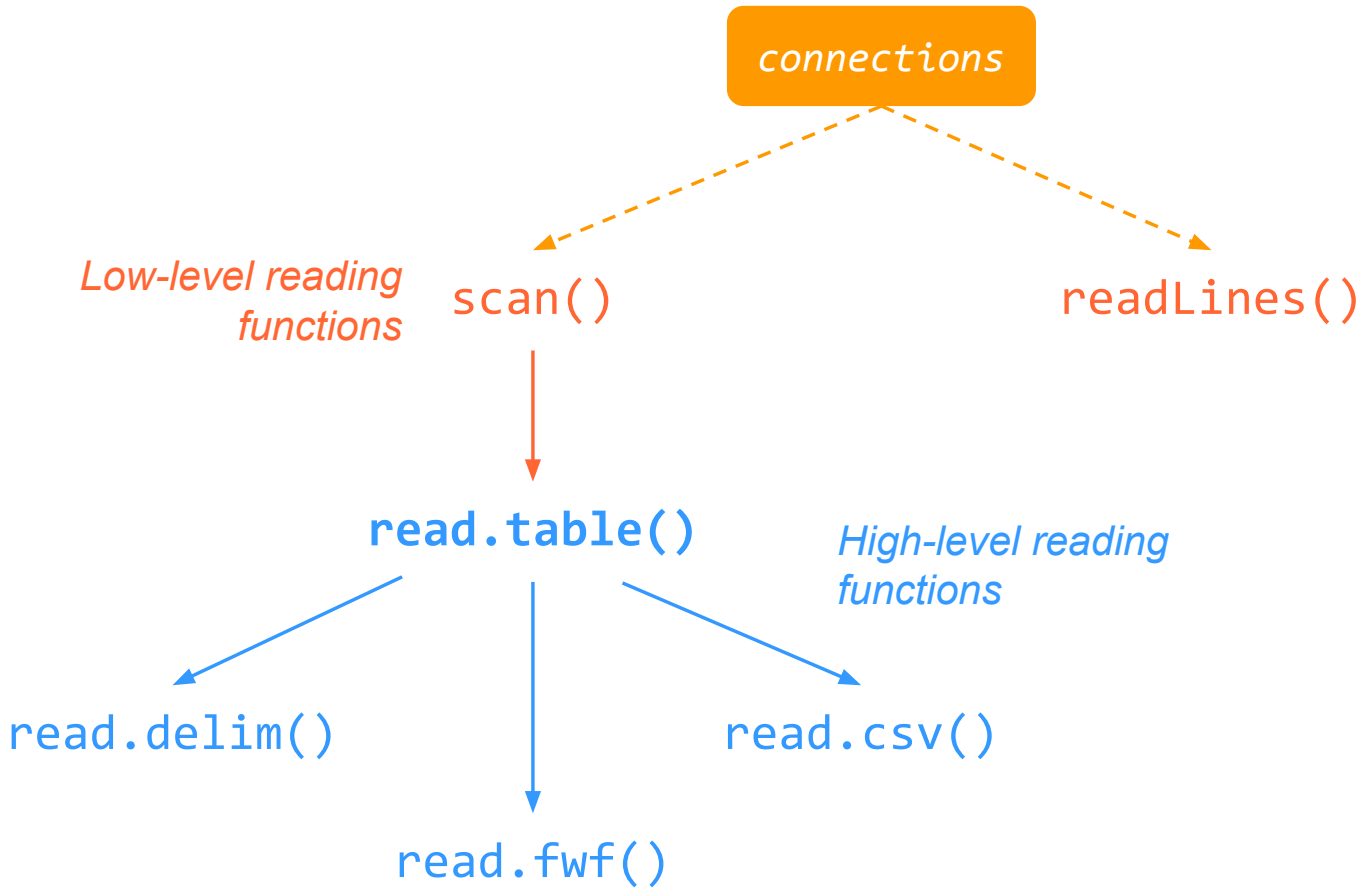
R comes with a family of functions that allows you to import most common data table formats:

Function	Description
`read.table()`	comma separated values
`read.csv()`	comma separated values
`read.csv2()`	semicolon separated values (Europe)
`read.delim()`	tab separated values
`read.delim2()`	tab separated values (Europe)
`read.fwf()`	fixed-width-format

Base R functions to read data



Base R functions to read data



R Data Import Manual

There's a wide range of ways and options to import data tables in R.

The authoritative document to know almost all about importing (and exporting) data is the manual **R Data Import/Export**.

https://cran.r-project.org/doc/manuals/r-release/R-data.html

Before importing a data table in R

- What is the character(s) used as field delimiter?
- Does the file contain names of columns?
- Does the file contain a column for row names?
- Are there any missing values?
- How are missing values codified?
- Do you want to read in all rows?

Before importing a data table in R

- Do you need to convert delimiter characters? (e.g. from space to comma)
- Can you determine the data-type of each column?
- Are there any uninformative numbers?
- Can you convert those uninformative numbers to informative labels?

Importing Data Tables in R

R Data Import/Export Manual

- There's a wide range of ways and options to import data tables in R.
- The authoritative document to know almost all about importing (and exporting) data is the manual **R Data Import/Export**

https://cran.r-project.org/doc/manuals/r-release/R-data.html

Importing Data Tables

The most common way to read and import tables in R is by using `read.table()` and friends (`read.csv()`, `read.delim()`, etc)

Function `read.table()`

Function `read.table()`

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", row.names, col.names,
           as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE,
           fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text,
           skipNul = FALSE)
```

Some `read.table()` arguments

Argument	Description
`file`	Name of file
`header`	Whether column names are in 1st line
`sep`	Character used as field separator
`quote`	Quoting characters
`dec`	Character for decimal point
`row.names`	Optional vector of row names
`col.names`	Optional vector of column names
`na.strings`	Characters treated as missing values
`colClasses`	Optional vector of data types for columns
`nrows`	Maximum number of rows to read in
`skip`	Number of lines to skip before reading data
`check.names`	Check valid column names
`stringsAsFactors`	Should characters be converted to factors

Assumptions

For simplicity's sake, we'll assume that all data files are located in your working directory.

Suppose you have the following data in a file: `starwarstoy.txt`

```
Name Sex Force Height Robots
Leia female true 150.0 0
Luke male false 172.0 2
Han male false 180.0 0
```

Importing table in blank separated file

```
# using read.table()
sw_txt <- read.table(
  file = "starwarstoy.txt",
  header = TRUE)
```

In versions of R < `4.0.0`, `read.table()` and friends convert character strings into factors by default.

Importing table in blank separated file

Limit the number of rows to read in (first 2 individuals):

```
sw_txt2 <- read.table(
  file = "starwarstoy.txt",
  header = TRUE,
  nrows = 2)
```

Importing table in blank separated file

Let's skip the first row (no header):

```
sw_txt3 <- read.table(
  file = "starwarstoy.txt",
  header = FALSE,
  skip = 1,
  nrows = 4)
```

Comma Delimited

Example of a **comma** delimited file (common file extension `.csv`)

```
Name,Sex,Force,Height,Robots
Leia,female,true,150.0,0
Luke,male,false,172.0,2
Han,male,false,180.0,0
```

Importing CSV table

Data in comma separated value (CSV) file

```
# using read.table()
sw_csv <- read.table(
  file = "starwarstoy.csv",
  header = TRUE,
  sep = ",")

# using read.csv()
sw_csv <- read.csv(file = "starwarstoy.csv")
```

Tab Delimited

Example of a **tab** delimited file (common file extensions `.txt` or `.tsv`)

```
Name    Sex Force   Height  Robots
Leia    female  true    150.0   0
Luke    male    false   172.0   2
Han male    false   180.0   0
```

Importing TSV table

Tab delimiter "\t"

```r
# using read.table()
sw_tsv <- read.table(
  file = "starwarstoy.tsv",
  header = TRUE,
  sep = "\t")

# using read.delim()
sw_tsv <- read.delim(file = "starwarstoy.tsv")
```

Other Delimiters

You could have a text file `starwarstoy.dat` with a non-standard delimiter, for example: `"%"`

```
Name%Sex%Force%Height%Robots
Leia%female%true%150.0%0
Luke%male%false%172.0%2
Han%male%false%180.0%0
```

No problem; just specify the `sep` argument:

```
# using read.table()
sw_dat <- read.table(
  file = "starwarstoy.dat",
  header = TRUE,
  sep = "%")
```

Some Considerations

Considerations

What is the field separator?

- space `" "`
- tab `"\t"`
- comma `","`
- semicolon `";"`
- other?

Considerations

Does the data file contains:

- row names?
- column names?
- missing values?
- special characters?

Considerations

So far . . .

- There are multiple ways to import data tables
- The workhorse function is `read.table()`
- But you can use the other wrappers, e.g. `read.csv()`
- The output is a `"data.frame"` object

Common files from other programs

Type	Package	Function
Excel	`"gdata"`	`read.xls()`
Excel	`"xlsx"`	`read.xlsx()`
Excel	`"readxl"`	`read_excel()`
Excel	`"XLConnect"`	`readWorksheet()`
SPSS	`"foreign"`	`read.spss()`
SAS	`"foreign"`	`read.ssd()`
SAS	`"foreign"`	`read.xport()`
Matlab	`"R.matlab"`	`readMat()`
Stata	`"foreign"`	`read.dta()`
Octave	`"foreign"`	`read.octave()`
Minitab	`"foreign"`	`read.mtp()`
Systat	`"foreign"`	`read.systat()`