Chapter 20: Reading and writing tabular data in plain-text files (CSV, TSV, etc.)

Parameter	Details
file	name of the CSV file to read
header	logical: does the .csv file contain a header row with column names?
sep	character: symbol that separates the cells on each row
quote	character: symbol used to quote character strings
dec	character: symbol used as decimal separator
fill	logical: when TRUE, rows with unequal length are filled with blank fields.
comment.char	character: character used as comment in the csv file. Lines preceded by this character are ignored.
...	extra arguments to be passed to `read.table`

Section 20.1: Importing .csv files

Importing using base R

Comma separated value files (CSVs) can be imported using `read.csv`, which wraps `read.table`, but uses `sep = ","` to set the delimiter to a comma.

```
# get the file path of a CSV included in R's utils package
csv_path <- system.file("misc", "exDIF.csv", package = "utils")

# path will vary based on installation location
csv_path
## [1] "/Library/Frameworks/R.framework/Resources/library/utils/misc/exDIF.csv"

df <- read.csv(csv_path)

df
##    Var1 Var2
## 1  2.70    A
## 2  3.14    B
## 3 10.00    A
## 4 -7.00    A
```

A user friendly option, `file.choose`, allows to browse through the directories:

```
df <- read.csv(file.choose())
```

Notes

- Unlike `read.table`, `read.csv` defaults to `header = TRUE`, and uses the first row as column names.
- All these functions will convert strings to `factor` class by default unless either `as.is = TRUE` or `stringsAsFactors = FALSE`.
- The `read.csv2` variant defaults to `sep = ";"` and `dec = ","` for use on data from countries where the comma is used as a decimal point and the semicolon as a field separator.

Importing using packages

The `readr` package's `read_csv` function offers much faster performance, a progress bar for large files, and more popular default options than standard `read.csv`, including `stringsAsFactors = FALSE`.

```r
library(readr)

df <- read_csv(csv_path)

df
## # A tibble: 4 x 2
##    Var1  Var2
##   <dbl> <chr>
## 1  2.70     A
## 2  3.14     B
## 3 10.00     A
## 4 -7.00     A
```

Section 20.2: Importing with data.table

The `data.table` package introduces the function `fread`. While it is similar to **`read.table`**, `fread` is usually faster and more flexible, guessing the file's delimiter automatically.

```r
# get the file path of a CSV included in R's utils package
csv_path <- system.file("misc", "exDIF.csv", package = "utils")

# path will vary based on R installation location
csv_path
## [1] "/Library/Frameworks/R.framework/Resources/library/utils/misc/exDIF.csv"

dt <- fread(csv_path)

dt
##     Var1 Var2
## 1:  2.70    A
## 2:  3.14    B
## 3: 10.00    A
## 4: -7.00    A
```

Where argument `input` is a string representing:

- the filename (*e.g.* `"filename.csv"`),
- a shell command that acts on a file (*e.g.* `"grep 'word' filename"`), or
- the input itself (*e.g.* `"input1, input2 \n A, B \n C, D"`).

`fread` returns an object of class `data.table` that inherits from class **`data.frame`**, suitable for use with the data.table's usage of `[ ]`. To return an ordinary data.frame, set the `data.table` parameter to `FALSE`:

```r
df <- fread(csv_path, data.table = FALSE)

class(df)
## [1] "data.frame"

df
##    Var1 Var2
## 1  2.70    A
## 2  3.14    B
## 3 10.00    A
## 4 -7.00    A
```

Notes

- `fread` does not have all same options as **`read.table`**. One missing argument is `na.comment`, which may lead

in unwanted behaviors if the source file contains #.
- `fread` uses only `"` for `quote` parameter.
- `fread` uses few (5) lines to guess variables types.

Section 20.3: Exporting .csv files

Exporting using base R

Data can be written to a CSV file using `write.csv()`:

```
write.csv(mtcars, "mtcars.csv")
```

Commonly-specified parameters include `row.names = FALSE` and `na = ""`.

Exporting using packages

`readr::write_csv` is significantly faster than `write.csv` and does not write row names.

```
library(readr)

write_csv(mtcars, "mtcars.csv")
```

Section 20.4: Import multiple csv files

```
files = list.files(pattern="*.csv")
data_list = lapply(files, read.table, header = TRUE)
```

This read every file and adds it to a list. Afterwards, if all data.frame have the same structure they can be combined into one big data.frame:

```
df <- do.call(rbind, data_list)
```

Section 20.5: Importing fixed-width files

Fixed-width files are text files in which columns are not separated by any character delimiter, like `,` or `;`, but rather have a fixed character length (*width*). Data is usually padded with white spaces.

An example:

```
Column1 Column2   Column3           Column4Column5
1647    pi        'important'       3.141596.28318
1731    euler     'quite important' 2.718285.43656
1979    answer    'The Answer.'     42     42
```

Let's assume this data table exists in the local file `constants.txt` in the working directory.

Importing with base R

```
df <- read.fwf('constants.txt', widths = c(8,10,18,7,8), header = FALSE, skip = 1)

df
#>     V1     V2                 V3         V4        V5
```

```
#> 1 1647     pi         'important'   3.14159   6.28318
#> 2 1731  euler   'quite important'   2.71828   5.43656
#> 3 1979 answer       'The Answer.'   42        42.0000
```

Note:

- Column titles don't need to be separated by a character (`Column4Column5`)
- The `widths` parameter defines the width of each column
- Non-separated headers are not readable with `read.fwf()`

Importing with readr

```
library(readr)

df <- read_fwf('constants.txt',
               fwf_cols(Year = 8, Name = 10, Importance = 18, Value = 7, Doubled = 8),
               skip = 1)
df
#> # A tibble: 3 x 5
#>    Year    Name        Importance    Value  Doubled
#>    <int>   <chr>           <chr>     <dbl>    <dbl>
#> 1  1647      pi       'important'  3.14159  6.28318
#> 2  1731   euler 'quite important'  2.71828  5.43656
#> 3  1979  answer     'The Answer.' 42.00000 42.00000
```

Note:

- readr's `fwf_*` helper functions offer alternative ways of specifying column lengths, including automatic guessing (`fwf_empty`)
- readr is faster than base R
- Column titles cannot be automatically imported from data file