

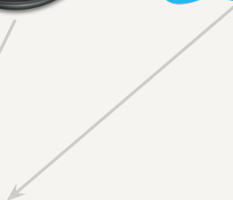
Getting Data from the Web with R

Part 2: Reading Files from the Web

Gaston Sanchez

April-May 2014

Content licensed under [CC BY-NC-SA 4.0](#)



Readme

License:

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
<http://creativecommons.org/licenses/by-nc-sa/4.0/>

You are free to:

- Share** — copy and redistribute the material
- Adapt** — rebuild and transform the material

Under the following conditions:

- Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made.
- NonCommercial** — You may not use this work for commercial purposes.
- Share Alike** — If you remix, transform, or build upon this work, you must distribute your contributions under the same license to this one.

Lectures Menu

Slide Decks

1. Introduction
2. **Reading files from the Web**
3. Basics of XML and HTML
4. Parsing XML / HTML content
5. Handling JSON data
6. HTTP Basics and the RCurl Package
7. Getting data via Web Forms
8. Getting data via Web APIs

Reading Files from the Web

Data Files from the Web...

← → ↻ www.gutenberg.org/ebooks/2701.txt.utf-8

The Project Gutenberg EBook of Moby Dick; or The Whale, by Herman Melville

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg license included with this eBook or online at www.gutenberg.org

Title: Moby Dick; or The Whale
 Author: Herman Melville
 Last Updated: January 3, 2009
 Posting Date: December 25, 2008 [EBook #2701]
 Release Date: June, 2001
 Language: English

*** START OF THIS PROJECT GUTENBERG EBOOK MOBY DICK; OR THE WHALE ***

Produced by Daniel Lazarus and Jonesey

← → ↻ <https://docs.google.com/spreadsheets/cc?key=0AjoVnZ9B261dHRIQVwUDRUShZQ1AA294TExtcD>

cars2004

File Edit View Insert Format Data Tools Help All changes saved in Drive

Model

	A	B	C	D	E	F	G
	Model	Cylinders	Horsepower	Speed	Weight	Width	Length
1	Citroen C2 1.1 Base	1124	61	156	932	1659	3666
2	Smart Fortwo Coupe	690	52	135	720	1515	2550
3	Mini 1.6 170	1598	170	218	1215	1690	3625
4	Nissan Micra 1.2 65	1240	65	154	950	1660	3715
5	Renault Clio 3.0 V6	2946	255	245	1400	1810	3812
6	Audi A1 1.9 TDI	1695	105	187	1295	1765	4203
7	Peugeot 307 1.4 HDi 70	1398	70	160	1179	1746	4202
8	Peugeot 407 3.0 V6 BVA	2946	211	229	1640	1811	4676
9	Mercedes Classe C 270 CDI	2955	170	230	1950	1728	4528
10	BMW 530d	2993	218	245	1995	1946	4841
11	Jaguar S-Type 2.7 V8 Bi-Turbo	2720	207	230	1722	1818	4905
12	BMW 745	4338	333	290	1870	1902	5029
13	Mercedes Classe S 400 CDI	3965	260	290	1915	2092	5038
14	Citroen C3 Pluriel 1.6i	1587	110	185	1177	1700	3934
15	BMW 24 2.5i	2494	182	235	1260	1781	4091
16	Audi TT 1.8T 180	1781	180	228	1280	1764	4041
17	Aston Martin Vantage	5925	460	305	1835	1923	4685

+ Peur!

← → ↻ <https://archive.ics.uci.edu/ml/machine-learning-databases...>

```

5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
  
```

W' World record progression

en.wikipedia.org/wiki/World_record_progression_1500_...

#	Time	Name	Nationality	Date	Meet	Location
1	22:48.4	Henry Taylor	Great Britain	Jul 25, 1908	Olympic Games	London, United Kingdom
2	22:00.0	George Hodgson	Canada	Jul 10, 1912	Olympic Games	Stockholm, Sweden
3	21:35.3	Arne Borg	Sweden	Jul 8, 1923	-	Gothenburg, Sweden
4	21:15.0	Arne Borg	Sweden	Jan 30, 1924	-	Sydney, Australia
5	21:11.4	Arne Borg	Sweden	Jul 13, 1924	-	Paris, France

Goal

From the web to R

The goal of these slides is to show you **different ways to read (data) files from the Web** into R

Synopsis

In a nutshell

We'll cover a variety of situations you most likely will find yourself dealing with:

- ▶ reading raw (plain) text
- ▶ reading tabular (spreadsheet-like) data
- ▶ reading structured data (xml, html) as text
- ▶ reading R scripts and Rdata files

Some References

- ▶ R Data Import / Export Manual
by R Core Team
- ▶ Data Manipulation with R
by Phil Spector
- ▶ R Programming for Bioinformatics
by Robert Gentleman
- ▶ The Art of R Programming
by Norm Matloff
- ▶ XML and Web Technologies for Data Sciences with R
by Deb Nolan and Duncan Temple Lang

Considerations

Keep in mind

All the material described in this presentation relies on 3 key assumptions:

- ▶ we know **where** the data is located
- ▶ we know **how** the data is stored (i.e. type of file)
- ▶ all we want is to import the data in R

Reading Files from the Web

R Data Import/Export

This is a guide to importing and exporting data to and from R.

This manual is for R, version 3.1.0 (2014-04-10).

Copyright © 2000–2013 R Core Team

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the R Core Team.

- [Acknowledgements:](#)
- [Introduction:](#)
- [Spreadsheet-like data:](#)
- [Importing from other statistical systems:](#)
- [Relational databases:](#)
- [Binary files:](#)
- [Image files:](#)
- [Connections:](#)
- [Network interfaces:](#)
- [Reading Excel spreadsheets:](#)
- [References:](#)
- [Function and variable index:](#)

Documentation

R Data Import / Export Manual

This is the authoritative source of information to read and learn *almost all* about importing —and exporting— data in R:

- ▶ html version

<http://cran.r-project.org/doc/manuals/r-release/R-data.html>

- ▶ pdf version

<http://cran.r-project.org/doc/manuals/r-release/R-data.pdf>

Good news: pretty much everything you need to know it's there

Bad news: it is not beginner friendly =(

Basics First

R is equipped with a set of handy functions that allow us to read a wide range of data files

The trick to use those functions depends on the format of the data we want to read, and the way R handles the imported values:

- ▶ what type of objects (eg `vector`, `list`, `data.frame`)
- ▶ what kind of modes (eg `character`, `numeric`, `factor`)

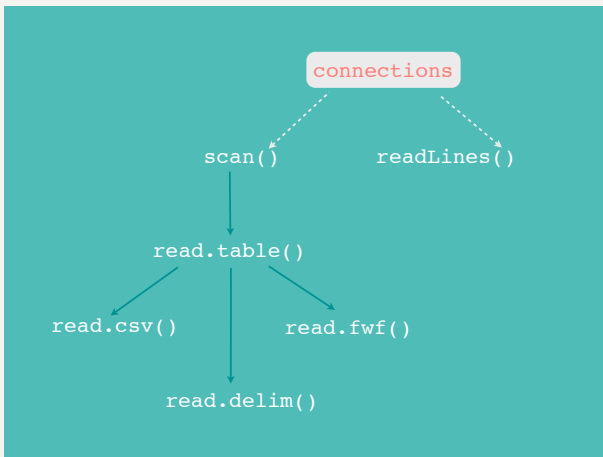
Basics First (con't)

Fundamentals

Let's start with the basic reading functions and some R technicalities

- ▶ `scan()`
- ▶ `readLines()`
- ▶ `connections`

Conceptual Diagram



Built-in reading functions

- ▶ The primary functions to read files in R are `scan()` and `readLines()`
- ▶ `readLines()` is the workhorse function to read raw text in R as character strings
- ▶ `scan()` is a low-level function for reading data values, and it is extended by `read.table()` and its related functions
- ▶ When reading files, there's the special concept under the hood called R `connections`
- ▶ Both `scan()` and `readLines()` take a `connection` as input

Connections

R connections?

Connection is the term used in R to define a mechanism for handling input (reading) and output (writing) operations.

What do they do?

A **connection** is just an object that tells R to be prepared for opening a data source (eg file in local directory, or a file url)

See the full help documentation with: `?connections`

Types of Connections

Functions to create connections

Function	Input
<code>file()</code>	path to the file to be opened or complete URL
<code>url()</code>	a complete URL
<code>gzfile()</code>	path to a file compressed by gzip
<code>bzfile()</code>	path to a file compressed by bzip2
<code>xzfile()</code>	path to a file compressed by xz
<code>unz()</code>	path to the zip file with .zip extension
<code>pipe()</code>	command line to be piped to or from
<code>fifo()</code>	path of the fifo

By default, creating a connection does not open the connection. But they may be opened with the argument `open`

Connections (con't)

Usefulness

Connections provide a **means to have more control** over the way R will “communicate” with the resources to be read (or written).

Keep in mind

Most of the times you don't need to use or worry about connections. However, you should know that they can play an important role behind the built-in functions to read data in R.

Important Connections

`file()`

The most commonly used connection is `file()`, which is used by most reading functions (to open a local file for reading or writing data).

`url()`

Because we're interested in getting data from the web, the one connection that becomes a protagonist is the `url()` connection.

Connection for the web

Using `url()`

```
url(description, open = "", blocking = TRUE,  
      encoding = getOption("encoding"))
```

The main input for `url()` is the `description` which has to be a complete URL, including scheme such as `http://`, `ftp://`, or `file://`

Example of url connection

For instance, let's create a connection to the R homepage:

```
# creating a url connection to the R homepage
r_home = url("http://www.r-project.org/")

# what's in r_home
r_home

##              description              class
## "http://www.r-project.org/"          "url"
##              mode                    text
##              "r"                     "text"
##              opened                  can read
##              "closed"                "yes"
##              can write
##              "no"

# is open?
isOpen(r_home)

## [1] FALSE
```

Note that we are just defining a connection. By default, the connection does not open anything

About Connections

Should we care?

- ▶ Again, most of the times we don't need to explicitly use `url()`.
- ▶ Connections can be used anywhere a file name could be passed to functions like `scan()` or `read.table()`.
- ▶ Usually, the reading functions —eg `readLines()`, `read.table()`, `read.csv()`— will take care of the URL connection for us.
- ▶ However, there may be occasions in which we will need to specify a `url()` connection.

Reading Text

Objective

Reading Text Files As Text

In this section we'll talk about reading text files with the purpose of importing their contents as *raw text* (ie character strings) in R.

About Text Files

“In computer literature, there is often a distinction made between text files and binary files. That distinction is somewhat misleading —every file is binary in the sense that it consists of 0s and 1s. Let’s take the term text files to mean a file that consists mainly of ASCII characters ... and that uses newline characters to give the humans the perception of lines.”

Norman Matloff (2011)

The Art of R Programming

About Text Files

Some considerations so we can all be on the same page:

- ▶ By **text files** we mean *plain text files*
- ▶ *Plain text* as an umbrella term for any file that is in a human-readable form (eg `.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

Reading Text Functions

Functions for reading text

- ▶ `readLines()` is the main function to read text files as *raw text* in R
- ▶ `scan()` is another function that can be used to read text files. It is more generic and low-level but we can specify some of its parameters to import content as text

About readLines()

Function readLines()

- ▶ `readLines()` is the workhorse function to read text files as *raw text* in R
- ▶ The main input is the file to be read, either specified with a `connection` or with the file name
- ▶ `readLines()` treats each line as a string, and it returns a character vector
- ▶ The output vector will contain as many elements as number of lines in the read file

readLines()

Using readLines()

```
readLines(con = stdin(), n = -1L, ok = TRUE,  
          warn = TRUE, encoding = "unknown")
```

- ▶ **con** a connection, which in our case will be a complete URL
- ▶ **n** the number of lines to read
- ▶ **ok** whether to reach the end of the connection
- ▶ **warn** warning if there is no End-Of-Line
- ▶ **encoding** types of encoding for input strings

Moby Dick



Example

Project Gutenberg

A great collection of texts are available from the **Project Gutenberg** which has a catalog of more than 25,000 free online books:

<http://www.gutenberg.org>

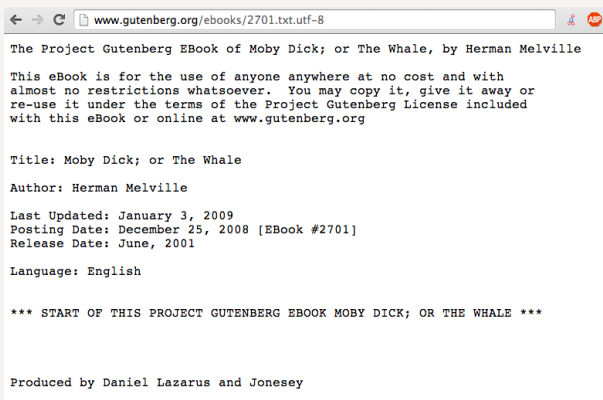
Moby Dick

Let's consider the famous novel **Moby Dick** by Herman Melville. A plain text file of Moby Dick can be found at:

<http://www.gutenberg.org/ebooks/2701.txt.utf-8>

Moby Dick text file

<http://www.gutenberg.org/ebooks/2701.txt.utf-8>



Reading Raw text

Here's how you could read the first 500 lines of content with `readLines()`

```
# url of Moby Dick (project Gutenberg)
moby_url = url("http://www.gutenberg.org/ebooks/2701.txt.utf-8")

# reading the content (first 500 lines)
moby_dick = readLines(moby_url, n = 500)
```

```
# first five lines
moby_dick[1:5]

## [1] "The Project Gutenberg EBook of Moby Dick; or The Whale, by Herman Melville"
## [2] ""
## [3] "This eBook is for the use of anyone anywhere at no cost and with"
## [4] "almost no restrictions whatsoever. You may copy it, give it away or"
## [5] "re-use it under the terms of the Project Gutenberg License included"
```

Note that each line read is stored as an element in the character vector `moby_dick`

Goot to Know

Terms of Service

Some times, reading data directly from a website may be against the **terms of use** of the site.

Web Politeness

When you're reading (and "playing" with) content from a web page, make a local copy as a courtesy to the owner of the web site so you don't overload their server by constantly rereading the page. To make a copy from inside of R, look at the **download.file()** function.

Download Moby Dick

Downloading

It is good advice to download a copy of the file to your computer, and then play with it.

Let's use `download.file()` to save a copy in our working directory. In this case we create the file `mobydick.txt`

```
# download a copy in the working directory
download.file("http://www.gutenberg.org/cache/epub/2701/pg2701.txt",
             "mobydick.txt")
```

About scan()

Function scan()

Another very useful function that we can use to read text is `scan()`. By default, `scan()` expects to read numeric values, but we can change this behavior with the argument `what`

```
scan(file = "", what = double(), nmax = -1, n = -1, sep = "",  
      quote = if(identical(sep, "\n")) "" else "'", dec = ".",  
      skip = 0, nlines = 0, na.strings = "NA",  
      flush = FALSE, fill = FALSE, strip.white = FALSE,  
      quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE,  
      comment.char = "", allowEscapes = FALSE,  
      fileEncoding = "", encoding = "unknown", text)
```

Function scan() (con't)

Some scan() arguments

- ▶ **file** the file name or a connection
- ▶ **what** type of data to be read
- ▶ **n** maximum number of data values to read
- ▶ **sep** type of separator
- ▶ **skip** number of lines to skip before reading values
- ▶ **nlines** maximum number of lines to be read

Moby Dick Chapter 1

Chapter 1 starting at line 536.

How do we get the first lines of that chapter?

```
moby_dick.txt
530     And King of the boundless sea."
531     --WHALE SONG.
532
533
534
535
536 CHAPTER 1. Loomings.
537
538
539 Call me Ishmael. Some years ago--never mind how long precisely--having
540 little or no money in my purse, and nothing particular to interest me on
541 shore, I thought I would sail about a little and see the watery part of
542 the world. It is a way I have of driving off the spleen and regulating
543 the circulation. Whenever I find myself growing grim about the mouth;
544 whenever it is a damp, drizzly November in my soul; whenever I find
545 myself involuntarily pausing before coffin warehouses, and bringing up
546 the rear of every funeral I meet; and especially whenever my hypos get
547 such an upper hand of me, that it requires a strong moral principle to
548 prevent me from deliberately stepping into the street, and methodically
549 knocking people's hats off--then, I account it high time to get to sea
550 as soon as I can. This is my substitute for pistol and ball. With a
551 philosophical flourish Cato throws himself upon his sword; I quietly
552 take to the ship. There is nothing surprising in this. If they but knew
553 it, almost all men in their degree, some time or other, cherish very
554 nearly the same feelings towards the ocean with me.
```


Reading Some Lines

Let's make it more interesting

If we want to read just a pre-specified number of lines, we have to loop over the file lines and read the content with `scan()`. For instance, let's skip the first 535 lines, and then read the following 10 lines of Chapter 1

```
# empty vector to store results
moby_dick_chap1 = rep("", 10)

# number of lines to skip until Chapter 1
skip = 535

# reading 10 lines (line-by-line using scan)
for (i in 1L:10) {
  one_line = scan(mobydick.txt, what = "", skip = skip, nlines = 1)
  # pasting the contents in one_line
  moby_dick_chap1[i] = paste(one_line, collapse = " ")
  skip = skip + 1
}
```

Note that we are using `paste()` to join (collapse) all the scanned values in `one_line`

Reading Some Lines (con't)

```
# lines 536-545
moby_dick_chap1

## [1] "CHAPTER 1. Loomings."
## [2] ""
## [3] ""
## [4] "Call me Ishmael. Some years ago--never mind how long precisely--having"
## [5] "little or no money in my purse, and nothing particular to interest me on"
## [6] "shore, I thought I would sail about a little and see the watery part of"
## [7] "the world. It is a way I have of driving off the spleen and regulating"
## [8] "the circulation. Whenever I find myself growing grim about the mouth;"
## [9] "whenever it is a damp, drizzly November in my soul; whenever I find"
## [10] "myself involuntarily pausing before coffin warehouses, and bringing up"
```

Reading an HTML file

HTML File

Our third example involves reading the contents of an html file. We're just illustrating how to import html content as raw text in R. (We are not *parsing* html; we'll see that topic in the next lecture)

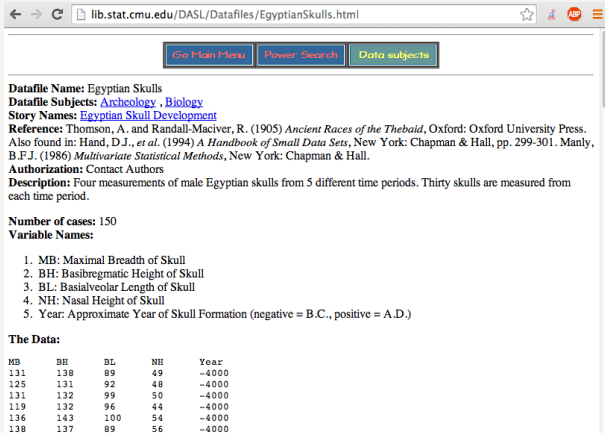
Egyptian Skulls

Let's consider the file containing information about the Egyptian Skulls data set by Thomson *et al*:

<http://lib.stat.cmu.edu/DASL/Datafiles/EgyptianSkulls.html>

HTML File

<http://lib.stat.cmu.edu/DASL/Datafiles/EgyptianSkulls.html>



← → ↻ lib.stat.cmu.edu/DASL/Datafiles/EgyptianSkulls.html ☆ 🔍 ADP ☰

[Go Main Menu](#) [Power Search](#) [Data subjects](#)

Datafile Name: Egyptian Skulls
Datafile Subjects: [Archeology](#) , [Biology](#)
Story Names: [Egyptian Skull Development](#)
Reference: Thomson, A. and Randall-Maciver, R. (1905) *Ancient Races of the Thebaid*, Oxford: Oxford University Press.
Also found in: Hand, D.J., *et al.* (1994) *A Handbook of Small Data Sets*, New York: Chapman & Hall, pp. 299-301. Manly, B.F.J. (1986) *Multivariate Statistical Methods*, New York: Chapman & Hall.
Authorization: Contact Authors
Description: Four measurements of male Egyptian skulls from 5 different time periods. Thirty skulls are measured from each time period.

Number of cases: 150
Variable Names:

1. MB: Maximal Breadth of Skull
2. BH: Basibregmatic Height of Skull
3. BL: Basialveolar Length of Skull
4. NH: Nasal Height of Skull
5. Year: Approximate Year of Skull Formation (negative = B.C., positive = A.D.)

The Data:

MB	BH	BL	NH	Year
131	138	89	49	-4000
125	131	92	48	-4000
131	132	99	50	-4000
119	132	96	44	-4000
136	143	100	54	-4000
138	137	89	56	-4000

Skulls Data

To read the html content we use `readLines()`

```
# read html file content as a string vector
skulls = readLines("http://lib.stat.cmu.edu/DASL/Datafiles/EgyptianSkulls.html")
```

```
head(skulls, n = 10)
```

```
## [1] "<TITLE>Egyptian Skulls Datafile</TITLE>"
## [2] ""
## [3] ""
## [4] "<hr size=2><center><table border=1 cellpadding=0 cellspacing=0><tr><td align=center><table border="
## [5] "<B><DT>Datafile Name:</B>    Egyptian Skulls"
## [6] "<B><DT>Datafile Subjects:</B>    <dsubjects>"
## [7] "<A HREF=\"\"/cgi-bin/dasl.cgi?query=Archeology&submit=Search&metaname=dsubjects&sort=swishrank\">Ar"
## [8] ", "
## [9] ""
## [10] "<A HREF=\"\"/cgi-bin/dasl.cgi?query=Biology&submit=Search&metaname=dsubjects&sort=swishrank\">Biolog
```

Reading Tabular Data

About Tabular Data

Tables

R is great for reading data in tabular (spreadsheet-like) format.

Tabular data, also known as rectangular data, are typically text files (ie can be read and manipulated with a text editor)

The conventional form is data values that can be seen as an array of rows and columns

Tabular Data File Formats

Two main formats

- ▶ delimited formats
- ▶ fixed-width formats

Delimited

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

Fixed-Width

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

Data Table Toy Example

Imagine we have some tabular data

Name	Gender	Homeland	Born	Jedi
Anakin	male	Tatooine	41.9BBY	yes
Amidala	female	Naboo	46BBY	no
Luke	male	Tatooine	19BBY	yes
Leia	female	Alderaan	19BBY	no
Obi-Wan	male	Stewjon	57BBY	yes
Han	male	Corellia	29BBY	no
Palpatine	male	Naboo	82BBY	no
R2-D2	unknown	Naboo	33BBY	no

Data table formats

space delimited

```
Name Gender Homeworld Born Jedi
Anakin male Tatooine 41.9BBY yes
Amidala female Naboo 46BBY no
Luke male Tatooine 19BBY yes
Leia female Alderaan 19BBY no
Obi-Wan male Stewjon 57BBY yes
Han male Corellia 29BBY no
Palpatine male Naboo 82BBY no
R2-D2 unknown Naboo 33BBY no
```

comma delimited

```
Name,Gender,Homeworld,Born,Jedi
Anakin,male,Tatooine,41.9BBY,yes
Amidala,female,Naboo,46BBY,no
Luke,male,Tatooine,19BBY,yes
Leia,female,Alderaan,19BBY,no
Obi-Wan,male,Stewjon,57BBY,yes
Han,male,Corellia,29BBY,no
Palpatine,male,Naboo,82BBY,no
R2-D2,unknown,Naboo,33BBY,no
```

tab delimited

```
Name  Gender  Homeworld  Born  Jedi
Anakin  male    Tatooine   41.9BBY  yes
Amidala female   Naboo      46BBY    no
Luke    male    Tatooine   19BBY    yes
Leia    female  Alderaan   19BBY    no
Obi-Wan male    Stewjon    57BBY    yes
Han      male    Corellia   29BBY    no
Palpatine male    Naboo      82BBY    no
R2-D2   unknown Naboo       33BBY    no
```

Fixed width

```
Name      Gender  Homeworld  Born      Jedi
Anakin     male    Tatooine   41.9BBY   yes
Amidala    female  Naboo      46BBY     no
Luke       male    Tatooine   19BBY     yes
Leia       female  Alderaan   19BBY     no
Obi-Wan    male    Stewjon    57BBY     yes
Han        male    Corellia   29BBY     no
Palpatine  male    Naboo      82BBY     no
R2-D2      unknown Naboo      33BBY     no
```

Functions

Main functions

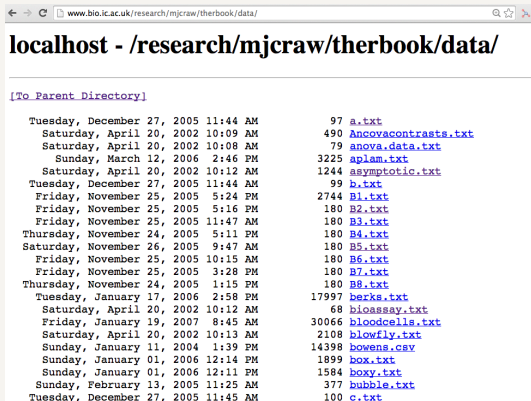
- ▶ `scan()` reads data values (one by one)
- ▶ `read.table()` main function for reading tabular data
- ▶ `read.csv()` convenient wrapper of `read.table()` designed for reading *comma separated values* (CSV) files
- ▶ `read.delim()` wrapper of `read.table()` for any delimited file format
- ▶ `read.fwf()` designed for reading files with fixed width separated values

Taxon Data

The R Book

Example from **The R Book** by Michael Crawley

<http://www.bio.ic.ac.uk/research/mjcraw/therbook/>



The screenshot shows a web browser window with the address bar displaying `www.bio.ic.ac.uk/research/mjcraw/therbook/data/`. The page title is `localhost - /research/mjcraw/therbook/data/`. Below the title, there is a link [\[To Parent Directory\]](#). The main content is a directory listing of files, each preceded by its size and a timestamp. The files are listed in two columns.

Tuesday, December 27, 2005 11:44 AM	97	a.txt
Saturday, April 20, 2002 10:09 AM	490	Ancovacontrasts.txt
Saturday, April 20, 2002 10:08 AM	79	anova.data.txt
Sunday, March 12, 2006 2:46 PM	3225	aplam.txt
Saturday, April 20, 2002 10:12 AM	1244	asymptotic.txt
Tuesday, December 27, 2005 11:44 AM	99	b.txt
Friday, November 25, 2005 5:24 PM	2744	B1.txt
Friday, November 25, 2005 5:16 PM	180	B2.txt
Friday, November 25, 2005 11:47 AM	180	B3.txt
Thursday, November 24, 2005 5:11 PM	180	B4.txt
Saturday, November 26, 2005 9:47 AM	180	B5.txt
Friday, November 25, 2005 10:15 AM	180	B6.txt
Friday, November 25, 2005 3:28 PM	180	B7.txt
Thursday, November 24, 2005 1:15 PM	180	B8.txt
Tuesday, January 17, 2006 2:58 PM	17997	berks.txt
Saturday, April 20, 2002 10:12 AM	68	bioassay.txt
Friday, January 19, 2007 8:45 AM	30066	bloodcells.txt
Saturday, April 20, 2002 10:13 AM	2108	blowfly.txt
Sunday, January 11, 2004 1:39 PM	14398	bowens.csv
Sunday, January 01, 2006 12:14 PM	1899	box.txt
Sunday, January 01, 2006 12:11 PM	1584	boxy.txt
Sunday, February 13, 2005 11:25 AM	377	bubble.txt
Tuesday, December 27, 2005 11:45 AM	100	c.txt

Taxon Data

Taxon Data (from The R Book)

<http://www.bio.ic.ac.uk/research/mjcraw/therbook/data/taxon.txt>

← → ↺ www.bio.ic.ac.uk/research/mjcraw/therbook/data/taxon.txt						
"Petals"	"Internode"	"Sepal"	"Bract"	"Petiole"	"Leaf"	"Fruit"
"1"	5.621498349	29.48059578	2.462106579	18.2034091	11.27909704	1.128032999
"2"	4.994616997	28.36024706	2.429320759	17.65204912	11.0408378	1.197616585
"3"	4.767504884	27.25431792	2.570497375	19.4083846	10.49072184	1.003808444
"4"	6.299445616	25.9242382	2.066051345	18.37915478	11.80182252	1.614051727
"5"	6.489375001	25.21130805	2.901582763	17.31304737	10.12159001	1.813333082
"6"	5.7858682	25.52433147	2.655642742	17.07215724	10.5581605	1.955524186
"7"	5.645575295	27.72735671	2.278691288	18.2902189	10.11689319	1.414356176
"8"	4.288953215	27.6381855	2.238467716	19.87376227	10.43071493	1.603331861
"9"	6.783500773	27.88777529	2.202762147	19.85326662	10.47399775	1.533533943
"10"	5.395076839	25.10904979	2.538375992	19.56545356	11.55456651	1.048182185
"11"	4.683617783	29.2459239	2.601945544	18.95335451	11.8203702	1.408570649
"12"	4.444226444	25.48653519	2.918513379	19.52866206	10.27840842	1.783670318
"13"	5.381888649	28.53008566	2.931223681	17.62409776	10.67172405	1.637521266
"14"	4.162398141	27.23922882	2.759099725	17.89805071	11.42458281	1.307677167
"15"	6.546125565	27.73424951	2.329725784	17.38995746	11.25045634	1.908787547
"16"	5.133151769	29.33237736	2.448247009	19.40622249	11.15433346	1.413823925
"17"	6.31750724	28.54598267	2.445203138	18.14507123	11.13155215	1.376095509
"18"	6.872582397	27.36377647	2.551340885	19.37856593	11.39634002	1.845322428
"19"	4.265699548	26.9166827	2.930159411	19.52993241	11.87800257	1.122888175
"20"	4.069245625	28.00872767	2.306900441	18.10018898	10.28587666	1.096098375
"21"	6.695262246	30.98241938	2.280554955	19.58999705	10.01733931	1.993469856
"22"	5.804502115	27.60866985	2.541535351	19.99502062	10.71206051	1.078787122
"23"	5.649255792	26.0145167	2.630009346	19.73151656	11.56895111	1.97965464
"24"	4.975115406	29.83647166	2.533115646	17.85816577	11.09149866	1.399543824
"25"	5.242381538	29.25618227	2.940088125	18.68002039	10.03097795	1.131223326
"26"	5.063066089	30.23405514	2.816805174	17.11427146	10.65715001	1.955607401
"27"	5.006042848	27.92346287	2.054945288	17.89395755	10.29285526	1.231612531
"28"	5.980790831	27.59682027	2.829002908	19.19108937	10.55412055	1.877992273

Taxon Data

Let's read the data "taxon"

```
# url of taxon data
taxon_url = "http://www.bio.ic.ac.uk/research/mjcraw/therbook/data/taxon.txt"

# import data in R
taxon = read.table(taxon_url, header = TRUE, row.names = 1)
```

```
head(taxon)
```

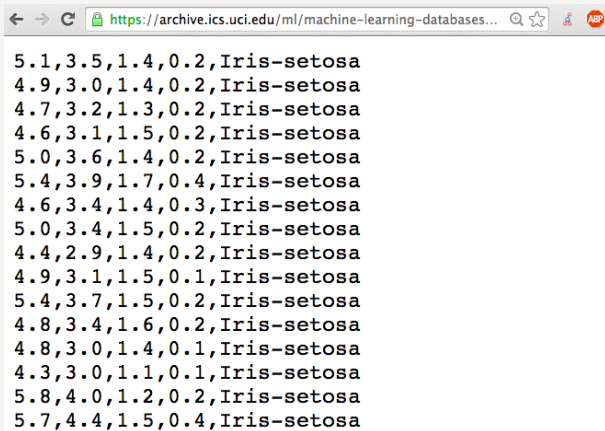
```
##   Petals Internode Sepal Bract Petiole  Leaf Fruit
## 1  5.621    29.48  2.462 18.20   11.28  1.128  7.876
## 2  4.995    28.36  2.429 17.65   11.04  1.198  7.025
## 3  4.768    27.25  2.570 19.41   10.49  1.004  7.817
## 4  6.299    25.92  2.066 18.38   11.80  1.614  7.672
## 5  6.489    25.21  2.902 17.31   10.12  1.813  7.758
## 6  5.786    25.52  2.656 17.07   10.56  1.956  7.881
```

Iris Example

Iris Data

Data set "iris" from UCI Machine Learning Repo

<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>



The screenshot shows a web browser window with the address bar containing the URL `https://archive.ics.uci.edu/ml/machine-learning-databases...`. The main content area displays 15 rows of the Iris dataset, each consisting of four numerical values followed by the species name 'Iris-setosa'.

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
```

Iris Data

How do we read the data?

If you try to simply use `read.csv()`, you'll be disappointed:

```
# URL of data file
iris_file = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# this won't work
iris_data = read.csv(iris_file, header = FALSE)
```

Note that the URL starts with `https://`, that means a secured connection. The solution requires some special functions:

- ▶ We need to use the R package "RCurl" to make an HTTPS request with `getURL()`
- ▶ We also need to use `textConnection()` inside `read.csv()`

Reading Iris Data

This is how to successfully read the iris data set in R:

```
# URL of data file
iris_file = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

library(RCurl)
iris_url = getURL(iris_file)
iris_data = read.csv(textConnection(iris_url), header = FALSE)
```

```
head(iris_data)

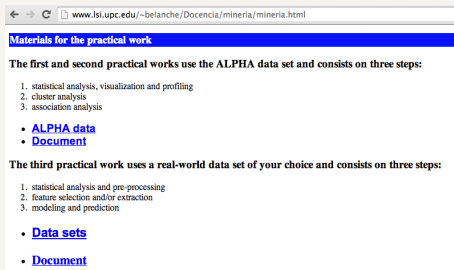
##      V1 V2 V3 V4      V5
## 1 5.1 3.5 1.4 0.2 Iris-setosa
## 2 4.9 3.0 1.4 0.2 Iris-setosa
## 3 4.7 3.2 1.3 0.2 Iris-setosa
## 4 4.6 3.1 1.5 0.2 Iris-setosa
## 5 5.0 3.6 1.4 0.2 Iris-setosa
## 6 5.4 3.9 1.7 0.4 Iris-setosa
```

Excel File Example

Excel file

Example from *Data Mining Course* by Lluís Belanche

<http://www.lsi.upc.edu/~belanche/Docencia/mineria/mineria.html>



We'll read the excel file named `alpha.xls` available at:

`alpha_xls="http://www.lsi.upc.edu/~belanche/Docencia/mineria/Practiques/alpha.xls"`

Excel alpha data

Alpha Data (from Data Mining course)

	A	B	C	D	E	F	G	H	I	J	K	L	M
	iden	beta1	gamma1	delta1	alfa1	alfa2	estr1	beta2	gamma2	delta2	alfa2	edat	memb
1	1	0	0	0	0	0	1	0	0	0	0	40	4
2	2	0	0	0	0	0	1	0	0	0	1	57	2
3	3	0	0	0	0	0	1	0	0	0	1	35	1
4	4	0	0	0	0	0	1	0	0	1	0	54	3
5	5	0	0	0	1	0	0	0	0	1	0	43	5
6	6	0	0	1	0	0	0	0	1	0	0	19	6
7	7	0	0	0	0	0	1	0	0	0	0	38	5
8	8	0	0	0	0	1	0	0	1	0	0	42	3
9	9	0	0	1	0	0	0	0	0	1	0	21	2
10	10	0	0	0	1	0	0	0	0	0	0	19	5
11	11	0	0	0	1	0	0	0	0	1	0	18	4
12	12	0	0	0	0	0	1	0	0	0	1	48	3
13	13	0	0	1	0	0	0	0	0	1	0	30	1
14	14	0	0	0	0	0	0	0	0	0	0	28	2
15	15	1	0	0	0	0	0	0	0	1	0	20	2
16	16	0	0	0	0	0	1	0	0	0	1	18	5
17	17	0	0	0	1	0	0	0	0	0	1	21	7
18	18	0	0	0	0	1	0	0	1	0	0	23	2
19	19	0	0	0	1	0	0	0	0	0	1	28	1
20	20	0	0	1	0	0	0	0	0	1	0	45	3
21	21	0	0	1	0	0	0	0	0	1	0	48	1
22	22	0	0	0	1	0	0	0	0	0	1	24	2
23	23	0	0	0	0	1	0	0	0	0	1	33	3
24	24	0	0	0	0	0	0	0	0	0	0	47	5

Reading alpha Data

We need to use the function `read.xls()` from the package `"gdata"` (you need to have Perl installed in your machine)

```
# load package 'gdata'
library(gdata)

# excel file (1st worksheet named "dades")
alpha_xls = "http://www.lsi.upc.edu/~belanche/Docencia/mineria/Practiques/alpha.xls"
```

Count the number of sheets in excel file, and list sheet names:

```
# how many sheets
sheetCount(alpha_xls)

## [1] 2

# names of sheets
sheetNames(alpha_xls)

## [1] "dades"      "diccionari"
```

Reading alpha Data

Since the data set is in the first worksheet we use the argument `sheet = 1`:

```
# import sheet 1 (dades) in R
alpha_data = read.xls(alpha_xls, sheet = 1)
```

```
head(alpha_data)
```

```
##   iden beta1 gamma1 delta1 alfa1 altra1 estr1 beta2 gamma2 delta2 alfa2 edat
## 1    1     0     0     0     0     1     0     1     0     0     0    40
## 2    2     0     0     0     0     1     0     0     0     0     1    57
## 3    3     0     0     0     0     1     0     0     0     0     1    35
## 4    4     0     0     0     0     1     0     0     1     0     0    54
## 5    5     0     0     1     0     0     0     0     0     1     0    43
## 6    6     0     0     1     0     0     0     0     1     0     0    19
##  memb estd eciv prof1 prof2 ingr
## 1     4   12    1     1     0  190
## 2     2   12    1     1     0  220
## 3     1   16    0     1     0  220
## 4     3   14    1     1     0  220
## 5     5   14    1     0     1  110
## 6     6   14    0     0     0  220
```

Google Spreadsheet

Cars2004 Data

Example with data in Google Docs Spreadsheet

cars2004

File Edit View Insert Format Data Tools Help All changes saved in Drive

Model

	A	B	C	D	E	F	G
1	Model	Cylinders	Horsepower	Speed	Weight	Width	Length
2	Citroen C2 1.1 Base	1124	61	158	932	1659	3666
3	Smart Fortwo Coupe	698	52	135	730	1515	2500
4	Mini 1.6 170	1598	170	218	1215	1690	3625
5	Nissan Micra 1.2 65	1240	65	154	965	1660	3715
6	Renault Clio 3.0 V6	2946	255	245	1400	1810	3812
7	Audi A3 1.9 TDI	1896	105	187	1295	1765	4203
8	Peugeot 307 1.4 HDI 70	1398	70	160	1179	1746	4202
9	Peugeot 407 3.0 V6 BVA	2946	211	229	1640	1811	4676
10	Mercedes Classe C 270 CDI	2685	170	230	1600	1728	4528
11	BMW 530d	2993	218	245	1595	1846	4841
12	Jaguar S-Type 2.7 V6 Bi-Turbo	2720	207	230	1722	1818	4905
13	BMW 745i	4398	333	250	1870	1902	5029
14	Mercedes Classe S 400 CDI	3966	260	250	1915	2092	5038
15	Citroen C3 Pluriel 1.6i	1587	110	185	1177	1700	3934
16	BMW Z4 2.5i	2494	192	235	1260	1781	4091
17	Audi TT 1.8T 180	1781	180	228	1280	1764	4041
18	Aston Martin Vanquish	5935	460	306	1835	1923	4665

Feuil1

<https://docs.google.com/spreadsheet/cc?key=0AjoVnZ9iB261dHRfQlVuWDRUSHdZQ1A4N294TEstc0E&usp=sharing>

Reading Cars2004 Google Doc

To read data from a Google Doc Spreadsheet we need to use the R package "RCurl" (to connect via a secured HTTP). In addition we need to know the **public key** of the document. Here's how to read the Cars2004 google doc:

```
# load package RCurl
library(RCurl)

# google docs spreadsheets url
google_docs = "https://docs.google.com/spreadsheet/"

# public key of data 'cars'
cars_key = "pub?key=0AjoVnZ9iB261dHRfQ1VuWDRUSHdZQ1A4N294TEstcOE&output=csv"

# download URL of data file
cars_csv = getURL(paste(google_docs, cars_key, sep = ""))

# import data in R (through a text connection)
cars2004 = read.csv(textConnection(cars_csv), row.names = 1, header = TRUE)
```

Reading Cars2004 Google Doc

```
cars2004 = read.csv(mycars, row.names = 1, header = TRUE)
```

	Cylinders	Horsepower	Speed	Weight	Width	Length
## Citroen C2 1.1 Base	1124	61	158	932	1659	3666
## Smart Fortwo Coupe	698	52	135	730	1515	2500
## Mini 1.6 170	1598	170	218	1215	1690	3625
## Nissan Micra 1.2 65	1240	65	154	965	1660	3715
## Renault Clio 3.0 V6	2946	255	245	1400	1810	3812
## Audi A3 1.9 TDI	1896	105	187	1295	1765	4203
## Peugeot 307 1.4 HDI 70	1398	70	160	1179	1746	4202
## Peugeot 407 3.0 V6 BVA	2946	211	229	1640	1811	4676
## Mercedes Classe C 270 CDI	2685	170	230	1600	1728	4528
## BMW 530d	2993	218	245	1595	1846	4841
## Jaguar S-Type 2.7 V6 Bi-Turbo	2720	207	230	1722	1818	4905
## BMW 745i	4398	333	250	1870	1902	5029
## Mercedes Classe S 400 CDI	3966	260	250	1915	2092	5038
## Citroen C3 Pluriel 1.6i	1587	110	185	1177	1700	3934
## BMW Z4 2.5i	2494	192	235	1260	1781	4091
## Audi TT 1.8T 180	1781	180	228	1280	1764	4041
## Aston Martin Vanquish	5935	460	306	1835	1923	4665
## Bentley Continental GT	5998	560	318	2385	1918	4804
## Ferrari Enzo	5998	660	350	1365	2650	4700
## Renault Scenic 1.9 dCi 120	1870	120	188	1430	1805	4259
## Volkswagen Touran 1.9 TDI 105	1896	105	180	1498	1794	4391
## Land Rover Defender Td5	2495	122	135	1695	1790	3883
## Land Rover Discovery Td5	2495	138	157	2175	2190	4705
## Nissan X-Trail 2.2 dCi	2184	136	180	1520	1765	4455

Wikipedia Table

Wikipedia Table

Let's read an HTML table from Wikipedia. This is not technically a file, but a piece of content from an html document



The screenshot shows a web browser window with the address bar displaying `en.wikipedia.org/wiki/World_record_progression_1500_...`. The browser tab is titled "W World record progression". The main content area displays a table with 7 columns: #, Time, Name, Nationality, Date, Meet, and Location. The table contains 5 rows of data, each representing a world record progression. The first row shows Henry Taylor from Great Britain with a time of 22:48.4, set on July 25, 1908, at the Olympic Games in London, United Kingdom. The second row shows George Hodgson from Canada with a time of 22:00.0, set on July 10, 1912, at the Olympic Games in Stockholm, Sweden. The third row shows Arne Borg from Sweden with a time of 21:35.3, set on July 8, 1923, at Gothenburg, Sweden. The fourth row shows Arne Borg from Sweden with a time of 21:15.0, set on Jan 30, 1924, at Sydney, Australia. The fifth row shows Arne Borg from Sweden with a time of 21:11.4, set on Jul 13, 1924, at Paris, France.

#	Time	Name	Nationality	Date	Meet	Location
1	22:48.4	Henry Taylor	 Great Britain	Jul 25, 1908	Olympic Games	 London, United Kingdom
2	22:00.0	George Hodgson	 Canada	Jul 10, 1912	Olympic Games	 Stockholm, Sweden
3	21:35.3	Arne Borg	 Sweden	Jul 8, 1923	-	 Gothenburg, Sweden
4	21:15.0	Arne Borg	 Sweden	Jan 30, 1924	-	 Sydney, Australia
5	21:11.4	Arne Borg	 Sweden	Jul 13, 1924	-	 Paris, France

http://en.wikipedia.org/wiki/World_record_progression_1500_metres_freestyle

Reading data in an HTML Table

To read an HTML table we need to use the function `readHTMLTable` from the R package "XML"

```
# load XML
library(XML)

# url
swim_wiki = "http://en.wikipedia.org/wiki/World_record_progression_1500_metres_freestyle"
```

Since we want the first table, we specify the parameter `which = 1`

```
# reading HTML table
swim1500 = readHTMLTable(swim_wiki, which = 1, stringsAsFactors = FALSE)
```

Note that we can pass parameters to `data.frame()`, in this case `stringsAsFactors = FALSE`

Reading data in an HTML Table

```
head(swim1500)
```

```
##   #      Time                Name      Nationality
## 1 1 22:48.4      Taylor , Henry Henry Taylor Great Britain
## 2 2 22:00.0  Hodgson , George George Hodgson      Canada
## 3 3 21:35.3      Borg , Arne Arne Borg            Sweden
## 4 4 21:15.0      Borg , Arne Arne Borg            Sweden
## 5 5 21:11.4      Borg , Arne Arne Borg            Sweden
## 6 6 20:06.6      Charlton , Boy Boy Charlton      Australia
##                                     Date      Meet
## 1 01908-07-25-0000Jul 25, 1908 Olympic Games
## 2 01912-07-10-0000Jul 10, 1912 Olympic Games
## 3 01923-07-08-0000Jul 8, 1923      -
## 4 01924-01-30-0000Jan 30, 1924      -
## 5 01924-07-13-0000Jul 13, 1924      -
## 6 01924-07-15-0000Jul 15, 1924 Olympic Games
##                                     Location Ref
## 1 United Kingdom, London ! London, United Kingdom
## 2      Sweden, Stockholm ! Stockholm, Sweden
## 3      Sweden, Gothenburg ! Gothenburg, Sweden
## 4      Australia, Sydney ! Sydney, Australia
## 5      France, Paris ! Paris, France
## 6      France, Paris ! Paris, France
```

R script and RData

R script and RData

Last but not least, we can also import data inside an R script and in and `.RData` file. In this case the data files come from John Maindonald's website

- ▶ The table is in the form of an R script

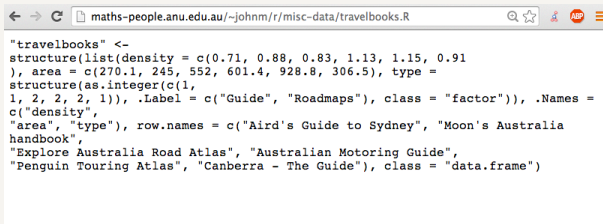
<http://maths-people.anu.edu.au/~johnm/r/misc-data/travelbooks.R>

- ▶ The other type of data is in reality a bunch of data sets in the form of an `.RData` file

<http://maths-people.anu.edu.au/~johnm/r/dsets/usingR.RData>

travelbooks Data

Travelbooks Data (by John Maindonald)



A screenshot of a web browser window. The address bar shows the URL `maths-people.anu.edu.au/~johnm/r/misc-data/travelbooks.R`. The browser's toolbar includes back, forward, and refresh buttons, along with search, star, and extension icons. The main content area displays an R script that defines a data frame named `travelbooks`. The script uses `structure()` to create a list of density values and an area vector, then uses `as.integer()` to create a factor for the type of travel guide. The row names are specified as `"Aird's Guide to Sydney", "Moon's Australia handbook", "Explore Australia Road Atlas", "Australian Motoring Guide", "Penguin Touring Atlas", "Canberra - The Guide"`.

```
"travelbooks" <-  
structure(list(density = c(0.71, 0.88, 0.83, 1.13, 1.15, 0.91  
, area = c(270.1, 245, 552, 601.4, 928.8, 306.5), type =  
structure(as.integer(c(1,  
1, 2, 2, 2, 1))), .Label = c("Guide", "Roadmaps"), class = "factor")), .Names =  
c("density",  
"area", "type"), row.names = c("Aird's Guide to Sydney", "Moon's Australia  
handbook",  
"Explore Australia Road Atlas", "Australian Motoring Guide",  
"Penguin Touring Atlas", "Canberra - The Guide"), class = "data.frame")
```

Reading R script with data.frame

To read the script we simply need to use the function `source()`

```
# url
travelbooks = "http://maths-people.anu.edu.au/~johnm/r/misc-data/travelbooks.R"

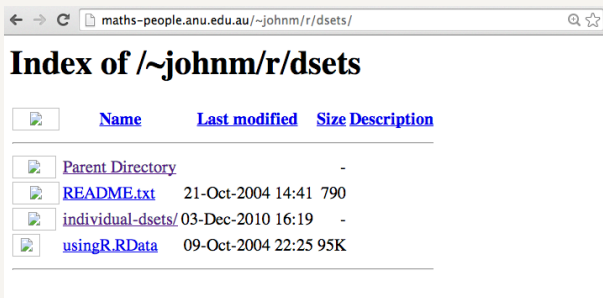
# sourcing file
source(travelbooks)
```

travelbooks






##	density	area	type
## Aird's Guide to Sydney	0.71	270.1	Guide
## Moon's Australia handbook	0.88	245.0	Guide
## Explore Australia Road Atlas	0.83	552.0	Roadmaps
## Australian Motoring Guide	1.13	601.4	Roadmaps
## Penguin Touring Atlas	1.15	928.8	Roadmaps
## Canberra - The Guide	0.91	306.5	Guide

RData

.RData file using R. RData containing several data frames



The screenshot shows a web browser window with the address bar containing the URL `maths-people.anu.edu.au/~johnm/r/dsets/`. The page title is "Index of /~johnm/r/dsets". Below the title is a table listing the contents of the directory. The table has four columns: "Name", "Last modified", "Size", and "Description". The entries are: "Parent Directory" (a link with a folder icon), "README.txt" (a file icon), "individual-dsets/" (a link with a folder icon), and "usingR.RData" (a link with a file icon).

	Name	Last modified	Size	Description
	Parent Directory		-	
	README.txt	21-Oct-2004 14:41	790	
	individual-dsets/	03-Dec-2010 16:19	-	
	usingR.RData	09-Oct-2004 22:25	95K	

Reading RData data sets

For those data sets that are inside an .RData file, we need to use the function `load()` and pass the file with `url()`

```
# let's remove all objects in session
rm(list = ls())

# url with .RData
load(url("http://maths-people.anu.edu.au/~johnm/r/dsets/usingR.RData"))
```

```
# list of read data sets
ls()
```

## [1]	"ais"	"alpha_csv"	"alpha_data"	"alpha_xls"
## [5]	"anesthetic"	"austpop"	"cars2004"	"Cars93.summary"
## [9]	"dewpoint"	"dolphins"	"elasticband"	"florida"
## [13]	"hills"	"huron"	"i"	"iris_data"
## [17]	"islandcities"	"kiwishade"	"leafshape"	"milk"
## [21]	"moby_dick"	"moby_dick_chap1"	"moby_text"	"moths"
## [25]	"mycars"	"myiris"	"myRData"	"myswim"
## [29]	"mytaxon"	"oddbooks"	"one_line"	"orings"
## [33]	"possum"	"primates"	"r_home"	"rainforest"
## [37]	"seedrates"	"skip"	"skulls"	"sw"
## [41]	"swim1500"	"taxon"	"thm"	"tinting"
## [45]	"travelbooks"			