

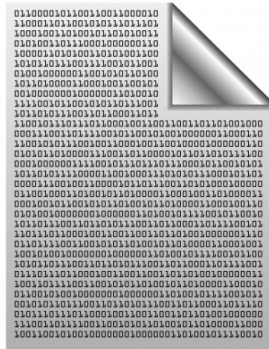
# Reading Data Tables

STAT 133

Gaston Sanchez

`github.com/ucb-stat133/stat133-fall-2016`

# Files and Formats



Every file is binary in the sense that it consists of 0s and 1s

# 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

# Mandatory Reading

## **Introduction to Data Technologies (ItDT)**

by Paul Murrell

- ▶ Preface
- ▶ Chap 1: Introduction
- ▶ Chap 5: Data Storage

# Tabular Datasets



# Data Tables

How to store tables in a file?

name	gender	height
Leia Skywalker	female	1.50
Luke Skywalker	male	1.72
Han Solo	male	1.80

# Dataset “starwarstoy”

name	gender	height	weight	jedi	species	weapon
Luke Skywalker	male	1.72	77	jedi	human	lightsaber
Leia Skywalker	female	1.50	49	no_jedi	human	blaster
Obi-Wan Kenobi	male	1.82	77	jedi	human	lightsaber
Han Solo	male	1.80	80	no_jedi	human	blaster
R2-D2	male	0.96	32	no_jedi	droid	unarmed
C-3PO	male	1.67	75	no_jedi	droid	unarmed
Yoda	male	0.66	17	jedi	yoda	lightsaber
Chewbacca	male	2.28	112	no_jedi	wookiee	bowcaster

Source: Wookieepedia <http://starwars.wikia.com/wiki>

# Data Table (computationally)

How to store data cells?

What type of format?

# 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 (txt)

```
name gender height weight jedi species weapon
"Luke Skywalker" male 1.72 77 jedi human lightsaber
"Leia Skywalker" female 1.50 49 no_jedi human blaster
"Obi-Wan Kenobi" male 1.82 77 jedi human lightsaber
"Han Solo" male 1.80 80 no_jedi human blaster
"R2-D2" male 0.96 32 no_jedi droid unarmed
"C-3P0" male 1.67 75 no_jedi droid unarmed
"Yoda" male 0.66 17 jedi yoda lightsaber
"Chewbacca" male 2.28 112 no_jedi wookiee bowcaster
```

# Comma Delimited (csv)

```
name,gender,height,weight,jedi,species,weapon  
Luke Skywalker,male,1.72,77,jedi,human,lightsaber  
Leia Skywalker,female,1.50,49,no_jedi,human,blaster  
Obi-Wan Kenobi,male,1.82,77,jedi,human,lightsaber  
Han Solo,male,1.80,80,no_jedi,human,blaster  
R2-D2,male,0.96,32,no_jedi,droid,unarmed  
C-3P0,male,1.67,75,no_jedi,droid,unarmed  
Yoda,male,0.66,17,jedi,yoda,lightsaber  
Chewbacca,male,2.28,112,no_jedi,wookiee,bowcaster
```



## Tab Delimited (txt, tsv)

name	gender	height	weight	jedi	species	weapon
"Luke Skywalker"	male	1.72	77	jedi	human	lightsaber
"Leia Skywalker"	female	1.50	49	no_jedi	human	blaster
"Obi-Wan Kenobi"	male	1.82	77	jedi	human	lightsaber
"Han Solo"	male	1.80	80	no_jedi	human	blaster
"R2-D2"	male	0.96	32	no_jedi	droid	unarmed
"C-3P0"	male	1.67	75	no_jedi	droid	unarmed
"Yoda"	male	0.66	17	jedi	yoda	lightsaber
"Chewbacca"	male	2.28	112	no_jedi	wookiee	bowcaster

# Fixed-width Formats

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

## Fixed-Width (txt)

name	gender	height	weight	jedi
"Luke Skywalker"	male	1.72	77	jedi
"Leia Skywalker"	female	1.50	49	no_jedi
"Obi-Wan Kenobi"	male	1.82	77	jedi
"Han Solo"	male	1.80	80	no_jedi
"R2-D2"	male	0.96	32	no_jedi
"C-3PO"	male	1.67	75	no_jedi
"Yoda"	male	0.66	17	jedi
"Chewbacca"	male	2.28	112	no_jedi

# In Summary

## Plain Text Formats

- ▶ The simplest way to store information in computer memory is a file with a **plain text format**
- ▶ The basic conceptual structure of a plain text format is that the **data are arranged in rows**, with several values stored on each row
- ▶ The main characteristic of a plain text format is that all of the information in a file, even numeric information, is stored as text

# Importing Data Tables in R

# 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**

<http://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

The read data output is always a **data.frame**

# 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
<code>file</code>	name of file
<code>header</code>	whether column names are in 1st line
<code>sep</code>	field separator
<code>quote</code>	quoting characters
<code>dec</code>	character for decimal point
<code>row.names</code>	optional vector of row names
<code>col.names</code>	optional vector of column names
<code>na.strings</code>	character treated as missing values
<code>colClasses</code>	optional vector of classes for columns
<code>nrows</code>	maximum number of rows to read in
<code>skip</code>	number of lines to skip before reading data
<code>check.names</code>	check valid column names
<code>stringsAsFactors</code>	should characters be converted to factors

## Consider some dataset

Num	Name	Full	Gender	Height	Weight
1	Anakin	"Anakin Skywalker"	male	1.88	84
2	Padme	"Padme Amidala"	female	1.65	45
3	Luke	"Luke Skywalker"	male	1.72	77
4	Leia	"Leia Skywalker"	female	1.50	NA

# Arguments for read.table()

`row.names = 1`

`header = TRUE`

Num	Name	Full	Gender	Height	Weight
1	Anakin	"Anakin Skywalker"	male	1.88	84
2	Padme	"Padme Amidala"	female	1.65	45
3	Luke	"Luke Skywalker"	male	1.72	77
4	Leia	"Leia Skywalker"	female	1.50	NA

`quote = "\"\""`

`dec = "."`

`na.strings = "NA"`

# Assumption

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

e.g. `"/Users/Gaston/Documents"`

# starwarstoy.txt

```
name gender height weight jedi species weapon
"Luke Skywalker" male 1.72 77 jedi human lightsaber
"Leia Skywalker" female 1.5 49 no_jedi human blaster
"Obi-Wan Kenobi" male 1.82 77 jedi human lightsaber
"Han Solo" male 1.8 80 no_jedi human blaster
"R2-D2" male 0.96 32 no_jedi droid unarmed
"C-3PO" male 1.67 75 no_jedi droid unarmed
"Yoda" male 0.66 17 jedi yoda lightsaber
"Chewbacca" male 2.28 112 no_jedi wookiee bowcaster
```

Lecture data files at:

<https://github.com/gastonstat/stat133/tree/master/datasets>

# Reading starwarstoy.txt

Blank space delimiter " "

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

Note: by default `read.table()` (and friends) convert character strings into factors

# Reading starwarstoy.txt

Compare to this other option:

```
# first column as row names  
sw_txt1 <- read.table(  
  file = "starwarstoy.txt",  
  header = TRUE,  
  row.names = 1)
```

# Reading starwarstoy.txt

Limit the number of rows to read in (first 4 individuals):

```
# first column as row names  
sw_txt2 <- read.table(  
  file = "starwarstoy.txt",  
  header = TRUE,  
  row.names = 1,  
  nrows = 4)
```



# Reading starwarstoy.txt

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

```
# first column as row names
sw_txt3 <- read.table(
  file = "starwarstoy.txt",
  header = FALSE,
  skip = 1,
  row.names = 1,
  nrows = 4)
```

# starwarstoy.csv

```
name,gender,height,weight,jedi,species,weapon
Luke Skywalker,male,1.72,77,jedi,human,lightsaber
Leia Skywalker,female,1.5,49,no_jedi,human,blaster
Obi-Wan Kenobi,male,1.82,77,jedi,human,lightsaber
Han Solo,male,1.8,80,no_jedi,human,blaster
R2-D2,male,0.96,32,no_jedi,droid,unarmed
C-3P0,male,1.67,75,no_jedi,droid,unarmed
Yoda,male,0.66,17,jedi,yoda,lightsaber
Chewbacca,male,2.28,112,no_jedi,wookiee,bowcaster
```

# Reading starwarstoy.csv

Comma delimiter ",", "

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

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

## starwarstoy.csv2

```
name;gender;height;weight;jedi;species;weapon
Luke Skywalker;male;1,72;77;jedi;human;lightsaber
Leia Skywalker;female;1,5;49;no_jedi;human;blaster
Obi-Wan Kenobi;male;1,82;77;jedi;human;lightsaber
Han Solo;male;1,8;80;no_jedi;human;blaster
R2-D2;male;0,96;32;no_jedi;droid;unarmed
C-3P0;male;1,67;75;no_jedi;droid;unarmed
Yoda;male;0,66;17;jedi;yoda;lightsaber
Chewbacca;male;2,28;112;no_jedi;wookiee;bowcaster
```

# Reading starwarstoy.csv2

Semicolon delimiter ";", " and decimal symbol ",", "

```
# using read.table()
sw_csv2 <- read.table(file = "starwarstoy.csv",
                      header = TRUE,
                      sep = ";", dec = ",")

# using read.csv2()
sw_csv2 <- read.csv2(file = "starwarstoy.csv2")
```

# starwarstoy.tsv

name	gender	height	weight	jedi	species	weapon
Luke Skywalker	male	1.72	77	jedi	human	lightsaber
Leia Skywalker	female	1.5	49	no_jedi	human	blaster
Obi-Wan Kenobi	male	1.82	77	jedi	human	lightsaber
Han Solo	male	1.8	80	no_jedi	human	blaster
R2-D2	male	0.96	32	no_jedi	droid	unarmed
C-3P0	male	1.67	75	no_jedi	droid	unarmed
Yoda	male	0.66	17	jedi	yoda	lightsaber
Chewbacca	male	2.28	112	no_jedi	wookiee	bowcaster

# Reading starwarstoy.tsv

Tab delimiter "\t"

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

# starwarstoy.dat

```
name%gender%height%weight%jedi%species%weapon
Luke Skywalker%male%1.72%77%jedi%human%lightsaber
Leia Skywalker%female%1.5%49%no_jedi%human%blaster
Obi-Wan Kenobi%male%1.82%77%jedi%human%lightsaber
Han Solo%male%1.8%80%no_jedi%human%blaster
R2-D2%male%0.96%32%no_jedi%droid%unarmed
C-3P0%male%1.67%75%no_jedi%droid%unarmed
Yoda%male%0.66%17%jedi%yoda%lightsaber
Chewbacca%male%2.28%112%no_jedi%wookiee%bowcaster
```



# Reading starwarstoy.dat

Note that this file has "%" as delimiter

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

## read.table() and friends

Function	Description
read.csv()	comma separated values
read.csv2()	semicolon separated values (Europe)
read.delim()	tab separated values
read.delim2()	tab separated values (Europe)

There is also the `read.fwf()` function for reading a table of **fixed width format**

# 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?

# Summary

## 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

# Location of data file

Sometimes the issue is not the type of file but its location

- ▶ zip file
- ▶ url (http standard)
- ▶ url (https HTTP secure)

# Reading compressed files

R provides various `connections` functions for opening and reading compressed files:

- ▶ `unz()` reads only a single zip file
- ▶ `gzfile()` for gzip, bzip2, xz, lzma
- ▶ `bzfile()` for bzip2
- ▶ `xzfile()` for xz

You pass a connection to the argument `file` in any of the reading files functions.

# Reading zip files

`unz(description, filename)`

- ▶ `description` is the full path to the zip file with `.zip` extension if required
- ▶ `filename` is the name of the file



# Reading a single zip file

starwarstoy.zip contains a copy of the file  
starwarstoy.txt; to import it in R type:

```
sw_zip <- read.table(  
  file = unz(description = "starwarstoy.zip",  
              "starwarstoy.txt")  
)
```

# 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 an url connection to

```
# creating a url connection to some file
edu <- url("http://gastonsanchez.com/education.csv")

# what's in 'edu'
edu

##              description
## "http://gastonsanchez.com/education.csv"
##              class
##              "url"
##              mode
##              "r"
##              text
##              "text"
##              opened
##              "closed"
##              can read
##              "yes"
##              can write
##              "no"

# is open?
isOpen(edu)

## [1] FALSE
```

# About Connections

## Should we care?

- ▶ 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 `read.table()`
- ▶ Usually, the reading functions —eg `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.

# Good 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.

# Downloading Files

## Downloading files from the web

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 `education.csv`

```
# download a copy in your working directory  
download.file("http://gastonsanchez.com/education.csv",  
             "education.csv")
```

# Reading files via https

To read data tables via https (to connect via a secured HTTP) we need to use the R package "RCurl"

```
# load package RCurl  
library(RCurl)  
  
# URL of data file  
url <- getURL("https://???)  
  
# import data in R (through a text connection)  
df <- read.csv(textConnection(url),  
               row.names = 1, header = TRUE)
```

# Clicker poll

Which of the following sentences is TRUE

- A) spreadsheet formats have no limits on the numbers of columns and rows
- B) spreadsheet format is always better than a plain text or binary data format
- C) a lot of unnecessary additional information is stored in a spreadsheet file
- D) All of the above



R package "readr"

# Package "readr"

The package "readr" (by Wickham *et al*) is a new package that makes it easy to read many types of tabular data

<http://blog.rstudio.org/2015/04/09/readr-0-1-0/>

<http://cran.r-project.org/web/packages/readr/vignettes/design.html>

# Package "readr"

```
# remember to install 'readr'  
install.packages("readr")  
  
# load it  
library(readr)
```

# "readr" Functions

- ▶ Fixed width files with `read_table()` and `read_fwf()`
- ▶ Delimited files with `read_delim()`, `read_csv()`, `read_tsv()`, and `read_csv2()`

# About "readr"

## "readr" functions ...

- ▶ are around 10x faster than base functions
- ▶ are more consistent (better designed)
- ▶ produce data frames that are easier to use
- ▶ they have more flexible column specification

# Input Arguments

- ▶ `file`
- ▶ `col_names`
- ▶ `col_types`
- ▶ `progress`

# Input Arguments

`file` gives the file to read; a url or local path. A local path can point to a a zipped, bziped, xzipped, or gzipped file it'll be automatically uncompressed in memory before reading.

# Input Arguments

`col_names`: describes the column names (equivalent to `header` in base R). It has three possible values:

- ▶ `TRUE` will use the the first row of data as column names.
- ▶ `FALSE` will number the columns sequentially.
- ▶ A character vector to use as column names.



# Input Arguments

`col_types` (equivalent to `colClasses` automatically detects column types:

- ▶ `col_logical()` contains only logical values
- ▶ `col_integer()` integers
- ▶ `col_double()` doubles (reals)
- ▶ `col_euro_double()` “Euro” doubles that use commas “,” as decimal separator
- ▶ `col_date()` Y-m-d dates
- ▶ `col_datetime()`: ISO8601 date times
- ▶ `col_character()`: everything else

# Column Types Correspondence

Type	Abbreviation
<code>col_logical()</code>	<code>l</code>
<code>col_integer()</code>	<code>i</code>
<code>col_numeric()</code>	<code>n</code>
<code>col_double()</code>	<code>d</code>
<code>col_euro_double()</code>	<code>e</code>
<code>col_date()</code>	<code>D</code>
<code>col_datetime()</code>	<code>T</code>
<code>col_character()</code>	<code>c</code>
<code>col_skip()</code>	<code>-</code>

# Column Types

## Overriding default choice of `col_types`

Use a compact string: `"dc__d"`. Each letter corresponds to a column so this specification means: read first column as double, second as character, skip the next two and read the last column as a double. (There's no way to use this form with column types that need parameters.)

# Column Types

## Overriding default choice of `col_types`

Another way to override the default choices of column types is by passing a list of `col_...` objects:

```
read_csv("iris.csv", col_types = list(  
  Sepal.Length = col_double(),  
  Sepal.Width = col_double(),  
  Petal.Length = col_double(),  
  Petal.Width = col_double(),  
  Species = col_factor(c("setosa", "versicolor", "virginica"))  
))
```

# Output

- ▶ Characters are never automatically converted to factors
- ▶ Column names are left as is  
(i.e. there is no `check.names = TRUE`)
- ▶ Use backticks to refer to variables with unusual names:

```
df$`Income ($000)`
```

- ▶ Row names are never set
- ▶ The output has class

```
c("tbl_df", "tbl", "data.frame")
```

## "starwarstoy.csv"

```
name,gender,height,weight,jedi,species,weapon
Luke Skywalker,male,1.72,77,jedi,human,lightsaber
Leia Skywalker,female,1.50,49,no_jedi,human,blaster
Obi-Wan Kenobi,male,1.82,77,jedi,human,lightsaber
Han Solo,male,1.80,80,no_jedi,human,blaster
R2-D2,male,0.96,32,no_jedi,droid,unarmed
C-3P0,male,1.67,75,no_jedi,droid,unarmed
Yoda,male,0.66,17,jedi,yoda,lightsaber
Chewbacca,male,2.28,112,no_jedi,wookiee,bowcaster
```

# String Columns as factors

By default, functions in "readr" do not convert character strings into factors. But you can specify what columns to be imported as factors (you must specify the levels):

```
sw1 <- read_csv(  
  file = "starwarstoy.csv",  
  col_types = list(  
    gender = col_factor(c("male", "female"))  
  )  
)
```

# Importing selected columns

"readr" allows you to import specific columns of a dataset

```
# importing just first 4 columns  
sw4 <- read_csv(  
  file = "starwarstoy.csv",  
  col_types = "ccnn___"  
)
```



# Main functions in "readr"

- ▶ `read_table()`
- ▶ `read_delim()`
- ▶ `read_csv()`
- ▶ `read_csv2()`
- ▶ `read_tsv()`
- ▶ `read_fwf()`

# Foreign Files

## Data Table (foreign files)

It is not uncommon to have tabular datasets in foreign files (e.g. from other programs)

## Files from other programs

Type	Package	Function
Excel	"gdata"	read.xls()
Excel	"xlsx"	read.xlsx()
Excel	"readxl"	read_excel()
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()