## **Reading Data**

There are a few principal functions reading data into R.

- read.table, read.csv, for reading tabular data
- · readLines, for reading lines of a text file
- source, for reading in R code files (inverse of dump)
- dget, for reading in R code files (inverse of dput)
- load, for reading in saved workspaces
- unserialize, for reading single R objects in binary form

# **Writing Data**

There are analogous functions for writing data to files

- write.table
- writeLines
- $\cdot$  dump
- · dput
- · save
- · serialize

### Reading Data Files with read.table

The read.table function is one of the most commonly used functions for reading data. It has a few important arguments:

- file, the name of a file, or a connection
- header, logical indicating if the file has a header line
- sep, a string indicating how the columns are separated
- colClasses, a character vector indicating the class of each column in the dataset
- nrows, the number of rows in the dataset
- comment.char, a character string indicating the comment character
- skip, the number of lines to skip from the beginning
- stringsAsFactors, should character variables be coded as factors?

#### read.table

For small to moderately sized datasets, you can usually call read.table without specifying any other arguments

```
data <- read.table("foo.txt")</pre>
```

#### R will automatically

- skip lines that begin with a #
- figure out how many rows there are (and how much memory needs to be allocated)
- figure what type of variable is in each column of the table Telling R all these things directly makes R run faster and more efficiently.
- · read.csv is identical to read.table except that the default separator is a comma.