

- `dumping` and `dputing` are useful because the resulting textual format is edit-able, and in the case of corruption, potentially recoverable.
- Unlike writing out a table or csv file, `dump` and `dput` preserve the *metadata* (sacrificing some readability), so that another user doesn't have to specify it all over again.
- Textual formats can work much better with version control programs like subversion or git which can only track changes meaningfully in text files
- Textual formats can be longer-lived; if there is corruption somewhere in the file, it can be easier to fix the problem
- Textual formats adhere to the “Unix philosophy”
- Downside: The format is not very space-efficient

# dput-ting R Objects

Another way to pass data around is by deparsing the R object with `dput` and reading it back in using `dget`.

```
> y <- data.frame(a = 1, b = "a")
> dput(y)
structure(list(a = 1,
               b = structure(1L, .Label = "a",
                             class = "factor")),
          .Names = c("a", "b"), row.names = c(NA, -1L),
          class = "data.frame")
> dput(y, file = "y.R")
> new.y <- dget("y.R")
> new.y
  a b
1 1 a
```

# Dumping R Objects

Multiple objects can be deparsed using the `dump` function and read back in using `source`.

```
> x <- "foo"
> y <- data.frame(a = 1, b = "a")
> dump(c("x", "y"), file = "data.R")
> rm(x, y)
> source("data.R")
> y
  a b
1 1 a
> x
[1] "foo"
```

# Interfaces to the Outside World

Data are read in using *connection* interfaces. Connections can be made to files (most common) or to other more exotic things.

- `file`, opens a connection to a file
- `gzipfile`, opens a connection to a file compressed with gzip
- `bzfile`, opens a connection to a file compressed with bzip2
- `url`, opens a connection to a webpage

```
> str(file)
function (description = "", open = "", blocking = TRUE,
         encoding = getOption("encoding"))
```

- description is the name of the file
- open is a code indicating
  - “r” read only
  - “w” writing (and initializing a new file)
  - “a” appending
  - “rb”, “wb”, “ab” reading, writing, or appending in binary mode (Windows)

In general, **connections** are powerful tools that let you navigate files or other external objects. In practice, we often don't need to deal with the connection interface directly.

```
con <- file("foo.txt", "r")  
data <- read.csv(con)  
close(con)
```

is the same as

```
data <- read.csv("foo.txt")
```

# Reading Lines of a Text File

The `readLines` function can be used to simply read lines of a text file and store them in a character vector.

```
> con <- gzfile("words.gz")
> x <- readLines(con, 10)
> x
[1] "1080"      "10-point" "10th"      "11-point"
[5] "12-point"  "16-point"  "18-point"  "1st"
[9] "2"         "20-point"
```

`writeLines` takes a character vector and writes each element one line at a time to a text file.

# Reading Lines of a Text File

`readLines` can be useful for reading in lines of webpages

```
## This might take time
```

```
con <- url("http://www.jhsph.edu", "r")
```

```
x <- readLines(con)
```

```
> head(x)
```

```
[1] "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">"
```

```
[2] ""
```

```
[3] "<html>"
```

```
[4] "<head>"
```

```
[5] "\t<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\">"
```