Textual Formats

- `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"
```