# 3. Vector (continued) and List

## Vector (continued)

### Sorting functions

`sort(x, decreasing = FALSE)` returns a sorted copy of `x`. e.g. `x = c(12, 11, 16, 11); sort(x)`

`v = rank(x, ties.method = "average")`: `v[i]` is rank of `x[i]`; also try `ties.method = "first"`

`v = order(x, ..., decreasing = FALSE)`: `v[i]` is index of $i$th smallest value in `x`, so `x[v]` is sorted. (`order()` sorts data frames, coming soon. More vectors may be given in `...` to break ties.)

### Structure, summary, quantile

`str(object)` displays the structure of `object`; e.g. `str(x)`

`summary(object)` summarizes it; e.g. `s = summary(x)`

`v = quantile(x, probs = seq(0, 1, 0.25))`: `v[i]` is the quantile corresponding to `probs[i]`

### NULL and NA special values

`NULL` is the *null object* returned by expressions and functions whose value is undefined; e.g. `names(x)`

`NA` ("not available") indicates a missing data value. `NA` propagates in calculations. e.g.

```
x[3] = NA; sum(x); sum(x, na.rm=TRUE) # "na.rm=FALSE" is a common function argument
```

Test for these values with `is.null()` and `is.na()`, not with "`== NULL`" or "`== NA`"

### (Vector) File input/output

`scan(file = "", what = numeric())` reads a vector of `numeric` from `file`, which defaults to `stdin` (the console). `what` options include `logical()`, `numeric()`, and `character()`.

`write(x, file = "data")` writes vector `x` to `file` (which defaults to `"data"`). e.g.

```
fifties = 50:59; write(x=fifties, file="50s.txt"); y = scan(file="50s.txt", what=integer())
```

## List

A list is an ordered collection of components not necessarily of the same type. e.g.

```
x = list("rope", 72, c(5, 7, 10), TRUE); length(x)
```

List components can be named via name=value pairs in `list()` or via `names()`.

```
y = list(self = "John",
         spouse = "Michele",
         kids.ages = c(0, 2, 5, 7, 9, 11)
        )
```

### List indexing with single brackets returns a sub-list

(Recall: vector indexing with `[...]` returns a sub-vector.) The index can be a vector of `integer` (select), negative `integer` (exclude), `logical` (select `TRUE`), or `character` names (select). e.g.

```
str(y[2]); str(y[2:3]); str(y[c(-2, -3)]); str(y[c("spouse", "kids.ages")])
```

### Indexing with double brackets or $ returns one component, dropping names

e.g. `x[4]` vs. `x[[4]]`

e.g. `str(y[[2]]); str(y[["spouse"]])`

The `$` operator used in the form `list.name$component.name` is the same as `list.name[[component.name]]`, except that `$` doesn't allow a computed index.

e.g. `y$spouse; y$kids.ages; y$kids.ages[3]`

e.g. `z = "spouse"; y[[z]]; y$z`

Add a component to a list by assigning it:

```
y$kids.names = c("Teresa", "Margaret", "Monica", "Andrew", "Mary", "Philip")
```

Remove a component from a list by setting it to `NULL`: `y$self = NULL`

e.g. Sort names alphabetically; then sort ages to keep up:

```
indices.ordered.by.name = order(y$kids.names); indices.ordered.by.name
(names.sorted = y$kids.names[indices.ordered.by.name]) # (...) calls print(...)
(ages.sorted.by.name = y$kids.ages[indices.ordered.by.name])
```

### Convert a list to a vector

`unlist(x, use.names=TRUE)` simplifies a list `x` to a vector (where possible). e.g. `unlist(y)`

A *data frame*, coming soon, is ($\approx$) a list of equal-length vectors (like a spreadsheet).