R Data Types

**Character vectors**

**Character/string** – each element in the vector is a string of one or more characters.

Built in character vectors are letters and LETTERS which provide the 26 lower (and upper) case letters, respectively.

```r
> y = c("a", "bc", "def")
> length(y)
[1] 3
> nchar(y)
[1] 1 2 3
> y == "a"
[1] TRUE FALSE FALSE
> y == "b"
[1] FALSE FALSE FALSE
```

**Factor**

A **factor**- type vector contains a set of numeric codes with character-valued levels.

Example - a family of two girls (1) and four boys (0),

```r
> kids = factor(c(1,0,1,0,0,0), levels=c(0,1), labels=c("boy", "girl"))
> kids
[1] girl boy girl boy boy
Levels: boy girl
> class(kids)
[1] "factor"
> mode(kids)
[1] "numeric"
```
Coercion

- All elements in a vector must be of the same type.
- R coerces the elements to a common type, in this case numeric. 1.2, 3, and 1.

```r
> x = c(TRUE, FALSE, TRUE)
> c(1.2, x)
[1] 1.2 1.0 0.0 1.0

> y = c("2", "3", ".2")
> c(1.2, y)
[1] "1.2" "2" "3" ".2" "TRUE" "FALSE" "TRUE"
```

- Sometimes this coercion occurs in order to perform an arithmetic operation:

```r
> 1 + x
[1] 2 1 2
```

- Other times we need to perform the coercion:

```r
> c(1.2, y)
[1] "1.2" "2" "3" ".2"

> c(1.2, as.numeric(y))
[1] 1.2 2.0 3.0 0.2
```

Functions to Provide Information about Vectors

- `length(x)` - number of elements in a vector or list
- Aggregator functions - `sum`, `mean`, `range`, `min`, `max`, `summary`, `table`, `cut`, ...
- `class(x)` - returns the type of an object.
- `is.logical(x)` - tells us whether the object is a logical type. There is also `is.numeric`, `is.character`, `is.integer`
- `is.null` - determines whether an object is empty, i.e. has no content. 'NULL' is used mainly to represent the lists with zero length, and is often returned by expressions and functions whose value is undefined.
- `is.na` - NA represents a value that is not available.

```r
> x
[1] 3 1 NA

> is.na(x)
[1] FALSE FALSE TRUE
```

- `as.numeric(x)` - we use the as-type functions to coerce objects from one type (e.g. logical) to another, in this case numeric. There are several of these functions, including `as.integer`, `as.character`, `as.logical`, as.POSIXct.

Logical Operators

Logical operators are extremely useful in subsetting vectors and in controlling program flow. We will cover these ideas soon.

- The usual arithmetic operators return logicals `>`, `<`, `>=`, `<=, ==`, and `!=`.
- Work element-wise on the two inputs.
- Output is a vector of logical elements (TRUE and FALSE) where the elements correspond to the test on the relevant elements on the inputs.
- Conditions can be combined with the use of `&` for AND and `|` for OR.
- `any()` returns TRUE if any of the values are TRUE, and `all()` returns TRUE if all of the values are TRUE.

Missing Values

- NA is different from 99999 or -8, which are numeric values that have special meaning in a particular context.
- NA is a recognized element in R.
- Functions have special actions when they encounter values of NA, and may have arguments to control the handling of NAs.

```r
> mean(x)
[1] NA

> mean(x, na.rm = TRUE)
[1] 2
```

- Note that NA is not a character value. In fact, it has meaning for character vectors too. `y = c("A", "d", NA, "ab", "NA")`
- Notice that the two uses, NA and "NA" mean very different things. The first is an NA value and the second is a character string.
- `na.omit()`, `na.exclude()`, and `na.fail()` are for dealing manually with NAs in a dataset.
The object x versus the character string "x"

> x = c(a = 3, z = 7, l, 2)
> y = c("a", "bc", "NA")
> z = c(TRUE, FALSE, FALSE, TRUE)

What is the return value for each of the following expressions?

- nchar(y)
- nchar("x")
- x + 2
- x + z
- c(x, NA)
- c(x, "NA")
- x[z]
- x["z"]
- x[x]
- is.na(y)
- is.na(x[x])

Return values

> nchar(y)
[1] 1 2 2
> nchar("y")
[1] 1
> x + 2
a z
5 9 3 4
> x + z
a z
4 7 1 3
> c(x, NA)
a z
3 7 1 2 NA
> c(x, "NA")

Vectors, Matrices, Arrays, Lists, and Data Frames

Vector – a collection of ordered homogeneous elements.

We can think of matrices, arrays, lists and data frames as deviations from a vector. The deviations are related to the two characteristics order and homogeneity.

Matrix - a vector with two-dimensional shape information.

> xx = matrix(1:6, nrow=3, ncol =2)
> xx
[,1] [,2]
[1,] 1 4
[2,] 2 5
[3,] 3 6
> class(x)
[1] "numeric"
> is.vector(xx)
[1] TRUE
> is.matrix(xx)
[1] TRUE
> length(xx)
[1] 6
> dim(xx)
[1] 3 2

The object x versus the character string "x"
Lists

A vector with possible heterogeneous elements. The elements of a list can be numeric vectors, character vectors, matrices, arrays, and lists.

\[
\text{myList} = \text{list}(a = 1:10, b = "def", c(\text{TRUE}, \text{FALSE}, \text{TRUE}))
\]

\[
\begin{array}{l}
\text{a} \\
[1] 1 2 3 4 5 6 7 8 9 10 \\
\text{b} \\
[1] "def" \\
[[3]] \\
[1] \text{TRUE} \text{ FALSE} \text{ TRUE}
\end{array}
\]

- \text{length(myList)} – there are 3 elements in the list
- \text{class(myList)} – the class is a “list”
- \text{names(myList)} – are “a”, “b” and the empty character “”
- \text{myList[1:2]} – returns a list with two elements
- \text{myList[1]} – returns a list with one element. What is \text{length(myList[1])}?
- \text{myList[[1]]} – returns a vector with ten elements, the numbers 1, 2, ..., 10 What is \text{length(myList[[1]])}?

Data Frames

A list with possible heterogeneous vector elements of the same length. The elements of a data frame can be numeric vectors, factor vectors, and logical vectors, but they must all be of the same length.

\[
\text{> intel} \\
\begin{array}{ccccccccc}
\text{Date} & \text{Transistors} & \text{Microns} & \text{Clockspeed} & \text{speed} & \text{Data} & \text{MIPS} \\
8080 & 1974 & 6000 & 6.00 & 2.0 & \text{MHz} & 8 & 0.64 \\
8088 & 1979 & 29000 & 3.00 & 5.0 & \text{MHz} & 16 & 0.33 \\
80286 & 1982 & 134000 & 1.50 & 6.0 & \text{MHz} & 16 & 1.00 \\
80386 & 1985 & 275000 & 1.50 & 16.0 & \text{MHz} & 32 & 5.00 \\
80486 & 1989 & 1200000 & 1.00 & 25.0 & \text{MHz} & 32 & 20.00 \\
\text{Pentium} & 1993 & 3100000 & 0.80 & 60.0 & \text{MHz} & 32 & 100.00 \\
\text{PentiumII} & 1997 & 7500000 & 0.35 & 233.0 & \text{MHz} & 32 & 300.00 \\
\text{PentiumIII} & 1999 & 9500000 & 0.25 & 450.0 & \text{MHz} & 32 & 510.00 \\
\text{Pentium4} & 2000 & 42000000 & 0.18 & 1.5 & \text{GHz} & 32 & 1700.00 \\
\text{Pentium4x} & 2004 & 125000000 & 0.09 & 3.6 & \text{GHz} & 32 & 7000.00 \\
\end{array}
\]
**Subsetting a Data Frame**

Using the fact that a data frame is a list which also support some matrix features, fill in the table specifying the **class** (data.frame or integer) and the **length** and **dim** of the subset of the data frame. Note that some responses will be NULL.

<table>
<thead>
<tr>
<th>Subset</th>
<th>class</th>
<th>length</th>
<th>dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>intel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel[1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel[[1]]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel[,1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel[&quot;Date&quot;]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel[, &quot;Date&quot;]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IntelSDate</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>