LECTURE 4: DATA STRUCTURES IN R (contd)

STAT598z: Intro. to computing for statistics

Vinayak Rao

Department of Statistics, Purdue University

Data frames

Very common and convenient data structures

Used to store tables:

- Columns are variables and rows are observations

	Age	**PhD**	**GPA**
Alice	25	TRUE	3.6
Bob	24	TRUE	3.4
Carol	21	FALSE	3.8

An R data frame is a list of equal length vectors

In [6]:
```
df <- data.frame(age = c(25L,24L,21L),  # Warning: df is an
                 PhD = c( T , T , F ),  #   R function
                 GPA = c(3.6,2.4,2.8))
```

In [7]:
```
print(df)
```

```
  age   PhD GPA
1  25  TRUE 3.6
2  24  TRUE 2.4
3  21 FALSE 2.8
```

In [3]:
```
typeof(df)
```

'list'

In [4]:
```
class(df)
```

'data.frame'

Since data frames are lists, we can use list indexing

Can also use matrix indexing (more convenient)

In [9]:
```
print(df[2,'age'])
```

```
[1] 24
```

In [10]:
```
print(df[2,])
```

```
  age  PhD GPA
2  24 TRUE 2.4
```

In [11]:
```
print(df$GPA)
```

```
[1] 3.6 2.4 2.8
```

In [14]:
```
nrow(df)*ncol(df)
```

```
9
```

list functions apply as usual

matrix functions are also interpreted intuitively

Useful functions are:

- 'length(), dim(), nrow(), ncol()'
- 'names()' (or 'colnames()')', rownames'
- 'rbind(), cbind()'

In [15]:
```
rownames(df) <- c("Alice", "Bob", "Carol")
```

In [18]:
```
df[4,1] <- 30L; print(df)
```

```
      age   PhD GPA
Alice  25  TRUE 3.6
Bob    24  TRUE 2.4
Carol  21 FALSE 2.8
4      30    NA  NA
```

Many R datasets are data frames

In [ ]:
```
library("datasets")
class(mtcars)
```

In [ ]:
```
print(head(mtcars)) # Print part of a large object
```

Factors

Categorical variables that take on a finite number of values

- **Employee type**: `student/staff/faculty`
- **Grade**: `A/B/C/F`

Useful when variable can take a fixed set of values (unlike character strings)

R implements these internally as integer vectors

Has two attributes to distinguish from regular integers:

`levels()` specifies possible values the factor can take

- E.g. `c("male", "female")`

`class = factor` tells R to check for violations

In [ ]:
```
# Character vector for 4 students
grades_bad <- c("a", "a", "b", "f")
```

In [ ]:
```
# Factor vector for 4 students
grades <- factor(c("a", "a", "b", "f"))
```

In [ ]:
```
print(grades)
```

In [ ]:
```
typeof(grades)
```

In [ ]:
```
class(grades)
```

In [ ]:
```
levels(grades) # Not quite what we wanted!
```

In [ ]:
```
grades <- factor(c("a", "a", "b", "f"))
str(grades)
```

In [ ]:
```
grades[2] <- "c"
```

In [ ]:
```
str(grades)
```

In [ ]:
```
grades <- factor(c("a","a","b","a","f"),
                 levels = c("a","b","c","f"))
```

In [ ]:
```
str(grades)
```

In [ ]:
```
table(grades)   # table also works with other data-types
```

Factors can be ordered:

In [ ]:
```
grades <- factor(c("a","a","b","f"),
            levels = c("f","c","b","a"),
            ordered = TRUE )
grades
```

In [ ]:
```
grades[1] > grades[3]
```

`gl()`: Generate factors levels

Usage (from the R documentation):

```
gl(n, k, length = n * k, labels = seq_len(n),
   ordered = FALSE )
```

Look at the examples there:

In [ ]:
```
# First control, then treatment:
gl(2, 8, labels = c("Control", "Treat"))
```

In [ ]:
```
gl(2, 1, 20) # 20 alternating 1s and 2s
```

In [ ]:
```
gl(2, 2, 20) # alternating pairs of 1s and 2s
```

An aside on assignment

From the R language definition:

`x[3 : 5] <- 13 : 15`

is as if the following had been executed

```
'*tmp*' <- x  # Don't use your own *tmp* variables!
x       <- "[<-"('*tmp*', 3 : 5, value = 13 : 15)
rm('*tmp*')
# ls() lists all objects in current session
```

From the R language definition:

`names(x) <- c("a","b")`

is equivalent to

```
'*tmp*' <- x
  x <- "names<-"('*tmp*', value = c("a","b"))
  # Note names<-
  rm('*tmp*')
```