Lists, Factors and Data Frames

Lists

Up to this points, all the data objects described are *atomic*, that is they contain data of only one mode. However, there are several instances that we may want to create data objects that contain mixed modes and also preserve the mode of each value. The solution is offered by the *lists* who made up of components, where each component consists of one data object, of any type.

```
> group1 <- c(rep(1,11), rep(2,17))
> group2 <- c(23,45,67,76,-8,3.5,2.19,4)
> groups <- list(case=group1, control=group2, descrip="An example")
> groups
$case:
 [1] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

$control:
[1]  23.00  45.00  67.00  76.00  -8.00   3.50   2.19   4.00

$descrip:
[1] "An example"
```

To extract a list component use `$` or `[[ ]]`.

```
> groups$case
 [1] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
> groups$control
[1]  23.00  45.00  67.00  76.00  -8.00   3.50   2.19   4.00
> groups[[1]]
 [1] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
> groups[[2]]
[1]  23.00  45.00  67.00  76.00  -8.00   3.50   2.19   4.00
> groups[[2]][1:2]
[1] 23 45
```

```
> length(groups)
[1] 3
> mode(groups)
[1] "list"
> names(groups)
[1] "case"    "control" "descrip"
```

Factors

For data analysis purposes, many types of variables are qualitative or categorical. Some examples include

- *gender* with values male and female,
- *marital status* where the values are single, married separated, divorced.

Categorical data in `S-Plus` are represented by a data object called factor. To create a factor use the `factor` function. Here are some examples:

```
> gender <- c("male", "female", "male", "male", "female", "female", "male")
> gender
[1] "male"   "female" "male"   "male"   "female" "female" "male"
> factor(gender)
[1] male   female male   male   female female male
> intensity <- factor(c("Hi", "Med", "Lo", "Hi", "Hi", "Lo"),
+ levels=c("Hi","Lo"))
> intensity
[1] Hi NA Lo Hi Hi Lo
> levels(intensity)
[1] "Hi" "Lo"
> intensity <- factor(c("Hi", "Med", "Lo", "Hi", "Hi", "Lo"),
+ levels=c("Hi","Lo"), labels=c("HighDose", "LowDose"))
> intensity
[1] HighDose NA       LowDose  HighDose HighDose LowDose
```

If the order is important we might use the function `ordered`.

```
> intensity <- ordered(c("Hi", "Med", "Lo", "Hi", "Hi", "Lo"),
+ levels=c("Lo", "Med", "Hi"))
> intensity
[1] Hi  Med Lo  Hi  Hi  Lo

 Lo < Med < Hi
```

Here is how you can use the function `cut` to create a factor from a continuous variable.

```
> fact <- rnorm(x)
> fact <- cut(fact, breaks=c(-5,-1,1,2,4))
> fact
[1] 2 3 2 1
attr(, "levels"):
[1] "-5+ thru -1" "-1+ thru  1" " 1+ thru  2" " 2+ thru  4"
> fact2 <- cut(fact, breaks=5)
> fact2
[1] 3 5 3 1
attr(, "levels"):
[1] "0.980+ thru 1.388" "1.388+ thru 1.796" "1.796+ thru 2.204" "2.204+ thru 2.612"
[5] "2.612+ thru 3.020"
```

Some other things useful for factors are the following:

```
> length(intensity)
[1] 6
> mode(intensity)
[1] "numeric"
> names(intensity)
NULL
> levels(intensity)
[1] "Lo"  "Med" "Hi"
> class(intensity)
[1] "ordered" "factor"
```

Data Frames

The main benefit of a data frame is that is allows you to mix data of different types into a single object in preparation for analysis and modelling. The idea of a data frame is to group data by variables (columns) regardless of their type. Then all the observations on a particular set of variables can be grouped into a data frame. For instance consider

```
> solder[test,]
    Opening Solder Mask PadType Panel skips
713       S   Thin   B3      L8     2    28
652       L   Thin   B3      L8     1     1
793       S  Thick   B6      L6     1     7
372       L  Thick   A3      D6     3     0
200       L  Thick   A3      L7     2     0
725       L  Thick   B6      D4     2     0
495       M   Thin   A6      L6     3     6
364       L  Thick   A3      D4     1     0
499       M   Thin   A6      L7     1     4
```

```
782       S  Thick   B6      W4     2    10
 29       L  Thick A1.5      L9     2     0
196       L  Thick   A3      D7     1     0
724       L  Thick   B6      D4     1     1
```

13 observations from the built-in data set `solder`. The variable `skips` is continuous while all the rest a various factors.

There are several ways to create a data frame:

- `read.table` reads in data from an external file,
- `data.frame` puts together objects of various kinds.
- `as.data.frame` coerces objects of a particular type to objects of class `data.frame`.

We will only examine the second option at this point.

```
 my.logic <- sample(c(T,F), size=20, replace=T)
> my.logic
 [1] T T T F F T F T F F F F T F T T F F F T
> my.complex <- rnorm(20)+runif(20)*1i
> my.complex
 [1]  2.48467422+0.48265416i -1.60470965+0.22767635i  1.35172992+0.50010095i
 [4] -0.58286780+0.12268995i -0.48598155+0.78922205i  0.96350421+0.26461911i
 [7] -0.56341008+0.65644492i  1.32382209+0.04703269i -0.87364793+0.79261444i
[10] -1.70070057+0.29504429i -1.42179049+0.87250394i  0.79639782+0.41410611i
[13] -0.24871898+0.36109209i -0.82794923+0.58787154i  0.74958274+0.18333409i
[16]  1.09769715+0.82699845i -2.23353769+0.88747224i  0.06592538+0.44815591i
[19] -0.31559966+0.50181774i -1.24223783+0.67073653i
> my.numeric  <- rnorm(20)
> my.numeric
 [1] -0.282444864  0.189648235  0.009124648 -1.957229831 -0.843219234 -1.142369140
 [7] -1.625863978 -0.593153260 -0.277026588 -1.036525482 -0.194730628 -0.952137384
[13]  0.372314331 -0.345667301  0.066380799 -0.712903265 -1.673258866 -0.646916651
[19] -0.446425318  0.489922573
> my.matrix <- matrix(rnorm(40), ncol=2)
> my.matrix
              [,1]        [,2]
 [1,]  0.449102068  0.24017164
 [2,]  0.008685083  0.87770835
 [3,] -0.047232037  0.22342894
 [4,]  1.693031734 -1.80726536
 [5,]  0.749590583 -2.61520830
 [6,]  0.186438048 -0.18790145
 [7,]  1.056075476 -0.13910055
 [8,] -0.181090548 -0.06089259
 [9,] -0.087113944  0.34434902
[10,] -0.388402911 -2.09778420
```

```
[11,]  1.238077005 -1.14091719
[12,] -0.184444523  1.90571136
[13,] -0.262926763  0.25632422
[14,]  0.085871143  1.56097070
[15,] -1.404724839  0.33637390
[16,] -0.075153038  1.33265693
[17,]  0.723236165 -0.45932412
[18,]  1.474432351 -0.18350542
[19,]  1.485927348  1.83562547
[20,] -0.109916665  0.63651418
> my.dataframe <- data.frame(my.logic, my.complex, my.numeric, my.matrix)
> my.dataframe
   my.logic              my.complex   my.numeric  my.matrix.1 my.matrix.2
 1     TRUE  2.48467422+0.48265416i -0.282444864  0.449102068  0.24017164
 2     TRUE -1.60470965+0.22767635i  0.189648235  0.008685083  0.87770835
 3     TRUE  1.35172992+0.50010095i  0.009124648 -0.047232037  0.22342894
 4    FALSE -0.58286780+0.12268995i -1.957229831  1.693031734 -1.80726536
 5    FALSE -0.48598155+0.78922205i -0.843219234  0.749590583 -2.61520830
 6     TRUE  0.96350421+0.26461911i -1.142369140  0.186438048 -0.18790145
 7    FALSE -0.56341008+0.65644492i -1.625863978  1.056075476 -0.13910055
 8     TRUE  1.32382209+0.04703269i -0.593153260 -0.181090548 -0.06089259
 9    FALSE -0.87364793+0.79261444i -0.277026588 -0.087113944  0.34434902
10    FALSE -1.70070057+0.29504429i -1.036525482 -0.388402911 -2.09778420
11    FALSE -1.42179049+0.87250394i -0.194730628  1.238077005 -1.14091719
12    FALSE  0.79639782+0.41410611i -0.952137384 -0.184444523  1.90571136
13     TRUE -0.24871898+0.36109209i  0.372314331 -0.262926763  0.25632422
14    FALSE -0.82794923+0.58787154i -0.345667301  0.085871143  1.56097070
15     TRUE  0.74958274+0.18333409i  0.066380799 -1.404724839  0.33637390
16     TRUE  1.09769715+0.82699845i -0.712903265 -0.075153038  1.33265693
17    FALSE -2.23353769+0.88747224i -1.673258866  0.723236165 -0.45932412
18    FALSE  0.06592538+0.44815591i -0.646916651  1.474432351 -0.18350542
19    FALSE -0.31559966+0.50181774i -0.446425318  1.485927348  1.83562547
20     TRUE -1.24223783+0.67073653i  0.489922573 -0.109916665  0.63651418
```

We can also use `cbind` and `rbind` to create a data frame together with many other options.

```
 my.dataframe2 <- cbind(1, my.dataframe)
> my.dataframe2
   X1 my.logic              my.complex   my.numeric  my.matrix.1 my.matrix.2
 1  1     TRUE  2.48467422+0.48265416i -0.282444864  0.449102068  0.24017164
 2  1     TRUE -1.60470965+0.22767635i  0.189648235  0.008685083  0.87770835
 3  1     TRUE  1.35172992+0.50010095i  0.009124648 -0.047232037  0.22342894
 4  1    FALSE -0.58286780+0.12268995i -1.957229831  1.693031734 -1.80726536
 5  1    FALSE -0.48598155+0.78922205i -0.843219234  0.749590583 -2.61520830
 6  1     TRUE  0.96350421+0.26461911i -1.142369140  0.186438048 -0.18790145
 7  1    FALSE -0.56341008+0.65644492i -1.625863978  1.056075476 -0.13910055
 8  1     TRUE  1.32382209+0.04703269i -0.593153260 -0.181090548 -0.06089259
 9  1    FALSE -0.87364793+0.79261444i -0.277026588 -0.087113944  0.34434902
10  1    FALSE -1.70070057+0.29504429i -1.036525482 -0.388402911 -2.09778420
11  1    FALSE -1.42179049+0.87250394i -0.194730628  1.238077005 -1.14091719
12  1    FALSE  0.79639782+0.41410611i -0.952137384 -0.184444523  1.90571136
13  1     TRUE -0.24871898+0.36109209i  0.372314331 -0.262926763  0.25632422
14  1    FALSE -0.82794923+0.58787154i -0.345667301  0.085871143  1.56097070
15  1     TRUE  0.74958274+0.18333409i  0.066380799 -1.404724839  0.33637390
16  1     TRUE  1.09769715+0.82699845i -0.712903265 -0.075153038  1.33265693
17  1    FALSE -2.23353769+0.88747224i -1.673258866  0.723236165 -0.45932412
18  1    FALSE  0.06592538+0.44815591i -0.646916651  1.474432351 -0.18350542
19  1    FALSE -0.31559966+0.50181774i -0.446425318  1.485927348  1.83562547
20  1     TRUE -1.24223783+0.67073653i  0.489922573 -0.109916665  0.63651418
```

Some other commands that might be useful are as follows

```
> length(my.dataframe)
[1] 5
> dim(my.dataframe)
[1] 20  5
> is.data.frame(my.dataframe)
[1] T
> is.list(my.dataframe)
[1] T
> is.matrix(my.dataframe)
[1] T
> is.vector(my.dataframe)
[1] F
> names(my.dataframe)
[1] "my.logic"    "my.complex"  "my.numeric"  "my.matrix.1" "my.matrix.2"
```

The functions `attach` and `detach` are very useful when we are working with a specific data frame. The statement

```
 attach(my.dataframe)
```

places the data frame in the search list at position 2 and therefore we can work directly with the variables of the data frame.

```
> my.logic
 [1] TRUE  TRUE  TRUE  FALSE FALSE TRUE  FALSE TRUE  FALSE FALSE FALSE FALSE TRUE
[14] FALSE TRUE  TRUE  FALSE FALSE FALSE TRUE
> my.complex
                   1                   2                  3                     4
 2.484674+0.4826542i -1.60471+0.2276764i 1.35173+0.5001009i -0.5828678+0.1226899i
                    5                    6                     7                    8
 -0.4859815+0.789222i 0.9635042+0.2646191i -0.5634101+0.6564449i 1.323822+0.04703269i
                     9                   10                 11                   12
 -0.8736479+0.7926144i -1.700701+0.2950443i -1.42179+0.8725039i 0.7963978+0.4141061i
                   13                    14                   15                  16
 -0.248719+0.3610921i -0.8279492+0.5878715i 0.7495827+0.1833341i 1.097697+0.8269985i
                   17                    18                    19
 -2.233538+0.8874722i 0.06592538+0.4481559i -0.3155997+0.5018177i
                   20
 -1.242238+0.6707365i
```

To detach a data frame use the function `detach`.