STAT 210
Applied Statistics and Data Analysis
Objects and Data III: Data frames and lists

Joaquín Ortega

joaquin.ortegasanchez@kaust.edu.sa

Data Frames

Data Frames

Data frames are matrix-like structures whose columns may be of different `mode` but must all have the same `length`.

The command for creating a data frame is

```
data.frame (data1, data2, ...)
```

Data Frames

For instance:

```
(dframe1 <- data.frame(int = 2001:2010,
                       real = rnorm(10),
                       lettrs = letters[sample.int(26,10)],
                       logic = (1:10)%%2==0))
```

```
##      int         real lettrs logic
## 1   2001  0.02823064      m FALSE
## 2   2002  0.66781926      z  TRUE
## 3   2003  0.27891774      y FALSE
## 4   2004  0.62305267      d  TRUE
## 5   2005 -1.55240067      g FALSE
## 6   2006  0.52271553      l  TRUE
## 7   2007 -0.49720615      r FALSE
## 8   2008 -1.01357500      n  TRUE
## 9   2009  1.28765682      o FALSE
## 10 2010  0.62229632      p  TRUE
```

Data Frames

However, if we try to add a new column with only three elements, an error message is produced:

```
dframe1 <- data.frame(int = 2001:2014,
                      numbs = rnorm(14),
                      real = letters[sample.int(26,14)],
                      logic = (1:14)%%2==0,
                      v = c(1,-1,2))
```

```
## Error in data.frame(int = 2001:2014, numbs = rnorm(14), real = lette
```

Data Frames

The variables in a data frame can be accessed using the following instructions

```
dframe1$int
```

```
##  [1] 2001 2002 2003 2004 2005 2006 2007 2008 2009
## [10] 2010
```

```
dframe1['int']
```

```
##      int
## 1   2001
## 2   2002
## 3   2003
## 4   2004
## 5   2005
## 6   2006
## 7   2007
## 8   2008
## 9   2009
## 10 2010
```

Data Frames

Observe, however, that they have different structures

```
str(dframe1$int)
```

```
##  int [1:10] 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
```

```
str(dframe1['int'])
```

```
## 'data.frame':    10 obs. of  1 variable:
##  $ int: int  2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
```

In the first case, we have a (row) vector with 14 components, while in the second, the result is a data frame with one variable.

Data Frames

This syntax to access a variable in a data frame is cumbersome.

When the information on a data frame is used frequently, a convenient resource is the function `attach(object)`, which places the object in a preferent place of the search path.

Let us first define a vector `lettrs` with the five vowels as components

```
lettrs <- c('a','e','i','o','u')
attach(dframe1)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     lettrs
```

Data Frames

```
int
```

```
##  [1] 2001 2002 2003 2004 2005 2006 2007 2008 2009
## [10] 2010
```

```
real
```

```
##  [1]  0.02823064  0.66781926  0.27891774
##  [4]  0.62305267 -1.55240067  0.52271553
##  [7] -0.49720615 -1.01357500  1.28765682
## [10]  0.62229632
```

```
lettrs
```

```
## [1] "a" "e" "i" "o" "u"
```

Data Frames

Observe that we get a value for `lettrs` is different from that in `dframe1`.

The reason for this is that there exists a variable with the same name in the working directory, and local variables precede those obtained from the command `attach()`.

This was alerted by the warning we got when using the `attach(dframe1)` command. To further explore this issue consider the following commands

```
ls()
```

```
## [1] "dframe1" "lettrs"
```

```
rm(lettrs)
lettrs
```

```
##  [1] m z y d g l r n o p
## Levels: d g l m n o p r y z
```

Data Frames

After removing object `lettrs` from the working environment, we get the correct answer from the attached data frame.

Let us now look at the effect of the `attach` function on the search path:

```
search()
```

```
##  [1] ".GlobalEnv"         "dframe1"
##  [3] "package:stats"      "package:graphics"
##  [5] "package:grDevices" "package:utils"
##  [7] "package:datasets"   "package:methods"
##  [9] "Autoloads"          "package:base"
```

Data Frames

We see that `dframe1` has been place in second position, just after `.GlobalEnv`, which is the working directory.

To eliminate it from the search path we can use the `detach()` function:

```
detach(dframe1)
search()
```

```
## [1] ".GlobalEnv"         "package:stats"
## [3] "package:graphics"   "package:grDevices"
## [5] "package:utils"      "package:datasets"
## [7] "package:methods"    "Autoloads"
## [9] "package:base"
```

It is advisable to `detach` data frames when you finish working with them, particularly when variable names may be repeated.

Lists

Lists

Lists are objects that can contain information with different modes and structures.

Recall that a data frame can contain data of different modes, but all the vectors must have the same length.

In lists, the elements can be arbitrary. Thus, a list is an ordered collection of objects.

To build lists use the command `list`:

Lists

```
(vec1 <- letters[5:10])
```

```
## [1] "e" "f" "g" "h" "i" "j"
```

```
(vec2 <- 1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
(mat3 <- matrix(rnorm(12),ncol=3))
```

```
##             [,1]       [,2]       [,3]
## [1,] -0.3862971 -0.5508705  1.5297737
## [2,] -1.6671776 -2.3867915  1.5175164
## [3,]  1.2808057 -0.4118114 -0.2309592
## [4,] -1.5148670  0.8504727  0.5013358
```

Lists

```
(mixed.list <- list(item1 = vec1, item2 = vec2,
                    item3 = mat3, item4 = cars))
```

```
## $item1
## [1] "e" "f" "g" "h" "i" "j"
##
## $item2
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $item3
##             [,1]       [,2]       [,3]
## [1,] -0.3862971 -0.5508705  1.5297737
## [2,] -1.6671776 -2.3867915  1.5175164
## [3,]  1.2808057 -0.4118114 -0.2309592
## [4,] -1.5148670  0.8504727  0.5013358
##
## $item4
##    speed dist
## 1      4    2
## 2      4   10
## 3      7    4
## 4      7   22
## 5      8   16
```

Lists

Named items in a list can be accessed in two different ways, either using the $ sign as we did for data frames or using double square brackets:

```
mixed.list$item1
```

```
## [1] "e" "f" "g" "h" "i" "j"
```

```
mixed.list[[1]]
```

```
## [1] "e" "f" "g" "h" "i" "j"
```

Lists

Double square brackets are used for getting items out of a list while single square brackets are used for extracting elements of a vector or array in the list:

```
mixed.list$item3[,2]
```

```
## [1] -0.5508705 -2.3867915 -0.4118114  0.8504727
```

```
mixed.list[[3]][,2]
```

```
## [1] -0.5508705 -2.3867915 -0.4118114  0.8504727
```

Lists

```
mixed.list$item4$dist
```

```
##  [1]  2 10  4 22 16 10 18 26 34 17 28
## [12] 14 20 24 28 26 34 34 46 26 36 60
## [23] 80 20 26 54 32 40 32 40 50 42 56
## [34] 76 84 36 46 68 32 48 52 56 64 66
## [45] 54 70 92 93 120 85
```

Lists

Lists are objects with great flexibility for storing data and other objects in `R`. The objects in a list may be completely independent and may be of different types and shapes.

Elements in a list can be assigned and changed as with other objects. To add a new element to a list, simply place it in a position that is not in use.

The `mixed.list` list has four elements (this can be determined as usual using the `length()` function). We can add a new element by assigning it to a free index.

Lists

```
length(mixed.list)
```

```
## [1] 4
```

```
mixed.list[[5]] <- c('The', 'new', 'element')
```

Any index can be used, but it is better to use the first free one since the intermediate elements are created. If we add an item with index 10 to a list of 3 items, we are automatically creating the items from 4 to 9, which are assigned the `NULL` value.

The following command adds a new element in the first free place:

```
mixed.list[[length(mixed.list)+1]] <-
  c('the last')
```

Lists

If we assign a value to an existing index, the element is replaced. To remove an existing element, assign it the value `NULL`:

```
mixed.list[[4]] <- -5:10
mixed.list[[2]] <- NULL
mixed.list
```

```
## $item1
## [1] "e" "f" "g" "h" "i" "j"
##
## $item3
##            [,1]       [,2]       [,3]
## [1,] -0.3862971 -0.5508705  1.5297737
## [2,] -1.6671776 -2.3867915  1.5175164
## [3,]  1.2808057 -0.4118114 -0.2309592
## [4,] -1.5148670  0.8504727  0.5013358
##
## $item4
##  [1] -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9
## [16] 10
##
## [[4]]
## [1] "The"     "new"     "element"
##
## [[5]]
## [1] "the last"
```

Lists

As we can see, by eliminating one element, the others move forward, and their indices are reduced by one unit.

It is impossible to remove several items from a list simultaneously, and it is also impossible to use negative indexes to delete list elements.

The function `names` gives the names of the elements in the list and can also be used to assign or change names:

Lists

```
names(mixed.list)
```

```
## [1] "item1" "item3" "item4" ""        ""
```

```
names(mixed.list)[4:5] <- c('item.new','item final')
```

```
names(mixed.list)
```

```
## [1] "item1"      "item3"      "item4"
## [4] "item.new"   "item final"
```

Lists

Lists are often used in R to store the output of a function that has some complexity.

To see an example, let's consider the result of linear regression.

The result of linear regression is an elaborate list that has stored a great deal of information about the fitted model.

Lists

```
fit1 <- lm(Fertility ~ Education, data=swiss)
str(fit1)
```

```
## List of 12
##  $ coefficients : Named num [1:2] 79.61 -0.862
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "Education"
##  $ residuals    : Named num [1:47] 10.9 11.3 17.2 12.2 10.2 ...
##   ..- attr(*, "names")= chr [1:47] "Courtelary" "Delemont" "Franches-Mnt" "Moutier" ...
##  $ effects      : Named num [1:47] -480.9 -56.2 14.9 10.2 9.4 ...
##   ..- attr(*, "names")= chr [1:47] "(Intercept)" "Education" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:47] 69.3 71.8 75.3 73.6 66.7 ...
##   ..- attr(*, "names")= chr [1:47] "Courtelary" "Delemont" "Franches-Mnt" "Moutier" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:47, 1:2] -6.856 0.146 0.146 0.146 0.146 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:47] "Courtelary" "Delemont" "Franches-Mnt" "Moutier" ...
##   .. .. ..$ : chr [1:2] "(Intercept)" "Education"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.15 1.03
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 45
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = Fertility ~ Education, data = swiss)
##  $ terms        :Classes 'terms', 'formula'  language Fertility ~ Education
##   .. ..- attr(*, "variables")= language list(Fertility, Education)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "Fertility" "Education"
##   .. .. .. ..$ : chr "Education"
##   .. ..- attr(*, "term.labels")= chr "Education"
##   .. ..- attr(*, "order")= int 1
```