Operations in R

Jan Rovny

#Objects in R ##Arrays, Vectors, Data Frames, and Matrices. R distinguishes between vectors, data frames, and matrices. Vectors are indexed by length and matrices are indexed by rows and columns. Data frames are a matrix that R designates as a data set. With a data frame, the columns of the matrix can be referred to as variables.

We can create objects directly in R:

```
a<-7 #this creates a scalar *a* with a value of 7
x<-c(1,2,3,4,5,6,7,8,9,10) #this creates a vector x with 10 values
y<-c(13,45,23,78,-5.2,4,43,8,12,-3) #this creates a vector y with 10 values
A<-matrix(              #this creates a matrix A
  c(2, 4, 3, 1, 5, 7, 5, 6, 8), # the data elements
  nrow=3,              # number of rows
  ncol=3,              # number of columns
  byrow = TRUE)        # fill matrix by rows
```

We can then ask R to show us these objects by simply calling on them:

```
a
```

```
[1] 7
```

```
x
```

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

```
y
```

```
 [1] 13.0 45.0 23.0 78.0 -5.2  4.0 43.0  8.0 12.0 -3.0
```

```
A
```

```
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
[3,]    5    6    8
```

We can also call the specific elements of matrix A:

```
A[,2] # 2nd column of matrix
```

```
[1] 4 5 6
```

```
A[3,] # 3rd row of matrix
```

```
[1] 5 6 8
```

```
A[2,2] #2nd row, 2nd column
```

```
[1] 5
```

```
A[1:2,1:3] # rows 1 to 2 of columns 1 to 3
```

```
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
```

##Datasets as objects After reading in a data set, R will treat your data as a data frame. Unlike in other statistical programs, R allows us to work with multiple data frames at the same time. This, however, means that when we wish to refer to a specific variable, we must identify it by its data frame name and variable name. We normally do this by saying:

```
data.frame$variable.name
```

As an example, let's look at a section from a French election study. Let's first load this dataset from a website.

```
library(rio)
D<-import("https://jan-rovny.squarespace.com/s/France.dta")
```

We have now created the object 'D' in our R memory. We can, for example, see which party a respondent would vote for. To do this, we print the variable 'vote1' in the dataset 'D':

```
D$vote1
```

```
  [1]  6  3 NA  3  3  3  2  3 NA NA  5  3 NA  3  1  1  3  3  3  1  6 NA  6  3
 [25]  3  4  5  1  3  3  3  1 NA  2  3 NA  3  3 NA NA NA  1  1  5 NA  1  3  3
 [49]  3  2  5  2  3 NA  1  6  1 NA NA NA NA  4  3  3  5  3  6 NA  3  1  5 NA
 [73]  1  5 NA  2  5  3  4  4  1  5  1  3 NA  5  1 NA  3  5  4 NA  1  5  3 NA
 [97]  3 NA  3  1  5  5 NA  5  1  1  1 NA NA  1  2  3 NA  1  5  1  4 NA  5 NA
[121]  1  5 NA  1 NA  1 NA  6  3  1  3  1  6 NA  3  3  5  1 NA NA  3 NA  1  1
[145] NA  1  5  1  3  2 NA  1  5  5 NA NA NA  5  1  3  4  5  1 NA  4  5  3  1
[169]  2 NA NA NA  1  1 NA  4 NA  1  5 NA  6  5  2  1  1  3  3 NA  3 NA NA NA
[193]  4  2  2  3  1  3  2  3  1 NA NA NA NA  1 NA  6  2  3  4  3 NA NA  3  4
[217]  1  5 NA  3  3  1 NA NA NA  3 NA  3  2  5  3  5 NA NA NA NA  6 NA NA  1
[241]  3 NA NA NA  5 NA  1  3  3  1 NA NA  3  5  4  6 NA  5  1  3 NA  5  3  4
[265]  1  1  1 NA  5  1  1  2  1  3 NA NA NA  1 NA  5  1  6 NA NA  1  5 NA  5
[289]  5  5 NA  1  3  3  3 NA  5  2  3  3  1  5  1 NA  3  4  3  1  5  5  4  3
[313] NA  6  2  3 NA  3  4 NA  1 NA  2  5  1 NA  3 NA NA  5  6 NA NA NA NA  3
[337]  6  5  5  2 NA NA  5  1  3  3  5  5  1  3  1  1  5  5  3 NA  3 NA  5  1
[361]  1  1  3  5 NA  5 NA  6  1 NA NA NA  1 NA  2  1  4  5  5  1 NA NA  6  1
[385]  1  1 NA NA  3  3 NA  1  6  2  1 NA  1  1 NA NA  3  1 NA  5  5  1  4 NA
[409] NA  3  2  1  3  1  1  5 NA  1  3  3 NA  5 NA  3  1  3  1  5  6  4 NA  6
[433]  3  3  3 NA NA  5 NA  1  1  1  3  6 NA NA  6  1  6 NA  5  5  3  1  5  6
[457]  2  1  3  1 NA  1 NA  3 NA  3  5  5  3  1 NA  5  3 NA  6  5 NA NA NA NA
[481]  3 NA NA  5  3  1  3 NA  5  1  5  1 NA  3  3  4 NA  3  1  6  1  1 NA NA
[505] NA  6 NA  4  3  1  2  5 NA NA  3  3 NA  3  2  5  1  2  6  1  2 NA  3 NA
[529]  1 NA NA NA  1  5  1 NA  3  4  1  3  1 NA  5  2 NA  3  6  3 NA  5  3  1
[553]  6  3  1  1 NA NA NA  4 NA  3 NA  6 NA NA NA NA NA  5 NA  1  3  5 NA  3
[577] NA  1  5 NA  3  1 NA  5  3  1 NA  5  1  3  3  5  2  2 NA  5  1  6  3  1
[601]  3  5  1  1  5  5 NA NA  1  5 NA NA  1  2 NA NA  3  1  3 NA  3  3 NA  1
[625] NA NA NA NA NA  6 NA  3 NA  5 NA NA NA  6  5  3 NA  3  3 NA  3  3  3 NA
[649] NA  3  3  3  5  2  1 NA  5 NA  4  5 NA  5 NA  5 NA NA  4 NA  3 NA NA NA
[673]  1  3  5  3  4  3  5 NA NA  1  3  4  1 NA  3  1 NA NA NA  1  5 NA  1  4
[697]  3 NA  5 NA  1  5  5  3  3 NA  3 NA  2  4  3  6  5 NA  1  5  6 NA  5 NA
[721]  1  1  6 NA NA  3  3 NA  1 NA  5  3 NA NA NA NA  1  3 NA NA  6  2  3 NA
[745] NA NA  3  2  3  2  2  5 NA  3 NA  3 NA  1 NA NA  4 NA  3 NA  3  3 NA  5
[769] NA NA  3 NA NA NA  5  1 NA NA NA  3  1 NA  3 NA  5 NA  1  2  4  1  1  1
[793]  1  5 NA NA NA  3  3  5  1  5 NA NA  3  5  3  5  1  1  1 NA NA  2  3 NA
```

```
 [817]  4 NA NA NA NA NA NA NA  5  5 NA  3  5 NA  3 NA  3 NA  1  1  5  5 NA  4
 [841] NA  3 NA NA  5 NA  2  5  1  5 NA  3  3 NA  3  1 NA  1  2  1  2  1  6 NA
 [865] NA  3  3  1  6 NA  3  1 NA  3  2 NA  4  2  3 NA NA  3  3  1  1  4  1 NA
 [889]  2  2 NA  3  1 NA  3 NA  3 NA NA  3 NA  3  5 NA  3  2  2 NA  4 NA  1  1
 [913]  1  5  3  1  1  3  1 NA  6  1  1  1  3 NA  1  3 NA NA NA NA NA NA  3  1
 [937]  3 NA  5  3 NA NA  3  3 NA  1  3 NA  3 NA  3  3 NA NA  5 NA  5 NA  5  5
 [961]  3 NA  3  4  1 NA  1  3 NA  1  2 NA  3  6  1 NA  3 NA  5  3 NA NA  3  1
 [985] NA  3  6  5  3 NA  1  3 NA  1  6  5 NA NA NA NA  3  1 NA  1  1  6  5  3
[1009]  5  1  5  3 NA  5  3 NA  2  5  3  1 NA  5 NA  6  5  5  5  2 NA  1  5  3
[1033]  3 NA  1  5  5  1  1 NA  1  5  5  2  3  1 NA  1  1 NA  4  2  3  3  1 NA
[1057]  1  5  3 NA  1  5 NA  6 NA NA  3 NA  3 NA  2  5 NA NA NA  3  1  5  5  3
[1081]  3  2 NA  6 NA  3 NA  1 NA  1  3  2  5 NA  1  1  3  3  3  1  5  1  1  6
[1105] NA  2  1  3 NA  5  1 NA  3  3  6  1  3  1  3  4  1  1  1  1 NA  3  5  3
[1129]  5  1  3 NA NA  2  6  5  1 NA NA  1  2  3 NA  4  1 NA NA  5 NA NA  5 NA
[1153]  5  3  3  1  3  1  1 NA  1  3 NA  1  2  5  4 NA NA  1 NA  3  1 NA NA  1
[1177] NA NA NA  1 NA NA  3  4  1  1 NA NA  2 NA  1 NA  1  5 NA NA NA  1  3  3
[1201]  1 NA NA  3  3 NA NA NA  2 NA NA  6  3  3  1  1  4  5  3 NA NA  4 NA  1
[1225]  3  6  5 NA  1 NA  5  5 NA  4  3  5 NA  3  1  1  5  1  5  1  1 NA  1  5
[1249] NA  1  1  3  5  1 NA NA  1 NA NA NA NA  1 NA  5 NA NA  5  3  5  5 NA  1
[1273] NA  1 NA NA  1  1 NA  5  3  1  1  5 NA  3 NA  4 NA NA  2  3  5 NA  1  1
[1297]  1 NA  6  1  1  1  1  5  1 NA  5 NA NA  1 NA NA  1  1  1 NA  2 NA  3 NA
[1321]  5  1 NA  5  1  3  6  1 NA  5  5  3  3 NA NA NA  1  5 NA  3  5 NA  5  5
[1345] NA  1  1  1 NA  5  1  2 NA  1  5 NA  1  5 NA  3  1  3  2 NA  1  1  1  3
[1369] NA  1  3 NA  3 NA  1  1  1 NA  1  1 NA  1 NA NA NA  1  1  5  1 NA  1  1
[1393] NA  3  2  1  1  5  1  5 NA NA  1  1 NA  1 NA NA  5 NA  3  5  5 NA NA  5
[1417]  3  5  5  1  5  3  3 NA NA NA  1  3 NA NA NA  1 NA NA  1  5 NA  1  3 NA
[1441]  6  1 NA  1  1  3  5 NA NA  1  1  1 NA NA  5  5  5  5  5 NA  6  2  1  5
[1465] NA NA  3 NA NA NA NA  5 NA NA  1  1  1  1  5  5  6  3  5 NA NA  5  5 NA
[1489]  1  6  1  5 NA NA  1  2  1 NA  3  1 NA NA  5 NA  1  1  1  1  1 NA  4  1
[1513] NA  1  1  1  1  1  1  1  1  1  1  5  5 NA  1  1  3  3  5  5 NA  3 NA  1
[1537] NA  3  1  3  3  5  5  3  6  1 NA NA NA NA  1  1  5  5 NA  1 NA NA  1  3
[1561]  3  5 NA  3 NA NA NA  5  5  1  1 NA  1  1  1  1  5  5  5  1 NA  1  3 NA
[1585]  1  5  3 NA NA  3  1  5  3  1 NA NA  5 NA  1 NA  1  1  1  1  3  3  1 NA
[1609]  1  3  1  5 NA  3  1 NA NA  4 NA  5 NA  1  3  6  5  1  3 NA  3  1  5  1
[1633] NA  5 NA  5 NA  3 NA  5 NA NA  6 NA NA NA NA  6  5 NA  3 NA NA  3  5 NA
[1657]  1  3  5 NA NA  6  1  5  4  1 NA NA  3 NA  5 NA  3  5  1  3  5  6 NA NA
[1681]  5 NA  3  1 NA NA NA NA NA  3  2 NA  6 NA NA NA NA  3  1  5  1 NA  3  1
[1705] NA NA NA  6  6 NA  6  5  1  5 NA  5 NA  1  1 NA NA  6
attr(,"label")
[1] "Vote 1st Round"
attr(,"format.stata")
[1] "%9.0g"
attr(,"labels")
```

```
Melenchon     Jadot    Macron  Pecresse    Le Pen   Zemmour
        1         2         3         4         5         6
```

#Operations R can be used as a calculator, performing mathematical operations:

```
  3+4
```

```
[1] 7
```

```
  17/5
```

```
[1] 3.4
```

```
  sqrt(2)
```

```
[1] 1.414214
```

```
  17/(12+4)*7
```

```
[1] 7.4375
```

```
  49/a #here we are calling up scalar *a* that we created earlier
```

```
[1] 7
```

We can use any type of mathematical operator. The list of the most common mathematical symbols in R is listed below. Do not forget the proper mathematical use of parentheses!

Operator	Meaning
+	addition
−	subtraction
*	multiplication
/	division
^	exponent
sqrt	square root
exp	exponentiation

Operator	Meaning
log	logatithm
abs	absolute value
pi	the constant π
exp(1)	the constant e

Logical statements

Logical statements in R are evaluated as to whether they are TRUE or FALSE. Here is a summary of the different logical operators in R

Operator	Meaning
<	Less Than
<=	Less Than or Equal To
>	Greater Than
>=	Greater Than or Equal To
==	Equal To
!=	Not Equal To
&	And
|	Or

For example, suppose we wanted to create a variable identifying the voters of Marine Le Pen who hold feminist views (higher than average):

```
D$fem_MLP<-D$vote1==5 & D$feminism>mean(D$feminism, na.rm=TRUE)
  #new variable, 'na.rm=TRUE' which ignores missing values
table(D$fem_MLP) #summarize new variable
```

```
FALSE  TRUE
 1263   113
```

This produces a vector of TRUE and FALSE for every observation.

Logical statements can also be used to constrain the universe of cases we use when assessing various statistics. Say, for example, that we want to see the age of respondents who are more feminist than average:

```
D$age[D$feminism>mean(D$feminism, na.rm=T)] # note option 'na.rm=TRUE' to 'na.rm=T'
```

```
  [1] 54 47 54 60 47 72 43 55 31 52 82 23 78 91 74 68 53 71 63 36 70 78 NA 29 57
 [26] 48 49 40 52 57 52 70 70 NA NA 37 74 70 48 26 76 35 34 47 23 NA 25 25 68 NA
 [51] 18 18 63 27 33 65 30 37 25 NA 38 21 46 22 23 23 58 33 19 33 23 25 NA 23 29
 [76] 20 35 27 32 18 38 56 78 37 77 62 55 52 80 25 63 24 58 37 52 60 62 51 79 68
[101] 74 NA 64 39 50 36 54 62 78 79 70 59 60 45 90 18 65 67 44 60 57 59 NA 79 48
[126] 64 50 NA NA 69 60 43 91 22 63 56 56 NA NA 62 53 79 41 86 25 NA 21 64 47 60
[151] 67 48 77 63 77 NA 66 64 55 47 24 59 72 63 62 59 84 28 27 32 23 34 29 34 21
[176] 33 47 21 29 NA 21 67 31 21 70 64 74 59 58 NA 77 71 75 56 54 66 44 85 70 80
[201] 84 59 NA NA 22 32 31 28 56 22 NA NA 68 20 25 19 20 29 22 28 35 NA 31 18 42
[226] 27 31 83 31 33 27 NA 72 NA 43 72 35 51 32 51 30 79 76 58 51 79 76 64 85 NA
[251] 78 74 69 84 58 70 83 86 89 53 NA 20 69 46 NA 55 33 NA 23 21 26 44 48 40 65
[276] 48 47 63 47 59 19 50 NA 35 41 43 59 72 18 54 83 66 72 NA 55 52 21 22 29 50
[301] 62 53 44 34 44 43 43 48 74 58 38 52 65 61 62 68 51 62 59 57 66 50 58 69 62
[326] 75 46 82 50 75 50 57 59 73 NA NA 56 NA 62 67 72 53 38 42 53 59 62 73 75 69
[351] 73 35 26 34 23 54 NA 31 28 25 19 34 56 40 73 79 81 82 79 75 81 80 NA 81 87
[376] 76 80 78 81 73 75 73 87 73 72 60 82 58 NA 70 80 78 28 75 26 24 22 70 62 25
[401] NA 25 33 33 21 23 28 35 68 60 55 76 48 NA 60 41 NA 72 50 73 74 77 69 65 69
[426] 90 64 87 68 44 52 48 30 41 40 40 52 30 45 54 64 40 40 46 NA 35 73 42 41 43
[451] 51 62 58 44 41 37 42 44 56 72 48 44 45 18 NA 60 44 50 NA 67 38 41 23 51 42
[476] 55 33 50 35 57 45 41 29 42 36 52 70 52 70 59 50 NA 42 51 58 20 46 49 26 68
[501] 49 48 59 46 48 42 50 42 52 23 40 48 55 42 29 41 47 61 47 35 49 33 40 48 44
[526] 45 34 47 51 NA 47 51 42 33 31 63 36 66 58 50 43 32 28 50 52 42 49 53 52 45
[551] 55 62 35 39 42 48 74 43 49 54 27 68 58 63 51 48 64 45 70 37 53 54 52 30 57
[576] 47 53 55 62 62 34 50 52 57 47 65 65 49 35 59 43 60 39 44 40 48 57 44 57 37
[601] 60 40 36 NA 40 24 54 40 50 53 49 NA 43 44 42 38 68 69 59 48 45 51 61 24 32
[626] 55 36 33 48 47 25 38 32 38 24 54 35 62 29 50 69 NA 54 24 36 42 47 47 23 39
[651] 21 36 44 39 51 36 53 36 23 NA 40 40 43 20 51 47 36 37 37 62 36 30 27 40 43
[676] 33 50 44 NA 30 35 37 30 36 37 38 47 42 36 35 33 67 72 48 33 25 35 56 70 63
[701] 72 60 55 NA 32 55 NA 33 27 54 52 53 76 58 NA 64 78 32 22 33 24 21 35 26 23
[726] 32 25 25 28 20 29 23 29 28 NA 21 18 29 20 19 21 19 28 NA 22 28 20 26 23 23
[751] 34 32 31 29 28 34 32 33 22 30 NA 19 26 18 25 22 28 33 35 23 23 33 18 NA 23
[776] 30 25 22 23 30 29 34 24 27 28 25 28 27 31 18 35 27 26 18 32 NA 25 29 NA 22
[801] 20 21 23 NA 20 21 21 22 19 19 23 20 21 29 23 20 19 21 24 23 18 18 24 21 18
[826] 24 NA 24 24 25 22 NA 28 26 30 21 29 28 29 28 27 23 NA 21 NA 27 30 19 22 30
[851] 23 35 33 31 28 26 30 35 NA 34 20 23 31 31 29 22 28 35 28 NA 27 35 24 22 20
[876] 24 29 21 32 32 35 25 35 26 21 NA 35 27 31 20 32 31 35 31 35 34 21 22 NA 33
[901] 30 21 30 23 33 29 NA 21 31 24 20 20 26 31 24 27 30 20 24 33 31 30 21 21 19
[926] 23 27 23 21 20 24 30 21 21 34 31 31 26 22 18 NA 21 20 24 33 35 19 NA 30 27
[951] 29 30 30 26 21 NA 31 29 NA 33 22 19 21 25 22 18 23 21 35 30 22 35 35 35 35
[976] 35 NA 35 34 22 31 35 23 35 28 29 35 19 27 23 23 35 24 27 26
```

Here notice the square brackets which contain the specific constraints.

##Recoding Often when working with data we need to recode variables, that is, to change their values for our particular purposes. There is a variety of ways to do this in R. The basic syntax for creating mathematical transformations of variables follows the form of the examples below.

Imagine, for example, that we have three variables describing a child's reading, writing and calculating ability. We are, however, interested in general academic ability of children and want to create a single summary variable. A very simple way to do this would be to create a variable which adds together the scores on the three variables we have: *ability<-reading+writing+calculating*

In our dataset, the gender variable identifies women – it is coded as 1 for women, and as 0 for men. Suppose we wanted to reverse this, and make it identify men (men==1, women==2). We could do the following

```
D$male <- -1*D$female+1 # this multiplies all values by -1 and adds 1 to all values
```

Another standard type of recoding we might want to do is the creation of a dummy variable that is coded as 1 if the observation meets a certain condition and 0 otherwise. For example, suppose instead of having categories of income, we just want to compare the highest category of income to all the others:

```
D$hi.inc.dummy <- as.numeric(D$inc==13) # codes as 1 everyone in inc cat. 13, all others a
```

Here we use a logical statement and modify the variable with the 'as.numeric' function which turns each TRUE into a 1 and each FALSE into a 0.

Now let's say that we wanted a 3 category ordinal level variable capturing education levels. Let's use the 'dplyr' package:

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
D <- D %>% #specify target and source data
  mutate(educ3 = case_when(
    is.na(educ) ~ NA,          # preserve missing data (otherwise var lengths will not matc
    educ < 2 ~ 1,              # Group values smaller than 2 as 1 on new variable
    educ %in% 3:5 ~ 2,         # Group values between 3 and 5 as 2 on new variable
    educ > 5 ~ 3               # Group values above 5 as 3 on new variable
  ))
```

Here we have created a new variable from 1 to 3, where individuals with education levels 2 or less are coded as 1, between 3 and 5 as 2 and above 5 as 3.

In certain cases we may want to change the nature of the variable from say numeric variable to a factor (a categorical variable). In this case we say:

```
D$vote1<-as.factor(D$vote1)
```

This turns the candidate one voted for into a categorical variable. From now on R will treat this variable as a categorical variable and thus when, for example, running a linear regression model with it as a predictor, R will automatically create dummy variables out of it.

Let's recode the vote variable, make it a factor, and give it labels.

```
D$vote<-factor(D$vote1, levels=c(1:6),
               labels=c("Melenchon", "Jadot", "Macron", "Peceresse", "Le Pen", "Zemmour"))
```

At this point the variable 'vote' has string values (candidate names) as opposed to numerical values (numbers). Remember, whenever we work with strings we must put their text into inverted commas "like this"!