Module 3

Data Classes

Andrew Jaffe
Instructor

Functions - Intro

- R revolves around functions: denoted by `[function name]()`
- Every function takes an input, defined by arguments, often provided by the user
- Many functions have default settings for these arguments
- `example([function name])` shows you how it is used

R Help

- If you know the name of a function, `?[function name]` or `help([function name])` will pop up the help menu

```
> ## ?str
> ## help("str")
```

Functions - Intro

For example, `length` is a function we briefly covered last module. You can try typing `?length` in the console and reading the help file.

You can also see examples of running a function using `example()` [which is another function!]

Data Classes:

One dimensional classes ('vectors'):

- Character: strings or individual characters, quoted
- Numeric: any real number(s)
- Integer: any integer(s)/whole numbers
- Factor: categorical/qualitative variables
- Logical: variables composed of TRUE or FALSE

Data Classes:

Two dimensional classes:

- `data.frame`: traditional 'Excel' spreadsheets
  - Each column can have a different class, from above
- Matrix: two-dimensional data, composed of rows and columns. Unlike data frames, the entire matrix is composed of one R class, e.g. all numeric or all characters.

Character and numeric

We have already covered `character` and `numeric`

```
> class(c("Andrew","Jaffe"))
```

```
[1] "character"
```

```
> class(c(1,4,7))
```

```
[1] "numeric"
```

Recall that `c()` and `class()` are both functions!

Integer

`Integer` is a special subset of `numeric` that contains only whole numbers

A sequence of numbers is an example of the integer class

```
> x = seq(from=1,to=5) # seq() is a function
> x
```

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

```
> class(x)
```

```
[1] "integer"
```

```
> 1:5 # shortcut; this makes a consecutive integer sequence from [num1] to [num2] by 1
```

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

Factor

`factor` are special `character` vectors where the elements have pre-defined groups or 'levels'. You can think of these as qualitative or categorical variables:

```
> x = factor(c("boy","girl","girl","boy","girl"))
> x
```

```
[1] boy  girl girl boy  girl
Levels: boy girl
```

```
> class(x)
```

```
[1] "factor"
```

Note that levels are, by default, alphabetical or alphanumerical order.

Logical

`logical` is a class that only has two possible elements: `TRUE` and `FALSE`

```
> x = c(TRUE,FALSE,TRUE,TRUE,FALSE)
> class(x)
```

```
[1] "logical"
```

`sum()` and `mean()` work on `logical` vectors - they return the total and proportion of `TRUE` elements, respectively.

Logical

Note that `logical` elements are NOT in quotes.

```
> z = c("TRUE","FALSE","TRUE","FALSE")
> class(z)
```

```
[1] "character"
```

Vector functions

Useful functions for exploring vectors (and other data types):

- `length()`
- `head()` and `tail()`
- `table()`
- `subset()` and brackets (`[ ]`)
- `unique()`
- `sum()`, `mean()`,`median()`, `min()`, `max()`

Head and Tail

- `head()` shows the first 6 (default) elements of an R object
- `tail()` shows the last 6 (default) elements of an R object
- `str()` shows the structure of an R object

```
> z = 1:100 # recall a sequence from 1 to 100
> head(z)
```

```
[1] 1 2 3 4 5 6
```

```
> tail(z)
```

```
[1]  95  96  97  98  99 100
```

```
> str(z)
```

```
 int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
```

These functions show a brief snapshot of the data which is useful for exploratory data analysis.

Table

`table()` is the basic tabulation function, which is often more useful for `character` and `factor` vectors

From the manual: "table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor level"

```
> x = c("boy","girl","girl","boy","girl")
> table(x)
```

```
x
 boy girl
   2    3
```

```
> y = c(1,2,1,2,1)
> table(x,y)
```

```
      y
x      1 2
  boy  1 1
  girl 2 1
```

Data Subsetting

Brackets are used to select/subset/extract data in R

```
> x1 = 10:20
> x1
```

```
 [1] 10 11 12 13 14 15 16 17 18 19 20
```

```
> length(x1)
```

```
[1] 11
```

Data Subsetting

```
> x1[1] # selecting first element
```

```
[1] 10
```

```
> x1[3:4] # selecting third and fourth elements
```

```
[1] 12 13
```

```
> x1[c(1,5,7)] # selecting first, fifth, and seventh elements
```

```
[1] 10 14 16
```

This is probably the most powerful and useful function in R. If you master this, you can literally do anything with R. Everything in the 'data analysis pipeline' revolves around subsetting (as you will soon see)

Matrices

```
> n = 1:9 # sequence from first number to second number incrementing by 1
> n
```

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

```
> mat = matrix(n, nrow = 3)
> mat
```

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

Matrix (and Data frame) Functions

These are in addition to the previous useful vector functions:

- `nrow()` displays the number of rows of a matrix or data frame
- `ncol()` displays the number of columns
- `dim()` displays a vector of length 2: # rows, # columns
- `colnames()` displays the column names (if any) and `rownames()` displays the row names (if any)

Data Selection

Matrices have two "slots" you can use to select data, which represent rows and columns, that are separated by a comma, so the syntax is `matrix[row,column]`.

```
> mat[1,1] # individual entry: row 1, column 1
```

```
[1] 1
```

```
> mat[1,] # first row
```

```
[1] 1 4 7
```

```
> mat[,1] # first columns
```

```
[1] 1 2 3
```

Data Selection

Note that the class of the returned object is no longer a matrix

```
> class(mat[1,])
```

```
[1] "integer"
```

```
> class(mat[,1])
```

```
[1] "integer"
```

Data Frames

The `data.frame` is the other two dimensional variable class.

Again, data frames are like matrices, but each column is a vector that can have its own class. So some columns might be `character` and others might be `numeric`, while others maybe a `factor`.

We can look at some of the example data frames that come with R.

Data Frames

```
> data(iris) ## just use some data in R already
> names(iris) ## get the column names
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[5] "Species"
```

```
> str(iris) # easy snapshot of data, like `describe` in Stata
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> head(iris, 3) # get top 3 rows
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
```

Data Selection

Data frames have special ways to select data, specifically by a `$` and the column name.

```
> head(iris$Petal.Length)
```

```
[1] 1.4 1.4 1.3 1.5 1.4 1.7
```

```
> class(iris$Petal.Width)
```

```
[1] "numeric"
```

Data Selection

You can also subset data frames like matrices, using row and column indices, but using column names is generally safer and more reproducible.

```
> head(iris[,2])
```

```
[1] 3.5 3.0 3.2 3.1 3.6 3.9
```

You can also use the bracket notation, but specify the name(s) in quotes if you want more than 1 column. This allows you to subset rows and columns at the same time

```
> iris[1:3,c("Sepal.Width","Species")]
```

```
  Sepal.Width Species
1         3.5  setosa
2         3.0  setosa
3         3.2  setosa
```

Data Frames

You can make your own data frames from "scratch" too, either from a matrix or using the `data.frame` function:

```
> x = c("Andrew","Andrew","John")
> y = 1:3
> df = data.frame(name=x, id=y)
> df
```

```
    name id
1 Andrew  1
2 Andrew  2
3   John  3
```

Data Frames

You can add variables to a `data.frame` using `$` as well:

```
> iris2 = iris # copy `iris` to a new df
> iris2$Index=1:nrow(iris2)
> head(iris2)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Index
1          5.1         3.5          1.4         0.2  setosa     1
2          4.9         3.0          1.4         0.2  setosa     2
3          4.7         3.2          1.3         0.2  setosa     3
4          4.6         3.1          1.5         0.2  setosa     4
5          5.0         3.6          1.4         0.2  setosa     5
6          5.4         3.9          1.7         0.4  setosa     6
```

```
> names(iris2)
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[5] "Species"      "Index"
```

Data Frames

- There is also the `subset` function for data frames that allows you to create a subsetted data frame:

```
setosa_small_length <- subset(iris, Sepal.Length < 6 & Species == "setosa")
head(setosa_small_length)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Data Classes: Extended

Extensions of "normal" data classes:

N-dimensional classes:

- Arrays: any extension of matrices with more than 2 dimensions, e.g. 3x3x3 cube
- Lists: more flexible container for R objects.