

Basics of R

setting up

- ▶ download R from CRAN
- ▶ work in the console (code not saved)
- ▶ open a script, type code in script, and save as a .R file

Example code is in red

R as calculator

```
> 5 + 4
```

```
[1] 9
```

```
> 8 * 2 - sqrt(9)
```

```
[1] 13
```

```
> log(4)/9^2
```

```
[1] 0.01711
```

objects

R is an object-oriented programming language. Use `<-` as assignment operator for objects.

```
> 5 + 4
[1] 9
> my.sum <- 5 + 4
> my.sum
[1] 9
> my.name <- "Brandon"
> my.name
[1] "Brandon"
\black
```

vectors

All objects consist of one or more **vectors**.

vector: a combination of elements (i.e. numbers, words), usually created using `c()`, `seq()`, or `rep()`

```
> empty.vector <- c()
> empty.vector
NULL
> one.to.five <- c(1, 2, 3, 4, 5)
> one.to.five
[1] 1 2 3 4 5
> poli.sci <- c("theory", "amer.", "comp.", "ir")
> poli.sci
[1] "theory" "amer."  "comp."  "ir"
```

All elements in a vector must be of the same data type!

data types

- ▶ numeric
- ▶ character
- ▶ logical

object classes

All objects consist of one or more vectors.

In addition to vector, objects can be of one of the following classes:

- ▶ matrix
- ▶ array
- ▶ dataframe
- ▶ list

matrix

A matrix is a two-dimensional ($r \times c$) object (think a bunch of stacked or side-by-side vectors).

```
> a.matrix <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2)
> a.matrix
      [,1] [,2]
[1,]    1    3
[2,]    2    4
> class(a.matrix)
[1] "matrix"
```

All elements in a matrix must be of the same data type.
character > numeric > logical

array

An array is a three-dimensional ($r \times c \times h$) object (think a bunch of stacked $r \times c$ matrices).

All elements in an array must be of the same data type (character > numeric > logical).

```
> an.array <- array(0, dim = c(2, 2, 3))
```

```
> an.array
```

```
, , 1
```

	[,1]	[,2]
[1,]	0	0
[2,]	0	0

```
, , 2
```

	[,1]	[,2]
[1,]	0	0
[2,]	0	0

```
, , 3
```

	[,1]	[,2]
[1,]	0	0
[2,]	0	0

dataframe

A dataframe is a two-dimensional ($r \times c \times h$) object (like a matrix).

- ▶ each column must be of the same data type, but data type may vary by column
- ▶ regression and other statistical functions usually use dataframes
- ▶ use `as.data.frame()` to convert matrices to dataframes

list

A list is a set of objects.

Each element in a list can be a(n):

- ▶ vector
- ▶ matrix
- ▶ array
- ▶ dataframe
- ▶ list

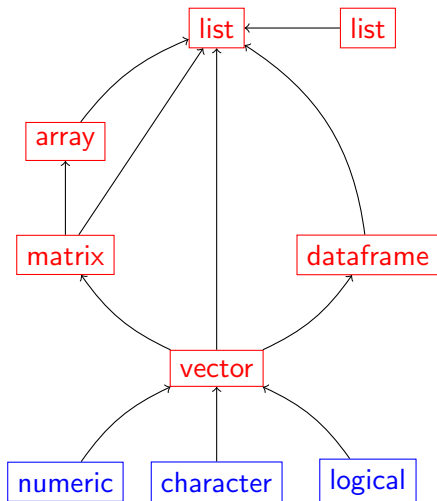
```
> a.vec <- 6:10
> a.matrix <- matrix(3, nrow = 2, ncol = 2)
> a.dataframe <- as.data.frame(a.matrix)
> a.list <- list(a.vec, a.matrix, a.dataframe)
> a.list

[[1]]
[1]  6  7  8  9 10

[[2]]
      [,1] [,2]
[1,]    3    3
[2,]    3    3

[[3]]
   V1 V2
1   3  3
2   3  3
```


brief review



object classes

data types

names

It's helpful to give names to elements or rows/columns within objects (i.e. variable names).

Use

- ▶ `names()` for vectors, dataframes and lists
- ▶ `rownames()` and `colnames()` for matrices and dataframes
- ▶ `dimnames()` for arrays

```
> leaders <- c("Obama", "Brown", "Merkel")
> names(leaders) <- c("US", "UK", "Germany")
> leaders
      US      UK  Germany
"Obama" "Brown" "Merkel"
> country.names <- names(leaders)
> country.names
[1] "US"      "UK"      "Germany"
```

indexing

Elements within objects are indexed using `[]` and `[[[]]]`.

- ▶ vectors: `[i]` for the *i*th element
- ▶ matrices and dataframes: `[i,j]` for the *i*th row, *j*th column
- ▶ arrays: `[i,j,k]` for the *i*th row, *j*th column, *k*th level
- ▶ lists: `[[i]]` for the *i*th element

the R environment

Any objects you create will be stored in the R environment.

To see all the objects in your environment:

```
> ls()  
[1] "a.dataframe"  "a.list"       "a.matrix"     "a.vec"
```

packages

To use packages, you need to install them (do this once) and load them (every time you open R).

To install a package named foo:

1. type `install.packages("foo")`
2. choose a CRAN repository

To load a package named foo:

1. type `library(foo)`

loading datasets

Suppose you want to load the foo dataset.

If the dataset is in

- ▶ an existing R package, load the package and type `data(foo)`
- ▶ .RData format, type `load(foo)`
- ▶ .txt or other text formats, type `read.table("foo.txt")`
- ▶ .csv format, type `read.csv("foo.txt")`
- ▶ .dta (Stata) format, load the foreign library and type `read.dta("foo.dta")`

To save objects into these formats, use the equivalent `write.table()`, `write.csv()`, etc. commands.

working directory

When loading or saving a dataset or object, R will look in the current working directory.

If your working directory is not where the file is at, R will not find it, so make sure you change the working directory.

- ▶ to change to the foo working directory, use `setwd("foo")`
- ▶ to see the current working directory, type `getwd()`

matrix algebra

- ▶ add/subtract matrices with $+/-$
- ▶ matrix multiply with `%*%`
- ▶ transpose with `t()`
- ▶ invert with `solve()`
- ▶ extract diagonal with `diag()`
- ▶ determinant with `det()`