

AQM

R Nuts & Bolts

Your two biggest friends:



and

> ?[insertfunctionname]

R Nuts & Bolts

Entering Inputs and assigning variables

```
> x <- 1
> print(x)
[1] 1
> x
[1] 1
> msg <- "hello"
```

```
> y <- 10:30
> y
[1] 10 11 12 13 14 15 16 17 18
19 20 21 22 23 24 25 26 27 28
29 30
```

******Notice that [1] is not actually part of the input, it is part of the printed output. It is the index of that vector. Usually, you will see printed output which is not actually part of your data.

R Nuts & Bolts

R Objects

R has five basic or “atomic” classes of objects:

- character
- numeric (real numbers)
- integer
- complex
- logical (True/False)

You can check what data type your object is by using the `class()` function. Most numbers like 1,2,... are actually represented as numeric. Put an L following a number if you must have an integer.

The most basic type of R object is a vector. Vectors can be created with the `c()` function. You can check if something is a vector by the `is.vector()` function.

There is really only one rule about vectors in R, which is that a vector can only contain objects of the same class. There is an exception, which is a list, which we will get to a bit later. A list is represented as a vector but can contain objects of different classes.

R Nuts & Bolts

R Objects

The `c()` function can be used to create vectors of objects by concatenating things together.

```
> x <- c(0.5, 0.6) ## numeric
```

```
> x <- c(TRUE, FALSE) ## logical
```

```
> x <- c(T, F) ## logical
```

```
> x <- c("a", "b", "c") ## character
```

```
> x <- 9:29 ## integer
```

```
> x <- c(1+0i, 2+4i) ## complex
```

R Nuts & Bolts

R Objects

Try:

```
> x <- c(1.7, "a")  
> y <- c(TRUE, 2)  
> z <- c("a", TRUE)
```

Check their class with class():

```
> class(x)  
> class(y)  
> class(z)
```

****Note** that x is implicitly coerced into a character, y into a numeric, z into a character.

****Note** that you can explicitly coerce a vector using `as.numeric()`, `as.character()`, `as.logical()` etc functions. If you try to do non-sensical coercions, R will give you an NA vector.

R Nuts & Bolts

Vector Operations

Try doing other stuff to these vectors:

```
> a<-c(1,2,3,4)
```

```
> b<-c(1,1,1,1)
```

Add both

Subtract both

Multiply both

Divide both

Multiply a by 2

Divide b by 2

R Nuts & Bolts

Housekeeping

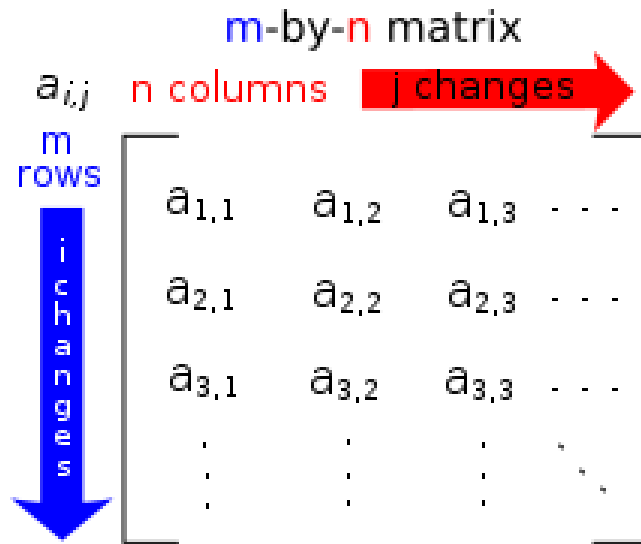
If you must remove objects in your environment use the `rm()` function.

If you must clear your console, use Ctrl L

******Note: Be careful using these, if you clear stuff without thinking, you will cry.

Matrix

In mathematics, a matrix (plural matrices) is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns—that is interpreted and manipulated in certain prescribed ways. The matrix can be described by m rows and n columns to make an $m \times n$ matrix. The individual items in a matrix are called its elements or entries.



$$\mathbf{A} = \begin{bmatrix} 0 & -1 & -2 & -3 \\ 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} -1.3 & 0.6 \\ 20.4 & 5.5 \\ 9.7 & -6.2 \end{bmatrix}.$$

R Nuts & Bolts

Matrices in R

```
> m <- matrix(1:6, nrow = 2, ncol = 3)
> m
[,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6
```

```
> matrixname[row#,col#]
Gives you access to the specified element
> m[1,1]
> 1
```

Matrices are constructed column-wise, so entries can be thought of as starting in the “upper-left” corner and running down the columns.

```
>matrixname[,col#]
Gives you the entire column specified
```

```
>matrixname[row#,,]
Gives you the entire row specified
```

Concatenate rows using `rbind()`
Concatenate columns using `cbind()`

R Nuts & Bolts

Matrices in R

A use of matrices is to find a solution to a system of linear equations. But, our primary use will be for regression. Matrices are really useful in solving a lot of problems.

Let's use the example of solving a system of linear equations to install a package, learn the meaning of a command and solve a linear problem.

R Nuts & Bolts

Matrices in R

What makes Google millions?

R Nuts & Bolts

Matrices in R

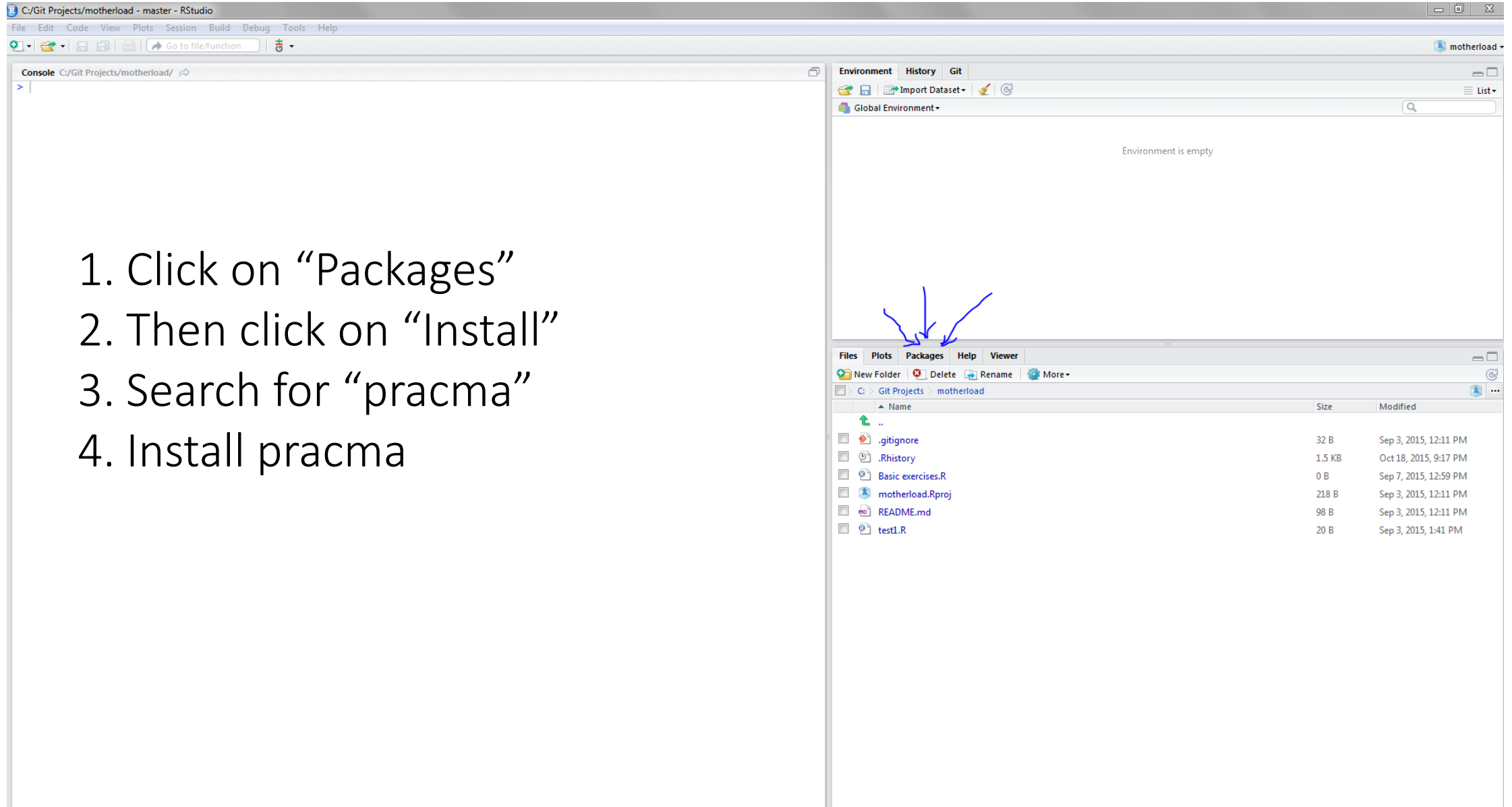
What makes Google millions?

The PageRank algorithm

Let's run through the concept using matrices in R!

R Nuts & Bolts Matrices in R

1. Click on “Packages”
2. Then click on “Install”
3. Search for “pracma”
4. Install pracma



R Nuts & Bolts

Matrices in R

How does the Google search engine decide how to rank pages in your search results? We'll start with a simple problem that illustrates the general ideas.

Suppose there are four people named Alice, Bob, Carol and Dave. They want to work out who in their group is the most popular.

They each list who is their friend follows: Alice is friends with Bob and Carol. Bob is friends with Alice, Carol and Dave. Carol is friends with Alice, Bob and Dave. Dave is friends with Alice and Carol. This can be rewritten as a table:

	Alice	Bob	Carol	Dave
Alice	0	1	1	1
Bob	1	0	1	0
Carol	1	1	0	1
Dave	0	1	1	0


**Note: Read this column wise

R Nuts & Bolts

Matrices in R

1 indicates a friendship (from the column name to the row name, but not necessarily vice versa). However, some people list more friends than others. To compensate for that, we normalize (divide each column by the number of people in it) to get the friendship matrix F :

	Alice	Bob	Carol	Dave
Alice	0	1	1	1
Bob	1	0	1	0
Carol	1	1	0	1
Dave	0	1	1	0


$$F = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{3} & 0 \\ \frac{1}{2} & \frac{1}{3} & 0 & \frac{1}{2} \\ 0 & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

R Nuts & Bolts

Matrices in R

	Alice	Bob	Carol	Dave
Alice	0	1	1	1
Bob	1	0	1	0
Carol	1	1	0	1
Dave	0	1	1	0

$$F = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{3} & 0 \\ \frac{1}{2} & \frac{1}{3} & 0 & \frac{1}{2} \\ 0 & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

We want to compute the popularity vector $\vec{r} = \begin{pmatrix} r_A \\ r_B \\ r_C \\ r_D \end{pmatrix}$ which contains the popularity of each member of the

group. The basic property that we will use is the following: *A person's popularity is the weighted sum of the popularity of people who reference that person.* For instance, we have that $r_A = \frac{1}{3}r_B + \frac{1}{3}r_C + \frac{1}{2}r_D$. In this equation, Bob's contribution to Alice's popularity is $\frac{1}{3}r_B$ because Alice makes up one third of Bob's friends. Similarly, Carol and Dave make contributions based on their friendships and their own popularity.

Written as a matrix equation, this is $F\vec{r} = \vec{r}$

Who is the most popular person given this popularity definition?