ARITHMETIC in R

Table of Contents

1 What will you learn?

Figure 2: The Arithmetic 1492-94 Fresco Palazzi Pontifici, Vatican Borgia Apartment

- Perform basic numerical operations
- Translate complex mathematical formulas
- Use logarithms and exponentials
- Brush up on mathematical E-notation
- Know R's special numbers
- Understand logical values and operators

Image: The Arithmetic 1492-94 Fresco Palazzi Pontifici, Vatican Borgia Apartment (Wikipedia).

Sources: Some material for this lesson comes from Davies (2016) and Matloff (2020). These and other sources have been important to me in preparing this course and getting into R int he first place. Check them out for a more systematic treatment of R. There is also a more philosophical, personal view on my use of sources in the Wiki for the 2020 version of this course.

What is this? When we say "Arithmetic", we don't mean that we "study" numbers but that we use them to perform computations. After this section, you'll be able to perform any arithmetic operation using R.

We will look at *operators* first, then at simple but important functions that occur again and again, especially in *statistics*.

How can you learn better? This presentation consists mostly of text and code chunks. Because this is dry stuff, I urge you (both if you hear this in class, and if you work through this on your own) to open an R session on the side and type along - this will build muscle memory and keep you entertained, too! Another trick, which you will find in Matloff's excellent beginner's tutorial, is to make your own little exercises by varying the instructions. A third way is to go through the lecture and create your own Emacs Org-mode notebook with R code blocks.

2 Please Excuse My Dear Aunt Sally

Operator order:

1. **P** arentheses: `()`
2. **E** xponentiation: `^` or `**`
3. **M** ultiplication: `*`
4. **D** ivision: `/`
5. **A** ddition: `+`

6. **S** ubtraction: -

Tip: You can check identity in R with the `identical` *function.*

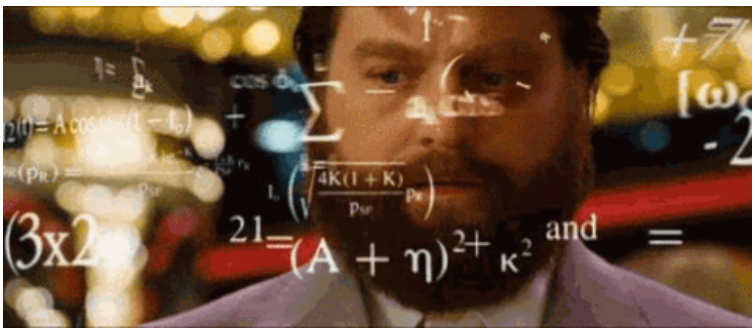
In R, standard mathematical rules apply. The order of operators is as usual - left to right, parentheses, exponents, multiplication, division, addition, subtraction (PEMDAS = Please Excuse My Dear Aunt Sally) mnemonic).

The operators `^` and `**` for exponentiation are identical, though `^` is more common. You can check that in the R console with the `identical` function - the result should be `TRUE` (this is a truth or Boolean value - more on this below) - see figure:

```
> 2**3
[1] 8
> 2^3 # same as 2 * 2 * 2
[1] 8
> 2**3 # same as 2 * 2 * 2
[1] 8
> identical(2^3, 2**3)  # checking that these are indeed the same
[1] TRUE
```

3 Practice: Formula Translator

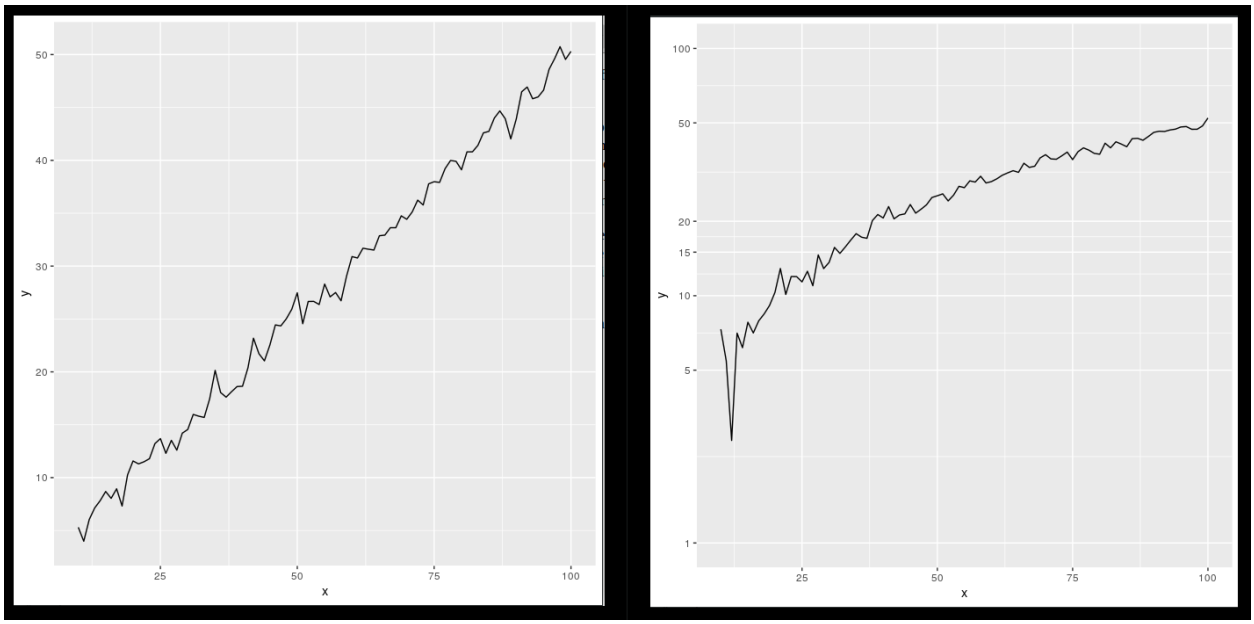
4 Mathematical functions



- `?sqrt` ($\sqrt{}$)
- `?log10` ($\log_{10}$)
- `?exp` (e)
- `?pi` (π)

Do you know how to compute these by hand?[1]

5 Logarithmic transformation



Some examples:

```
log10(1 * 10^7) = 7

log10(100) = 2 , log10(1000) = 3 , log10(1e3) = log10(1 * 10^3) = 3

log(1) = 0, log10(1) = 0

log(x=100,b=100) = 1 , log(4.583,4.583) = 1

log(x=100,b=10) = 2 , log(b=10,x=100) = 2
```

More examples: The Economist/Off The Charts 04/20/2021[2]

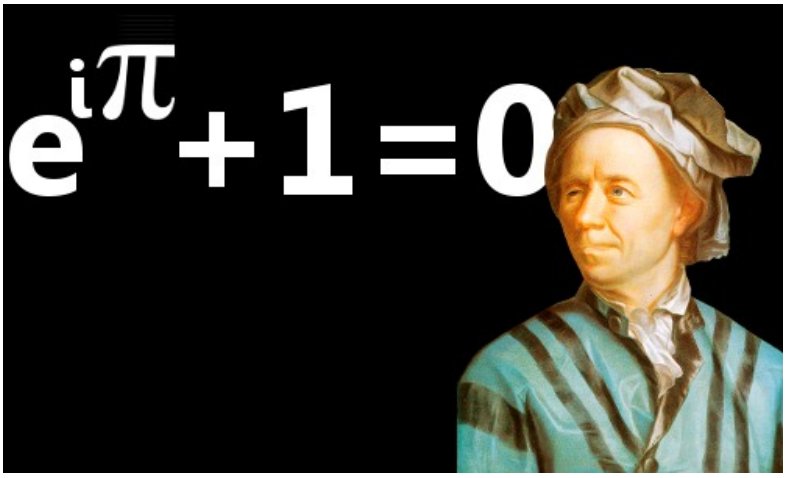
It is often necessary to transform numerical data, e.g. transforming data using the logarithm leading e.g. from the left to the right graph in the figure. As you can see, this transformation leads to a compression of the `y`-values, so that more of these values can be shown.

The *logarithm* of a number `x` is always computed using a *base* `b`. In the diagram, `b=10`, the numbers on the `x` axis were transformed using the `log()` function, the logarithm with base `10`. The logarithm of `x=100` to the base `10` is `2`, because $10^2 = 100$. In R, `log(x=100,b=10) = 2` (try this yourself!).

6 Logarithm rules

- Argument `x` and base `b` must be positive
- For all x: `log(x, b=x) = 1` since only $x^1 = x$
- For all b: `log(x=1, b) = 0` since $b^0 = 1$

7 Exponential function

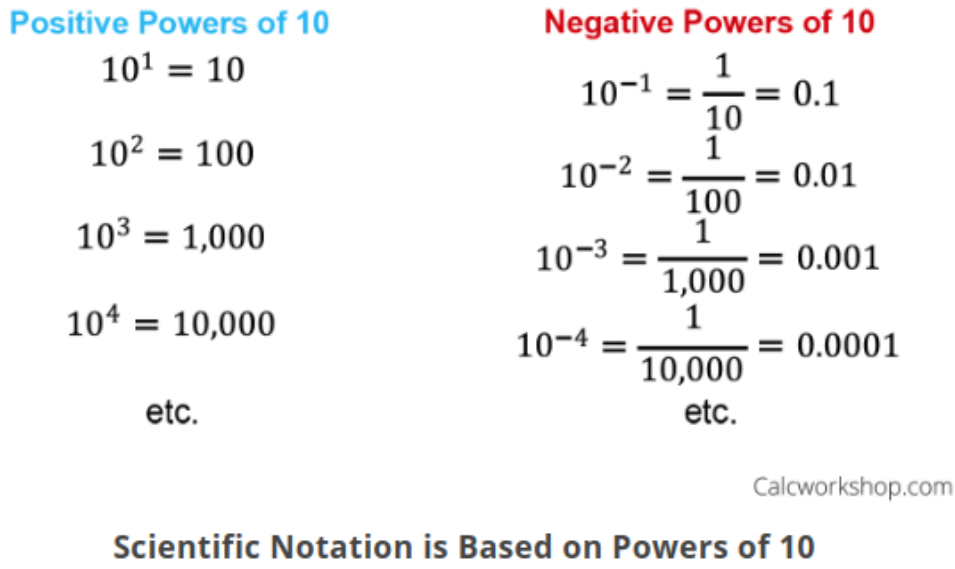


- In R, `log(x)` implies `b = e \approx 2.7182`
- In mathematics, the *Euler constant* `e` is as magical as the other mysterious constants π, 0, 1 and i (the imaginary unit). There are different ways to arrive at its value of approximately 2.718282.
- The Wikipedia entry on `e` contains some fun stuff for nerds (here). Apparently, *Steve Wozniak* computed `e` to 116,000 digits on an "ancient" Apple II computer in *1981*!
- For now, we only care about the fact that *e* is the base of the natural logarithm, denoted as `ln` or $\log_e(x)$.

8 Practice: logarithms and constants

$e^{ln^x} = ln(e^x)$

9 E-notation



You already know that the number of digits that is displayed by R can be changed using the `options()` utility function. The default number of digits displayed is 7.

In order to display values with many more digits than that - either very large, or very small numbers, we use the scientific or e-notation. In this notation, any number is expressed as a multiple of 10.

10 E-xamples

- 10,0000 = `10 * 10 * 10 * 10 * 10` = $1 * 10^5$ = `1e+05`
- `7.45678389e12` = $7.45678389 * 10^{12} = 745.678389 * 10^{10}$
- `exp(1)` = e = `271828182845e-11` = $271828182845 \times 10^{-11}$

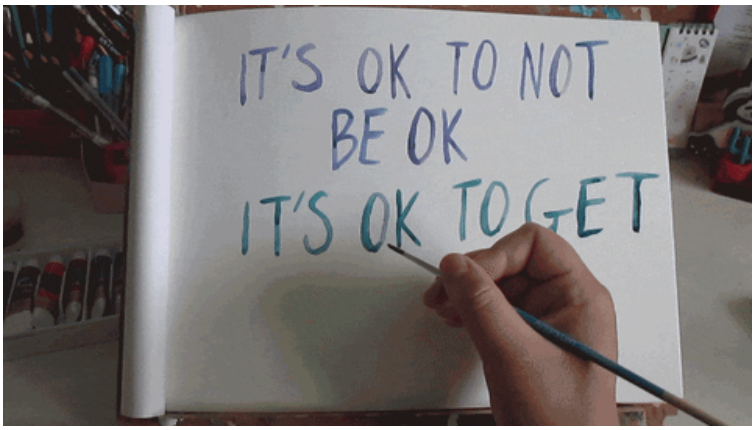
11 Practice: e-notation

To get from the e-notation with exponent y or -y to the complete number of digits, simply move the decimal point by y places to the right or to the left, resp.

No information is lost even if R hides digits; e-notation is purely to improve readability. Extra bits are stored by R

`Inf`, `-Inf` and `NaN` are special numbers.

12 Math help in R



- `?Arithmetic`
- `?Math`
- `?Comparison` etc.

13 Special numbers

- `Inf` for positive infinity (∞)
- `-Inf` for negative infinity ($-\infty$)
- `NaN` for "not-a-number" (not displayable)
- `NA` for "not available" (missing value)

1. `NA` values are especially important when we clean data and must remove missing values. There are Boolean (logical) functions to test for special values.
2. Missing values can be created easily by doing "forbidden" stuff. An example is trying to compute the square root of a negative number, e.g. `(-2)^(1/2)`. The result is a complex number (in this case the solution to the quadratic equation $x^2+1=0$, called the imaginary number i). You can also use the function `is.na` to test for missing values: compute `is.nan(sqrt(-1))` for example.

14 Practice: special numbers

15 Special functions

`is.finite(Inf)`	`is.infinite(Inf)`
`is.finite(NA)`	`is.na(NA)`
`is.nan(NaN)`	`is.nan(NA)`

```
> is.finite(NA) # Missing values don't count as 'finite'
[1] FALSE
> is.infinite(Inf)  # Checking infinity is dodgy but works
[1] TRUE
> is.finite(Inf)
[1] FALSE
> is.nan(NaN)  # Checking "Not a Number"
[1] TRUE
> is.na(NA) # Checking missing values "Not Available"
[1] TRUE
> is.nan(NA) # Missing values are not non-numbers!
[1] FALSE
```

16 Practice: special functions

17 Logical values and operators

Figure 19: George Boole

`TRUE` and `FALSE` are reserved in R for logical values, and the variables `T` and `F` are already predefined. This can cause problems, because these variable names are not reserved, i.e. you can redefine them. So better stay away from saving time by using the short versions of these values.

18 Practice: logical values

19 Logical operators

There are three logical operators in R:

`!` for "*not*": `1 != 1`

`&` for "*and*": `(1==1) & (1==2)`

`|` for "*or*": `(1==2) ~| (1!=1)`

```
> 1 == 1
[1] TRUE
> 1 == 2
[1] FALSE
> 1 != 1
[1] FALSE
> 1 != 2
[1] TRUE
> 1 | 2
[1] TRUE
> 1 | 1
[1] TRUE
> (1==2) | (1!=1)
[1] FALSE
```

In the last command, we generated a `FALSE` value by comparing two `FALSE` values, which is the only way to make an `|` statement `FALSE`.

20 Practice: logical operators

21 Concept summary

- In R mathematical expressions are evaluated according to the *PEMDAS* rule.
- The natural logarithm ln(x) is the inverse of the exponential function e^x.
- In the scientific or e-notation, numbers are expressed as positive or negative multiples of 10.
- Each positive or negative multiple shifts the digital point to the right or left, respectively.
- Infinity `Inf`, not-a-number `NaN`, and not available numbers `NA` are *special values* in R.

22 Code summary

CODE	DESCRIPTION
`log(x=,b=)`	logarithm of `x`, base `b`
`exp(x)`	e^x, exp[onential] of x
`is.finite(x)`	tests for finiteness of `x`
`is.infinite(x)`	tests for infiniteness of `x`
`is.nan(x)`	checks if `x` is not-a-number
`is.na(x)`	checks if `x` is not available
`all.equal(x,y)`	tests near equality
`identical(x,y)`	tests exact equality
`1e2`, `1e-2`	$10^2 = 100$, $10^{-2} = \frac{1}{100}$

23 References

- Richard Cotton (2013). Learning R. O'Reilly Media.
- Tilman M. Davies (2016). The Book of R. (No Starch Press).
- Rafael A. Irizarry (2020). Introduction to Data Science (also: CRC Press, 2019).
- Norman Matloff (2020). fasteR: Fast Lane to Learning R!.

Footnotes:

[1] I've recently been reminded through this article how important it may be to be able to do computations without the help of machines. Here are 4 ways to compute `sqrt` in C (though not very fast). In general: 1) using logarithms and exponentials (sqrt(x)=$e^{0.5 \times \ln(x)}$), 2) using successive approximate numerical methods like Newton's iteration, 3) using modified long division (prime factorization), 4) looking it up in a table (source: quora.com)

[2] The log transformation uses "Covid-19, confirmed deaths" data. In hindsight, I don't find the explanation in the text particularly satisfying: "Plotted on a linear scale countries' caseload seemed to rocket up out of nowhere. But plotted on a log scale it became easy to see which countries were on the path to an outbreak, and how far ahead or behind each one was, relative to the others." I just don't see this information more clearly in the log plot, do you?

Author: Introduction to data science (DSC 105) Fall 2022
Created: 2022-09-22 Thu 15:15