Math for Data Analytics I

QBS 103: Foundations of Data Science

August 20, 2024

Lesson Objectives

At the end of this lecture you should be able to:

1. Read and interpret mathematical notation
2. Write and plot mathematical functions in R
3. Write logarithmic and trigonometric functions in R

Resources

LaTeX Cheat Sheet for Set Notation: https://www.overleaf.com/learn/latex/List_of_Greek_letters_and_math_symbols

Good Explanation of Mathematical Notation: https://www.youtube.com/watch?v=kaJuXx6uYR0

LaTeX in R Plots Using *latex2exp*: https://cran.r-project.org/web/packages/latex2exp/vignettes/using-latex2exp.html

Mathematical Notation

Mathematical equations and proofs often rely on set notation to efficiently state information. This is a quick overview (not a comprehensive list) of some mathematical notation you might encounter as you move forward in statistics.

For example, we can define

$$A = \{x, y, z\}$$

which means that A is a set that contains elements x, y, and z. We can refer to elements of the set using $\in$, such that we can say

$$x \in A$$

Similarly, we can use the element symbol dashed through like this $\notin$ to indicate something isn't an element of that set, such that

$$s \notin A$$

One special set that you might see is $\mathbb{R}$ which indicates the set of all real numbers.

We can also define a set as

$$\{x | x > 0\}$$

rather than listing out individual values which would indicate the set of x in which x is greater than 0.

We also use $\forall$ to indicate "for all" such that we can say

$$\forall\ x \in A, x > 0$$

Reading across this just says, "for all x elements of A, x is greater than zero" or, more simply, all the elements of A are greater than zero.

We can also use the $\exists$ symbol to indicate "exists" such that we can say $\forall \in A \; \exists$ such that $x > 0$ Reading across this means that there exists an element in A that is greater than zero. Similarly to $\notin$ we also have $\nexists$ which indicates "there does not exist".

Simple Mathematical Functions in R

In this course already, we have been using functions, simulating data, making plots, and even using some linear algebra - all of which are foundational to calculus!

For instance, supposed we are interested in the following mathematical function:

$$f(x) = mx + b$$

To create this function in R, we represent this function as:

```
myFunction <- function(m,x,b){
  m*x+b
}
```

And then if we were to solve this function for $m = 1$, $x = 2$, and $b = 3$ we could:

$$f(2) = 1(2) + 3 = 5$$

or, use our function:

```
myFunction(m = 1,x = 2,b = 3)
```

```
## [1] 5
```

It is then also super easy to evaluate this function over a range of x values. For instance, we may want to know the solution of $f(x)$ over the domain $\{x \in \mathbb{N} | 0 \leq x \leq 1000\}$. You could solve this by hand, or you could generate values of x over your domain and solve like so:

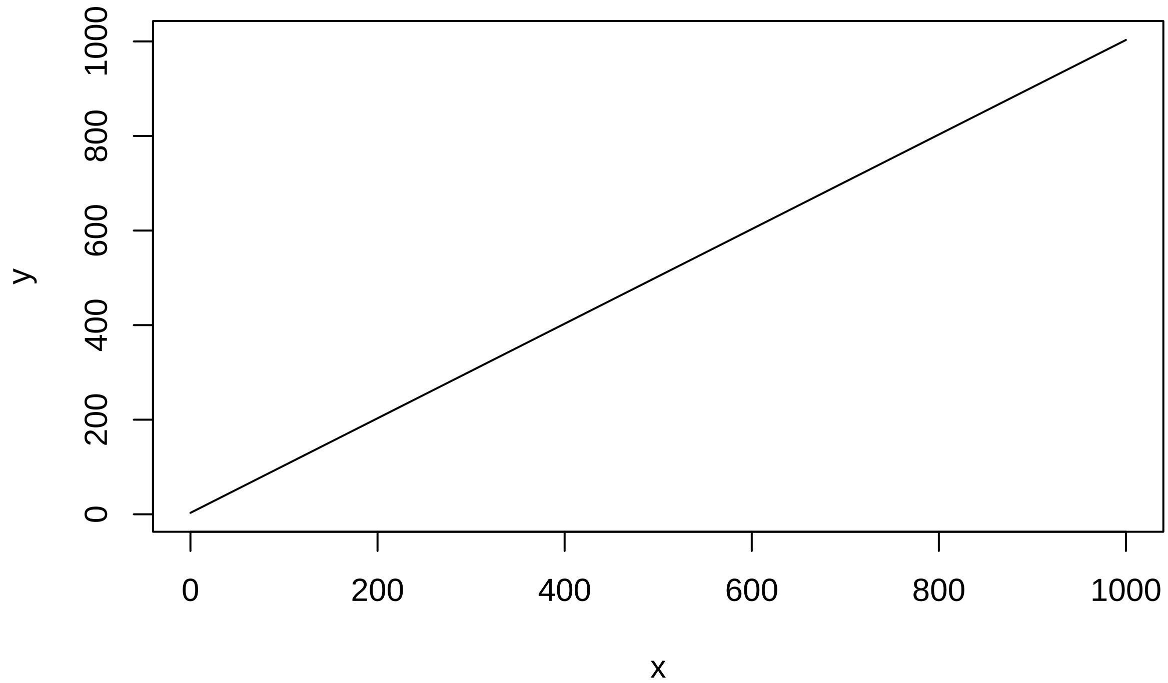
```
# Set the domain for our function
x <- seq(from = 0,to = 1000,by = 1)
# Calculate all values of f(x) within the domain defined above
y <- myFunction(m = 1,x = x,b = 3)

# Look at some values of y
head(cbind(x,y))
```

```
##      x y
## [1,] 0 3
## [2,] 1 4
## [3,] 2 5
## [4,] 3 6
## [5,] 4 7
## [6,] 5 8
```

And you can see that we then have each solution for every value of x! Now we can easily plot these values.

```r
# Using base R
plot(x,y,type = 'l')
```



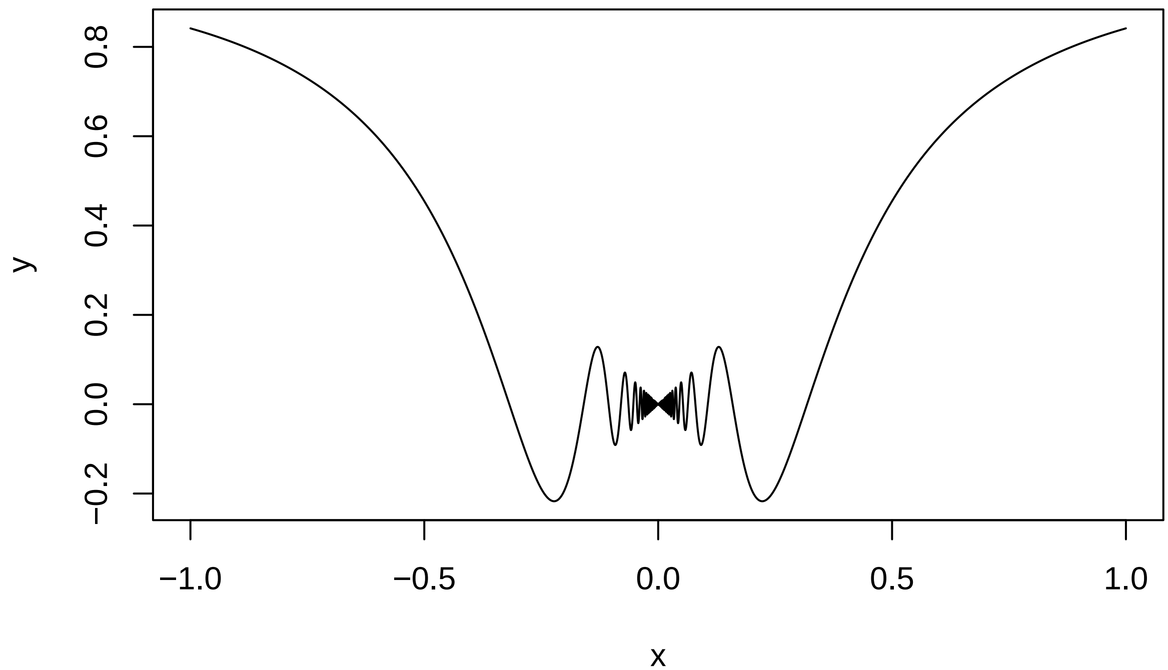
Representing functions this way becomes increasingly useful with increasingly complex functions. For instance, suppose we have the following function:

$$f(x) = x\sin\left(\frac{1}{x}\right)$$

and we want to solve it over the domain $\{x \in \mathbb{R}| -1 \leq x \leq 1\}$. You could solve this by hand over many possibilities of x, or:

```r
# Define function
myFunction <- function(x) {
  return(x*sin(1/x))
}
# Set the domain
x <- seq(from = -1,to = 1,length.out = 10000)
# Calculate f(x) over the domain of x
y <- myFunction(x)

# Plot the function
plot(x,y,type = "l")
```



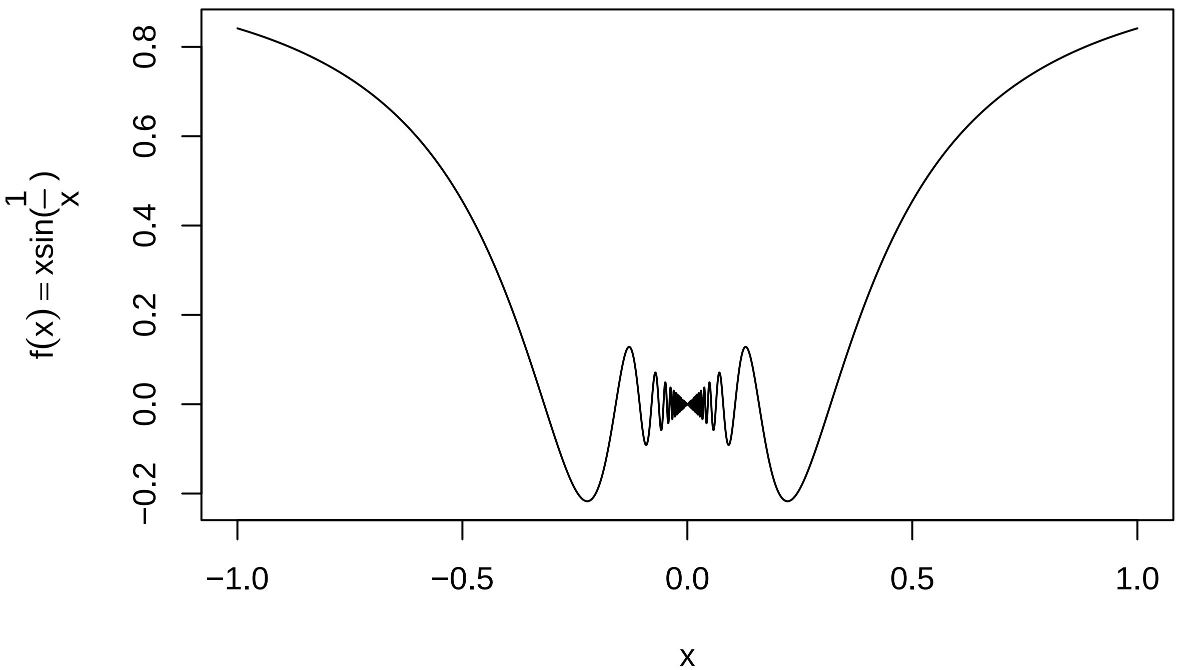
Sometimes, you might want to use LaTeX within your plots. To do so, we'll use the *latex2exp* library. Additional information and examples are linked at the start of the lecture under resources.

```r
# Load package
#install.packages('latex2exp')
library(latex2exp)
```

```
## Warning: package 'latex2exp' was built under R version 4.1.2
```

```r
# Increase margin size for function
# Syntax here is: par(mar = c(bottom, left, top, right)
par(mar = c(5, 5, 4, 2) + 0.1)

# Define y axis label
axis <- TeX(r'($f(x)=x$sin($\frac{1}{x}$)$)')
# Generate plot
plot(x,y,type = "l",ylab = axis)
```



Exponents amd Logarithms in R

In R, *log(x)* calculates the natural log of x (i.e. base-e). If you type in ?log you'll see that you have the ability to customize the base within the log function as well.

```
# Define a range of x
x <- 1:10

# Calculate log x
log(x)
```

```
##  [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 1.9459101
##  [8] 2.0794415 2.1972246 2.3025851
```

Given that log_2 and $log_1 0$ are both commonly used logarithms, there are separate functions for these as well.

```
# Calculate log2
log(x,base=2)
```

```
##  [1] 0.000000 1.000000 1.584963 2.000000 2.321928 2.584963 2.807355 3.000000
##  [9] 3.169925 3.321928
```

```r
log2(x) # same as above
```

```
##  [1] 0.000000 1.000000 1.584963 2.000000 2.321928 2.584963 2.807355 3.000000
##  [9] 3.169925 3.321928
```

```r
# Calculate log10
log(x,base=10)
```

```
##  [1] 0.0000000 0.3010300 0.4771213 0.6020600 0.6989700 0.7781513 0.8450980
##  [8] 0.9030900 0.9542425 1.0000000
```

```r
log10(x) # same as above
```

```
##  [1] 0.0000000 0.3010300 0.4771213 0.6020600 0.6989700 0.7781513 0.8450980
##  [8] 0.9030900 0.9542425 1.0000000
```

We've already gotten very used to using the ˆ symbol for creating exponents. When we want to raise something to the power of e we use the *exp()* function in R.

```r
# Calculate e using exp() function
exp(1)
```

```
## [1] 2.718282
```

```r
# Confirm the same
log(x)
```

```
##  [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 1.9459101
##  [8] 2.0794415 2.1972246 2.3025851
```

```r
log(x,base = exp(1))
```

```
##  [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 1.9459101
##  [8] 2.0794415 2.1972246 2.3025851
```

You'll recall that there are special rules for using math with logarithms. You'll want to remember these before you dive back into statistics and calculus. We can also confirm these in R pretty easily. They include:

$$log_{10}(ab) = log_{10}(a) + log_{10}(b)$$

```r
# Define our variables a and b
a <- 2
b <- 4

# Confirm law
log10(a * b)
```

```
## [1] 0.90309
```

```r
log10(a) + log10(b)
```

```
## [1] 0.90309
```

$$ln(\frac{a}{b}) = ln(a) - ln(b)$$

```r
log(a/b)
```

```
## [1] -0.6931472
```

```r
log(a) - log(b)
```

```
## [1] -0.6931472
```

$$ln(1/x) = -ln(x)$$

```r
# Define x and y
x <- 5
y <- 8

log(1/x)
```

```
## [1] -1.609438
```

```r
-log(x)
```

```
## [1] -1.609438
```

$$log_2(x^y) = ylog_2(x)$$

```r
log2(x^y)
```

```
## [1] 18.57542
```

```r
y*log2(x)
```

```
## [1] 18.57542
```

$$ln(e) = 1$$

```r
log(exp(1))
```

```
## [1] 1
```

$$ln(1) = 0$$

```r
log(1)
```

```
## [1] 0
```

Similarly, there are a variety of exponent laws worth reviewing:

$$e^a e^b = e^{a+b}$$

```r
exp(a)*exp(b)
```

```
## [1] 403.4288
```

```r
exp(a+b)
```

```
## [1] 403.4288
```

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

```r
exp(log(x*y))
```

```
## [1] 40
```

```r
x*y
```

```
## [1] 40
```

$$2^{log_2(xy)} = xy$$

```r
2^(log2(x*y))
```

```
## [1] 40
```

```r
x*y
```

```
## [1] 40
```

$$\frac{e^a}{e^b} = e^{a-b}$$

```r
exp(a)/exp(b)
```

```
## [1] 0.1353353
```

```r
exp(a-b)
```

```
## [1] 0.1353353
```

$$(e^a)^b = e^{ab}$$

```r
exp(a)^b
```

```
## [1] 2980.958
```

```r
exp(a*b)
```

```
## [1] 2980.958
```

$$(xy)^a = x^a y^a$$

```r
(x*y)^a
```

```
## [1] 1600
```

```r
x^a * y^a
```

```
## [1] 1600
```

$$x^{-a} = \frac{1}{x^a}$$

```r
x^(-a)
```

```
## [1] 0.04
```

```r
1/(x^a)
```

```
## [1] 0.04
```

$$x^{\frac{a}{b}} = \sqrt[b]{x^a}$$

```r
x^(a/b)
```

```
## [1] 2.236068
```

```r
(x^a)^(1/b)
```

```
## [1] 2.236068
```

Trigonometry

You'll often need to use trigonometry in Statistics and Calculus, so naturally our trig functions are built into R as well. Above, we used the *sin()* function in a plot but there are also functions for *cos()* and *tan()* as well as arcsin, arcos, and arctan, all shown below.

```r
# Define values for x
x <- c(-1,0,1)

# Functions named exactly
sin(x)
```

```
## [1] -0.841471  0.000000  0.841471
```

```r
cos(x)
```

```
## [1] 0.5403023 1.0000000 0.5403023
```

```r
tan(x)
```

```
## [1] -1.557408  0.000000  1.557408
```

```r
# Functions named slightly differently
asin(x)
```

```
## [1] -1.570796  0.000000  1.570796
```

```r
acos(x)
```

```
## [1] 3.141593 1.570796 0.000000
```

```r
atan(x)
```

```
## [1] -0.7853982  0.0000000  0.7853982
```

You'll reall that there are also a variety of trig identities you can use to make solving functions with trigonometry easier! A complete cheat sheet can be found here: https://sciencenotes.org/trig-identities-study-sheet/ but below are some useful ones you'll frequently use:

$$\tan\theta = \frac{\sin\theta}{\cos\theta}$$

```r
# Define theta
theta <- pi/7

tan(theta)
```

```
## [1] 0.4815746
```

```r
sin(theta)/cos(theta)
```

```
## [1] 0.4815746
```

$$\sin^2\theta + \cos^2\theta = 1$$

```r
sin(theta)^2 + cos(theta)^2
```

```
## [1] 1
```

$$\sin(2\theta) = 2\sin\theta\cos\theta$$

```r
sin(2*theta)
```

```
## [1] 0.7818315
```

```r
2*sin(theta)*cos(theta)
```

```
## [1] 0.7818315
```

$$\cos(2\theta) = \cos^2\theta - sin^2\theta = 2\cos^2\theta - 1$$

```r
cos(2*theta)
```

```
## [1] 0.6234898
```

```r
cos(theta)^2 - sin(theta)^2
```

```
## [1] 0.6234898
```

```r
2*cos(theta)^2 - 1
```

```
## [1] 0.6234898
```

$$sin(-\theta) = -\sin(\theta)$$

```r
sin(-theta)
```

```
## [1] -0.4338837
```

```r
-sin(theta)
```

```
## [1] -0.4338837
```

You may be wondering why we are reviewing so much of this in the context of R, and one very great reason is because R will be a useful tool in checking your work when you are required to use these laws to manually simplify functions. Consider for instance:

$$f(\theta) = \frac{1+\sin\theta}{\cos\theta} + \frac{\cos\gamma}{1-\sin\gamma}$$

And you simplify to:

$$f(\theta) = \frac{2\sin\theta - 2\sin\gamma}{\sin(\theta-\gamma) + \cos\theta - cos\gamma}$$

We can test this in R:

```r
theta <- 2
gamma <- 5

# Calculate original function
(1+sin(theta))/cos(theta)+cos(gamma)/(1-sin(gamma))
```

```
## [1] -4.443233
```

```r
# Calculate simplified function
(2*sin(theta)-2*sin(gamma))/(sin(theta-gamma)+cos(theta)-cos(gamma))
```

```
## [1] -4.443233
```