Lab_X

David Björelind, davbj395

10/22/2020

1. Be careful when comparing

```
## [1] "Subtraction is wrong"
```

```
## [1] "Subtraction is correct"
```

The first expression tells us "Subtraction is wrong" and the second expression "Subtraction is correct". The first get it incorrect, because 1/3-1/4 cannot be represented in an exact way in binary. 1-1/2 can be represented correctly, which is why it gets the calculations correct.

Improvements: use the comparing statement 'all.equal()' instead of using '==' will help this.

```
options(digits=20)
1/3-1/4
```

```
## [1] 0.083333333333333331483
```

```
1/12
```

```
## [1] 0.083333333333333328707
```

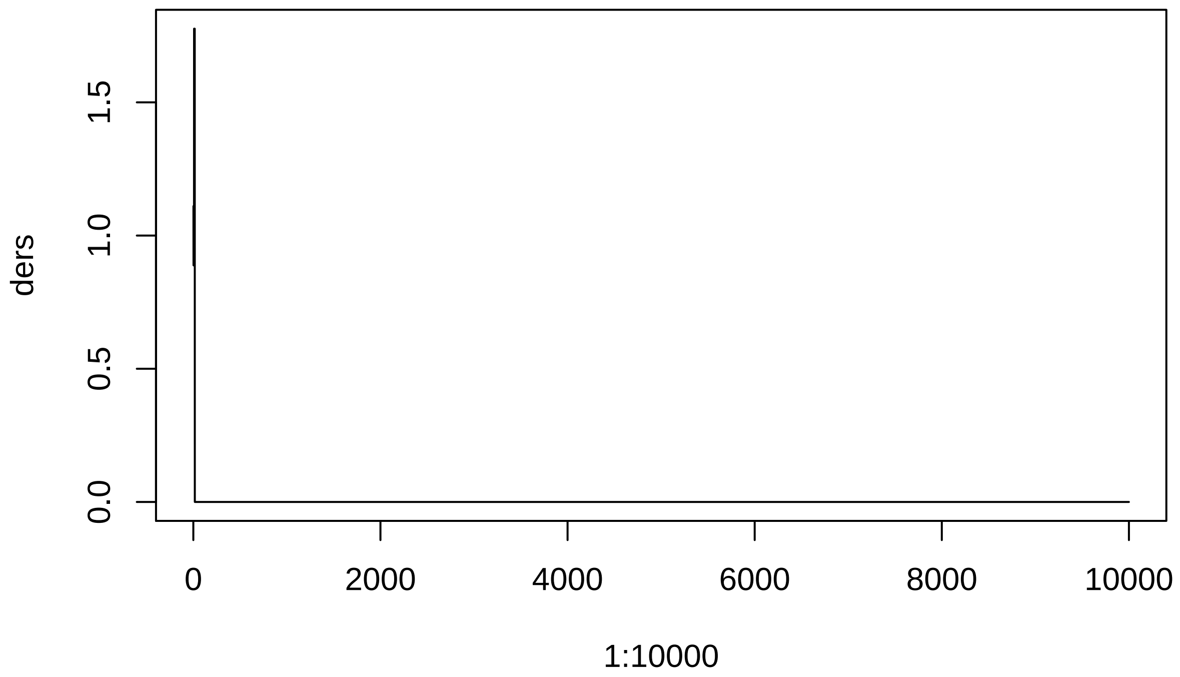
```
x1<- 1/3
x2<- 1/4

# First expression again
if ( all.equal(x1-x2,1/12) ) {
print ("Subtraction is correct")
} else {
print ("Subtraction is wrong")
}
```

```
## [1] "Subtraction is correct"
```

Now it gets it correct!

2. Derivative



The results are very surprising! The expected answer would be **1** for each value from 1-1000, f'(x) = 1. This is not observed however. The values quickly becomes 0, and this can be explained by underflow. **f(x+e)-f(x)** will produce a very small number. When **x** gets larger, the misrepresentation of **f(x+e)-f(x)** will grow and at some point it will be **0**.

```
## [1] 1.7763568394002504647e-15
```

```
## [1] 1.7763568394002504647e-15
```

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

3. Variance

4. Binomial Coefficient

Another chunk

Include all code for this report

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, message=FALSE)
# Include packages here
```

```r
x1<- 1/3
x2<- 1/4

# First expression
if ( x1-x2 == 1/12 ) {
print ("Subtraction is correct")
} else {
print ("Subtraction is wrong")
}

# Second expression
x1 <- 1
x2 <- 1/2
if ( x1-x2 == 1/2 ) {
print ("Subtraction is correct")
} else {
print ("Subtraction is wrong")
}
options(digits=20)
1/3-1/4
1/12

x1<- 1/3
x2<- 1/4

# First expression again
if ( all.equal(x1-x2,1/12) ) {
print ("Subtraction is correct")
} else {
print ("Subtraction is wrong")
}
derivative = function(x, func, epsilon){
  return((func(x+epsilon)-func(x))/epsilon)
}
f = function(x){
  return(x)
}

epsilon = 10^-15

# Evaluating derivative
ders = c()
for (i in 1:10000){
  temp = derivative(i, f, epsilon)
  ders = c(ders, temp)
}

plot(x=1:10000, y=ders, type = ’l’)
(10+epsilon)-10
(15+epsilon)-15
(20+epsilon)-20
```