Session 3: More About Functions

QM 1

2022-11-01

if else vs. ifelse

```
a <- 4
b <- 8
if (TRUE) { print(a) }
```

```
## [1] 4
```

```
if (FALSE) { print(a) }
if (a > 3) { print(a) }
```

```
## [1] 4
```

```
if (a > 5) {
  print(a)
} else {
  print(b)
}
```

```
## [1] 8
```

```
if (c(a,b) > 5) { cat("Yeah.\n") } else { cat("No!\n") }
```

```
## Warning in if (c(a, b) > 5) {: the condition has length > 1 and only the first
## element will be used
```

```
## No!
```

```
if (c(b,a) > 5) { cat("Yeah.\n") } else { cat("No!\n") }
```

```
## Warning in if (c(b, a) > 5) {: the condition has length > 1 and only the first
## element will be used
```

```
## Yeah.
```

```
ifelse(c(b,a) > 5, "Yeah", "No")
```

```
## [1] "Yeah" "No"
```

```
ifelse((1:10)%%2 == 0, "divisible by 2", "not divisible by 2")
```

```
##  [1] "not divisible by 2" "divisible by 2"     "not divisible by 2"
##  [4] "divisible by 2"     "not divisible by 2" "divisible by 2"
##  [7] "not divisible by 2" "divisible by 2"     "not divisible by 2"
## [10] "divisible by 2"
```

```
# %% - modulo operator - rest after division
```

max vs. pmax

```r
data <- 1:10
max(5, data)
```

```
## [1] 10
```

```r
max(15, data)
```

```
## [1] 15
```

```r
pmax(5, data)
```

```
##  [1]  5  5  5  5  5  6  7  8  9 10
```

THE TAX EXAMPLE, slides 40, 41:

```r
T1 <- function(y) { y^2 / 1e6 - 100 }
T2 <- function(y) { 9900 + (y - 1e5)/4 }

y <- c(9000, 50000, 80000, 120000, 160000)
T2(y[4:5])
```

```
## [1] 14900 24900
```

```r
T1(y[1:3])
```

```
## [1]  -19 2400 6300
```

```r
# wrong implementation using max instead of pmax ...
T_wrong1 <- function(y) {
  ifelse(y < 1e5, max(0, T1(y)), T2(y))
}
# ...returns the maximum of 0, T1(y[1]), T1(y[2]), T1(y[3]), T1(y[4]), whenever y<1e5
# and T2(y) otherwise
T_wrong1(y)
```

```
## [1] 25500 25500 25500 14900 24900
```

```r
# wrong implementation using if (){} else {} instead of ifelse () ...
 T_wrong2 <- function(y) {
  if (y < 1e5) {
   pmax(0, T1(y))
  } else {
    T2(y)
  }
}
# ...returns pmax(0, T1(y)) for all input values
# since the condition considers only the first input argument
T_wrong2(y)
```

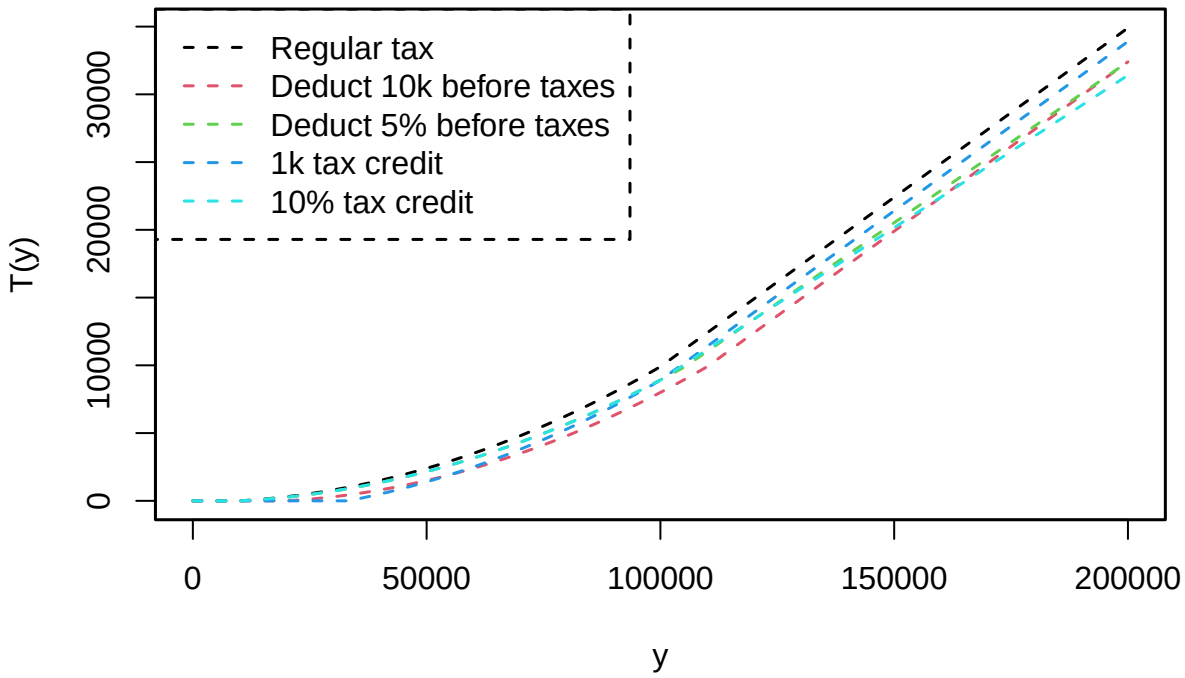
```
## Warning in if (y < 1e+05) {: the condition has length > 1 and only the first
## element will be used
```

```
## [1]     0  2400  6300 14300 25500
```

```r
# correctly implemented function
T <- function(y) {
  ifelse(y < 1e5, pmax(0, T1(y)),  T2(y))
}
T(y)
```

```
## [1]     0  2400  6300 14900 24900
```

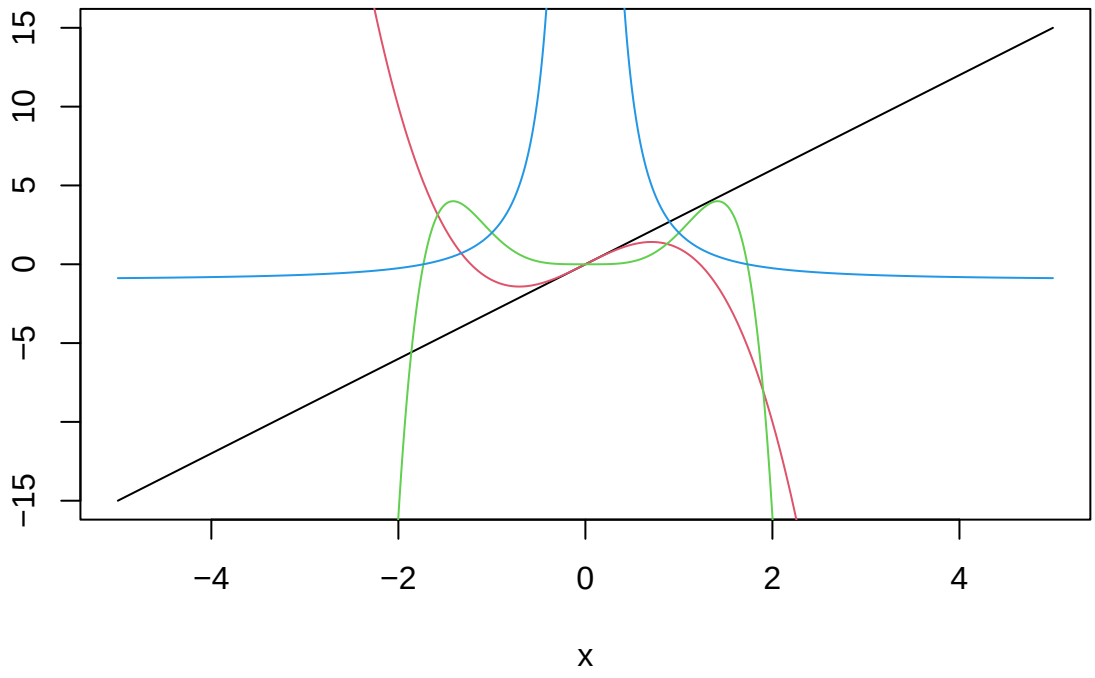
```
# plot the function and suggested modifications
y <- seq(0, 2e5, by = 100)

# set graphical parameters 'line width' to 1.5 (by default 1)
# and 'line type' to '2=dashed' (default 1=solid)
par(lwd = 1.5, lty=2)

plot(y, T(y), type = 'l')
lines(y, T(y - 1e4), col = 2)
lines(y, T(.95*y), col = 3)
lines(y, pmax(T(y) - 1e3, 0), col = 4)
lines(y, .9*T(y), col = 5)

# add a legend in topleft corner
legend("topleft", c("Regular tax",
                    "Deduct 10k before taxes",
                    "Deduct 5% before taxes",
                    "1k tax credit",
                    "10% tax credit"),
       col = 1:5, lty = 2)
```



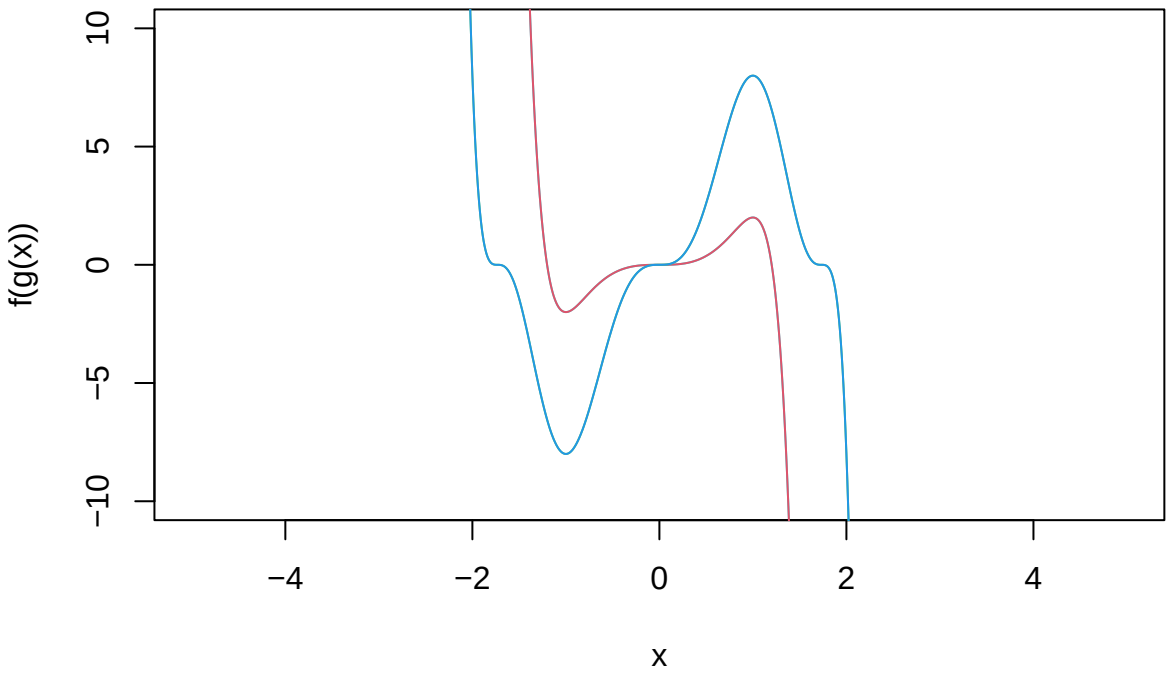
NEW FUNCTIONS FROM OLD

```
f <- function(x) { 3*x - x^3 }
g <- function(x) { x^3 }
```

```
x <- seq(-5, 5, by = .01)
plot(x, f(x) + g(x), type = 'l', ylab = '')
lines(x, f(x) - g(x), col = 2)
lines(x, f(x) * g(x), col = 3)
lines(x, f(x) / g(x), col = 4)
```




```
plot(x, f(g(x)), type='l',col=5, ylim=c(-10,10))
lines(x, 3*x^3-x^9, col=2)

lines(x, g(f(x)), col=3)
lines(x,(3*x-x^3)^3,col=4)
```




```
f(g(1))
```

```
## [1] 2
```

```
g(f(1))
```

```
## [1] 8
```

PLOTTING THE INVERSE

```
x <- seq(-2, 2, 0.01)
y <- seq(-9, 7, 0.01)

f <- function(x) { x^3-1 }

# if b is odd and/or a >=0 (not necessarily both conditions),
# then the b-th root of a exists,
# otherwise not
nthroot <- function(a,b) ifelse(b%%2==1 | a>=0, sign(a)*abs(a)^(1/b), NaN)
nthroot(-1,3)
```

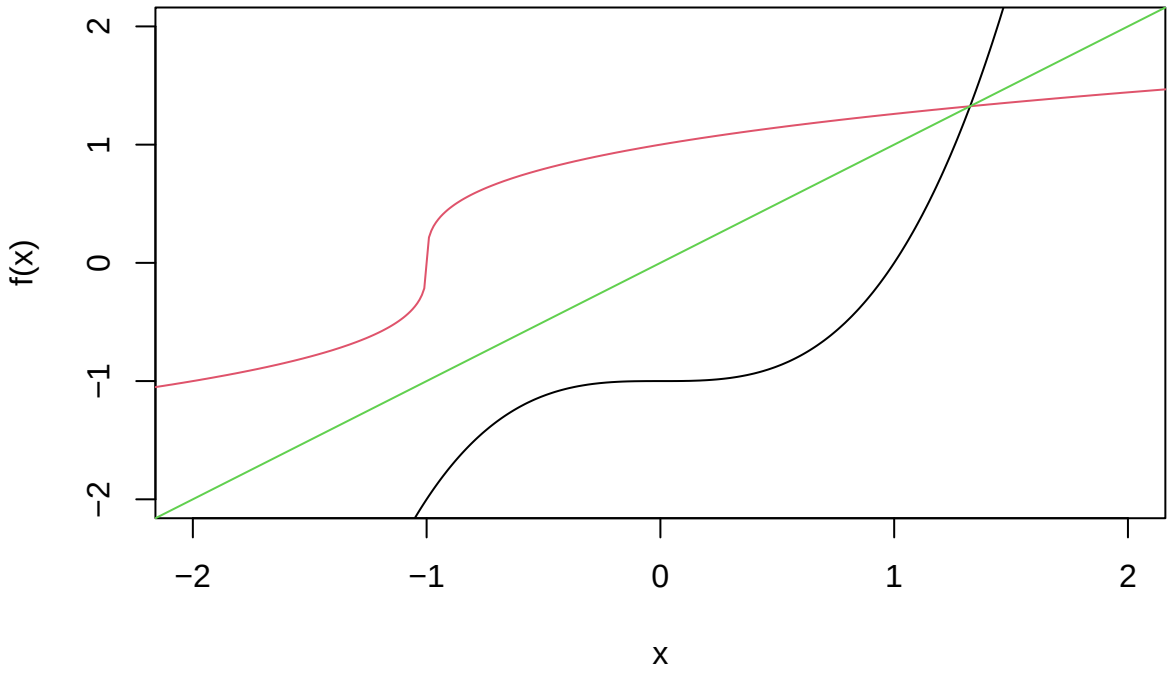
```
## [1] -1
```

```
nthroot(-1,2)
```

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

```
# inverse of f: set parameter a = y+1 and b = 3
f_inv <- function(y) nthroot(y+1,3)
```

```
plot(x, f(x), type='l', ylim=c(-2,2))
lines(y, f_inv(y), col=2)
abline(0, 1, col=3)
```




```
(-8)^(1/3)
```

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

```
nthroot(-8,3)
```

```
## [1] -2
```

```
f_inv(-9)
```

```
## [1] -2
```