Optimizers in R

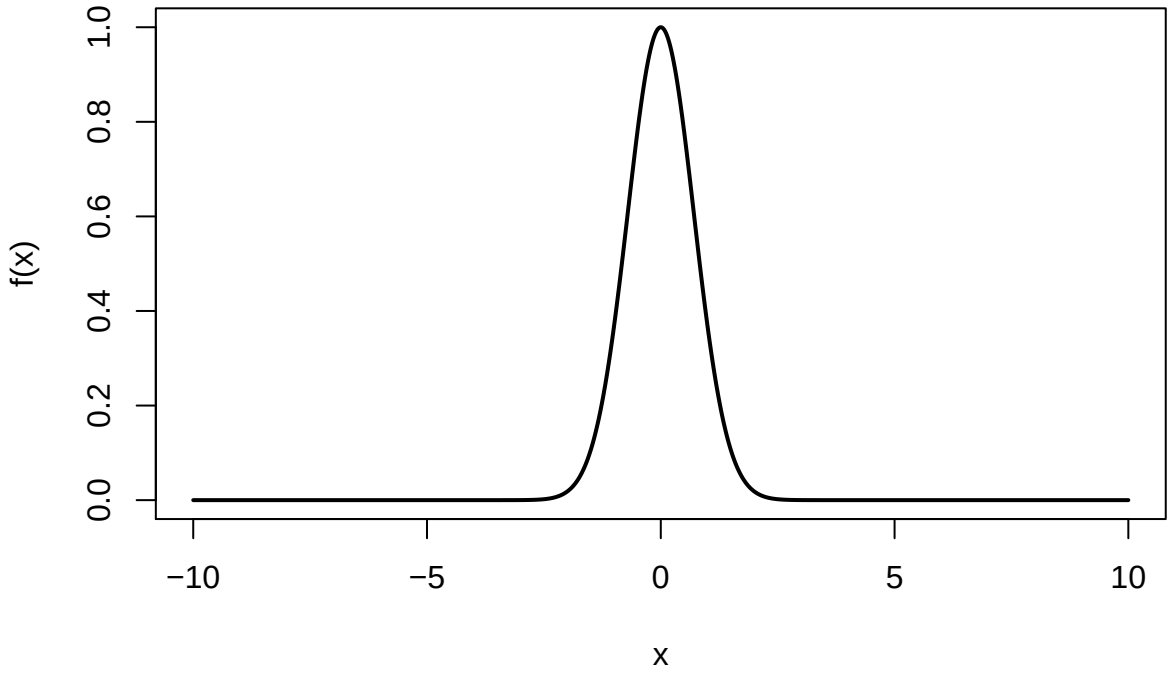
Adriana F. Chávez De la Peña

11 February, 2022

Exercise 1

Using a hill-climbing optimization algorithm, find the maximum of $\log(f(x))$ where $f(x) = e^{-x^2}$. Compare to the maximum of $f(x)$.

```
#Define function
f <- function(x){exp(-x^2)}
x <- seq(-10,10,0.01)
plot(x,f(x), type="l", lwd=2)
```

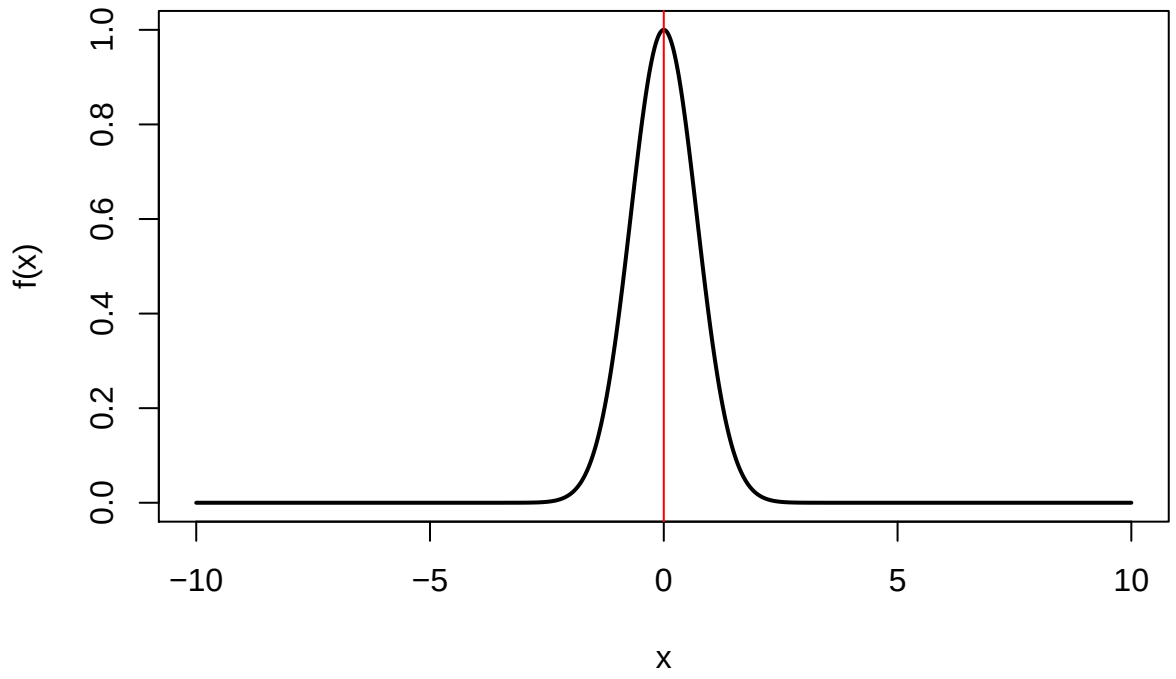



```
#Define Hill-climbing algorithm
step_size = 0.05
current_x  = 1
current_y <- f(current_x)
n_steps = 200;

  for(i in 1:n_steps){
        new_x  = current_x + step_size
```

```
        new_y <- f(new_x)
        if(new_y >= current_y){
            current_x = new_x
            current_y = new_y}
        else
            {step_size = -step_size * 2/3}}
print(current_x)
```

```
## [1] 7.619184e-10
```

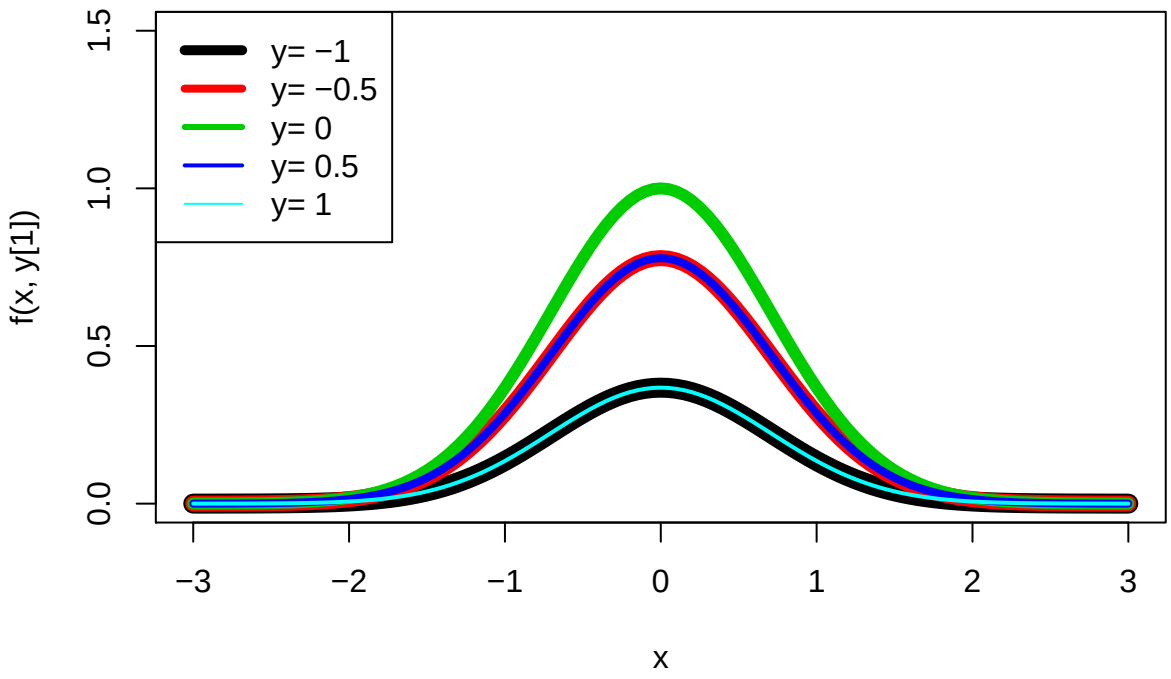
```
plot(x,f(x), type="l", lwd=2)
abline(v=current_x, col="red")
```



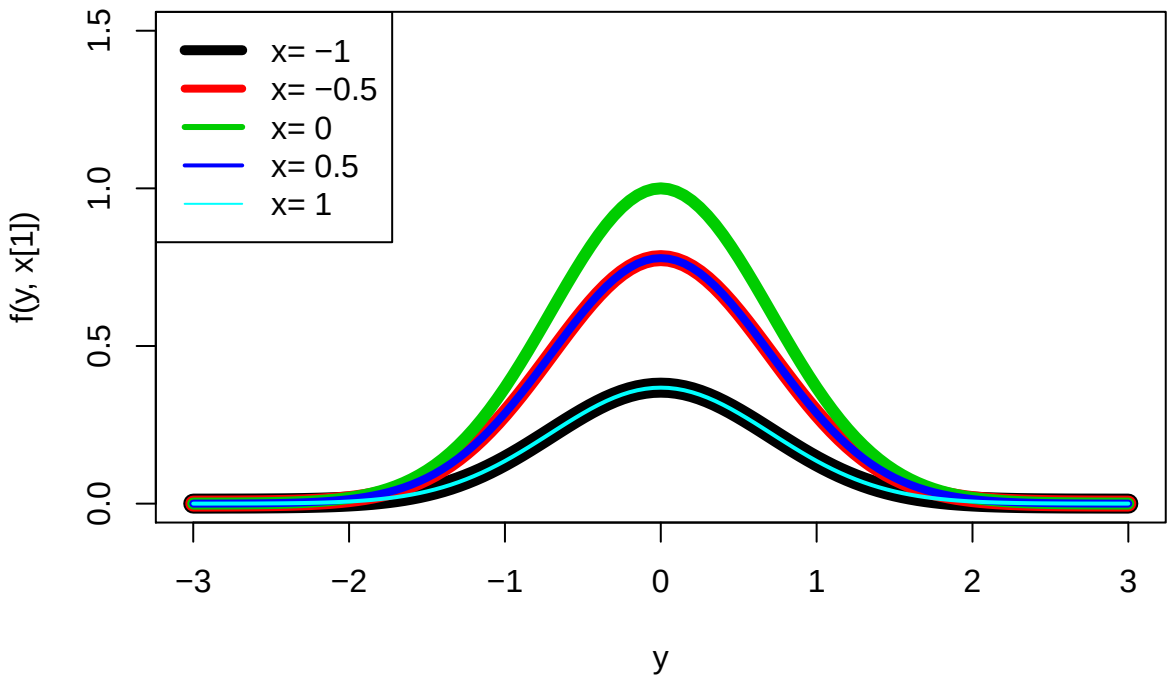
Exercise 2

Using a hill-climbing optimization algorithm, find the maximum of $f(x, y) = e^{-x^2-y^2}$.

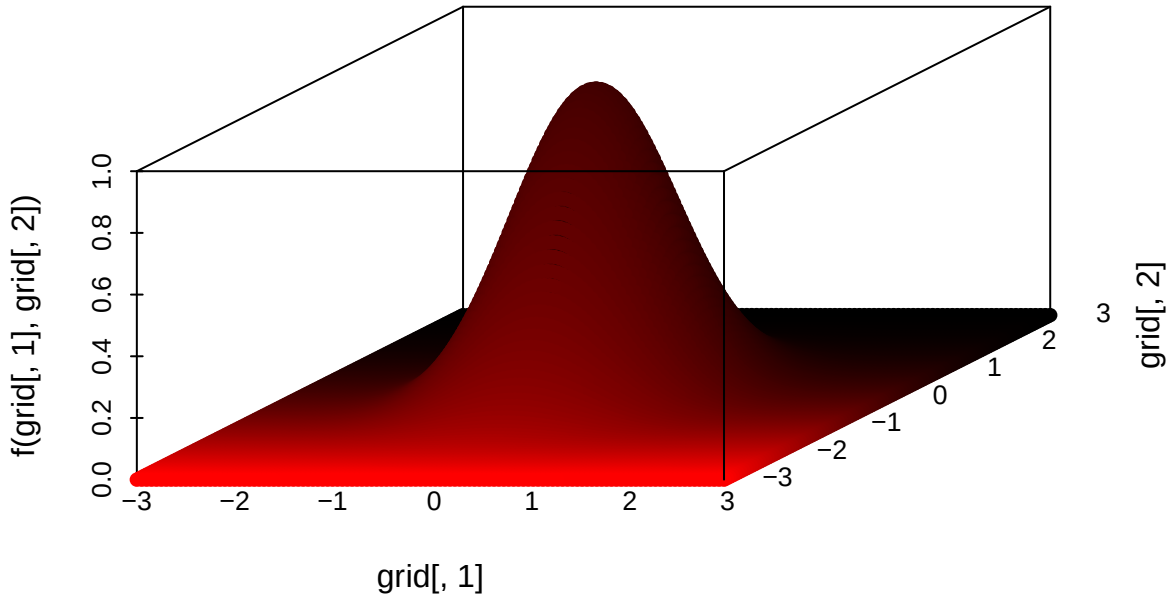
```
#Define function
f <- function(x,y){exp(-x^2-y^2)}

##########2D plot 1
x <- seq(-3,3,0.05)
y <- seq(-1,1,0.5)
y_i <- NULL
plot(x,f(x,y[1]), type="l", ylim=c(0,1.5), col=1, lwd=length(y)*2)
for(a in 2:length(y)){
y_i[a-1] = y[a]
lines(x,f(x,y_i[a-1]), col=a, lwd=(length(y)-(a-1):1)*2)}
legend("topleft", paste("y=",legend=c("-1",y_i)), lwd=length(y):1, col=1:length(y))
```




```
##########2D plot 2
y <- seq(-3,3,0.05)
x <- seq(-1,1,0.5)
x_i <- NULL
plot(y,f(y,x[1]), type="l", ylim=c(0,1.5), col=1, lwd=length(x)*2)
for(a in 2:length(x)){
x_i[a-1] = x[a]
lines(y,f(y,x_i[a-1]), col=a, lwd=(length(x)-(a-1):1)*2)}
legend("topleft", paste("x=",legend=c("-1",x_i)), lwd=length(x):1, col=1:length(x))
```




```
x <- seq(-3,3,0.05)
grid <- expand.grid(x, y)
library(scatterplot3d)
scatterplot3d(grid[,1], grid[,2], f(grid[,1],grid[,2]), highlight.3d = TRUE, col.axis = "black",
col.grid = "lightblue", pch = 16)
```




```
step_x = 0.025
step_y = 0.025
current_x = -10
current_y = -10

current_z = f(current_x, current_y)

nIter = 10000000
for(i in 1:nIter){
    new_x = current_x + step_x;
    new_z = f(new_x, current_y);
    if(new_z >= current_z){
        current_x = new_x;
        current_z = new_z;}
    else{step_x = -step_x * 2/3}

    new_y = current_y + step_y;
    new_z = f(current_x, new_y);
    if(new_z >= current_z){
        current_y = new_y;
        current_z = new_z;}
    else{
        step_y = -step_y * 2/3;}
    if(current_z > .9999){break}}

print(c(current_x,current_y))
```

```
## [1] 7.48776e-14 7.48776e-14
```

Exercise 3

Use simmulated annealing to find the global **minimum** of $C(x,y) = (4 - 2.1x^2 + \frac{1}{3}x^4)x^2 + xy + 4(y^2 - 1)y^2$.

```
#Define function
f <- function(x,y){
  a <- (4)-(2.1*(x^2))+((1/3)*(x^4))
  b <- x*y
  c <- 4*((y^2)-1)*(y^2)
  return(a+b+c)}

x <- seq(-1,1,0.05)
y <- seq(-1,1,0.05)
grid <- expand.grid(x, y)
library(scatterplot3d)
scatterplot3d(grid[,1], grid[,2], f(grid[,1],grid[,2]), highlight.3d = TRUE, col.axis = "black",
col.grid = "lightblue", pch = 16)
```

```
library(stats)

zvalue = f(x, y);
Te = .1;
counter = 0;
current_x = 4 * runif(1,0,1) - 2;
current_y = 2 * runif(1,0,1) - 1;
current_z = f(current_x, current_y);

nIter = 100000
for(i in 1:nIter){
    if(Te > .001){
        rndm <- rnorm(3,0,1)
        rndm2 <- runif(3,0,1)
        new_x = min(max(current_x + rndm[1]*Te*10,-2),2)
        new_y = min(max(current_y + rndm[2]*Te*10,-1),1)
        new_z = f(new_x, new_y)

        if(new_z <= current_z | rndm2[1] < exp(-(new_z-current_z)/Te)){
          current_x = new_x
          current_y = new_y
          current_z = new_z}

        counter = counter + 1

        if(counter > 100){
          counter = 0
          Te = Te * .025
          print(c(Te,current_x,current_y,current_z))}

        if(current_z < -2 | Te < .001)
        {break}}
        else{break}}

print(c(current_x,current_y,current_z))
```