Exercise Set 0: Prerequisite Knowledge

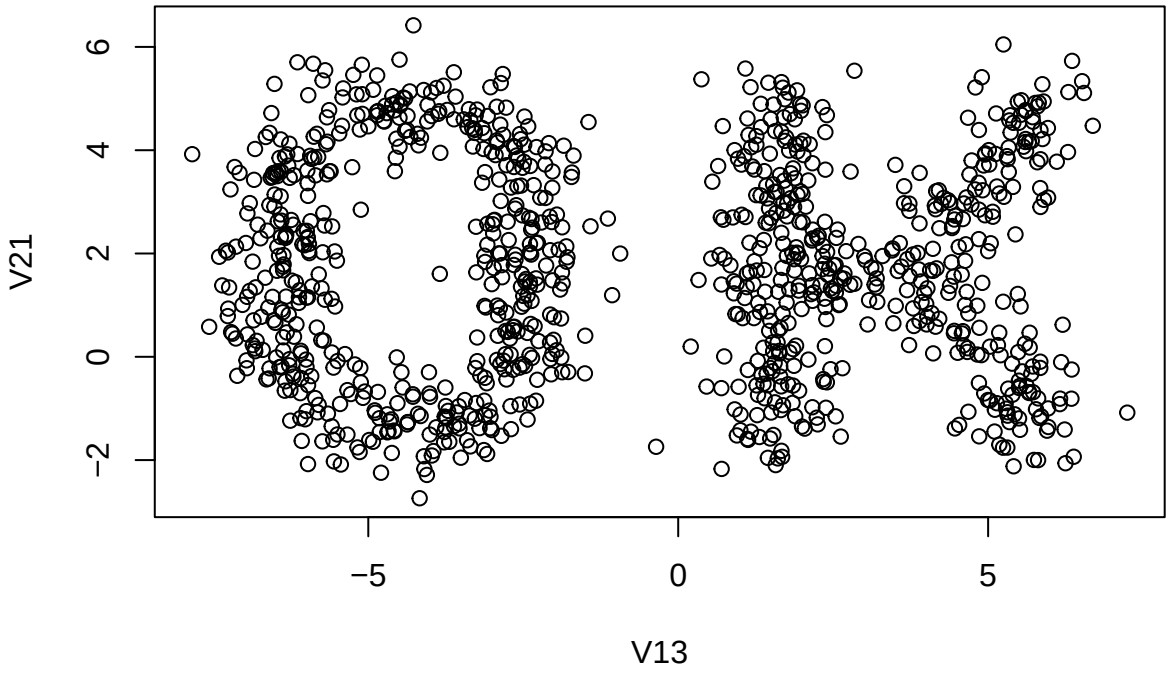
Enrico Buratto

4/10/2021

Problem 1

Task a

```
x <- read.csv("./x.csv")
variances <- sapply(x, var)
variances <- variances[order(variances, decreasing = TRUE)]
first <- x[names(variances[1])]
second <- x[names(variances[2])]
to_plot <- data.frame(first, second)
plot(to_plot)
```



Problem 2

Task a

Task b

Problem 3

Task a

Task b

Problem 4

Task a

Task b

Problem 5

Task a

The value $x \in \mathbb{R}$ that minimizes the value of $f(x)$ can be found calculating the first derivative and setting it to zero, so $f'(x) = 0$. The first derivative is $f'(x) = 4ax^3 + b$, and it's easy to find that it has only one solution (or, better, three coincident solutions), that is $x_0 = \sqrt[3]{\frac{-b}{4a}}$. This point is a critical point; in order to be a minimum, it must be the point for which the second derivative on that point is greater than 0, so $f"(x) = 12ax^2 > 0$ on $x_0 = \sqrt[3]{\frac{-b}{4a}} \Rightarrow 12a(\sqrt[3]{\frac{-b}{4a}})^2 > 0$, that is true for every $a > 0$. The value that minimized $f(x)$ is then $\sqrt[3]{\frac{-b}{4a}}$ with $a > 0$.

Task b

Since the first derivative has three coincident solutions, it means that there is only one critical point; furthermore, in order to be that critical point a minimum point, the second derivative in that point must be greater than 0. We can then conclude that the condition for the function to have a finite and unique point is to have $a > 0$, since with this condition $(\sqrt[3]{\frac{-b}{4a}})^2$ can't be undefined ($a \neq 0$) and the equation is true.

Problem 6

Task a

```
1. Fibonacci(n)
2.    a <- 1
3.    b <- 1
4.    print a
5.    print b
6.    i <- 2
7.    while i<n
8.        c = a+b
9.        print c
10.       a = b, b = c
11.       i += 1
```

Task b

The time complexity of the algorithm is $O(n)$; let's analyze the algorithm in-depth:

- Instruction 1. is constant ($O(1)$), since it only receives a number n in input
- Instructions $2. - 6.$ and $8. - 11.$ can be executed in $O(1)$ time, since they are only variable assignments and prints
- Instructions $8. - 11.$ are repeated $O(n)$ times

The algorithm time complexity is then a sum of constants, except for instructions $8. - 11.$ that are repeated exactly $n - 2$ times ($O(n)$).