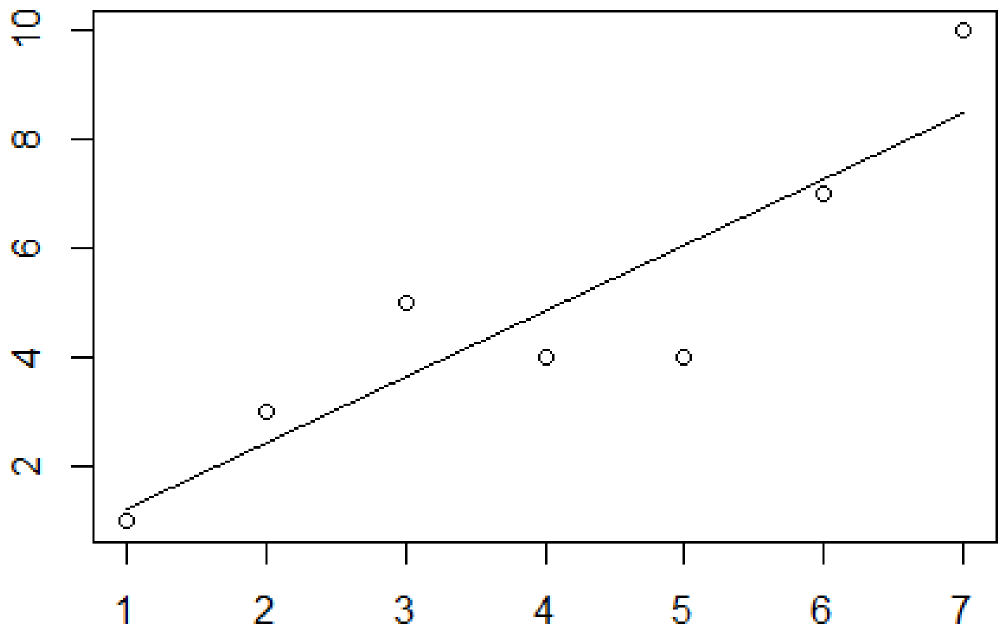
SEGMENTIRANA METODA NAJMANJIH KVADRATA

Mihael Petrinjak

METODA NAJMANJIH KVADRATA

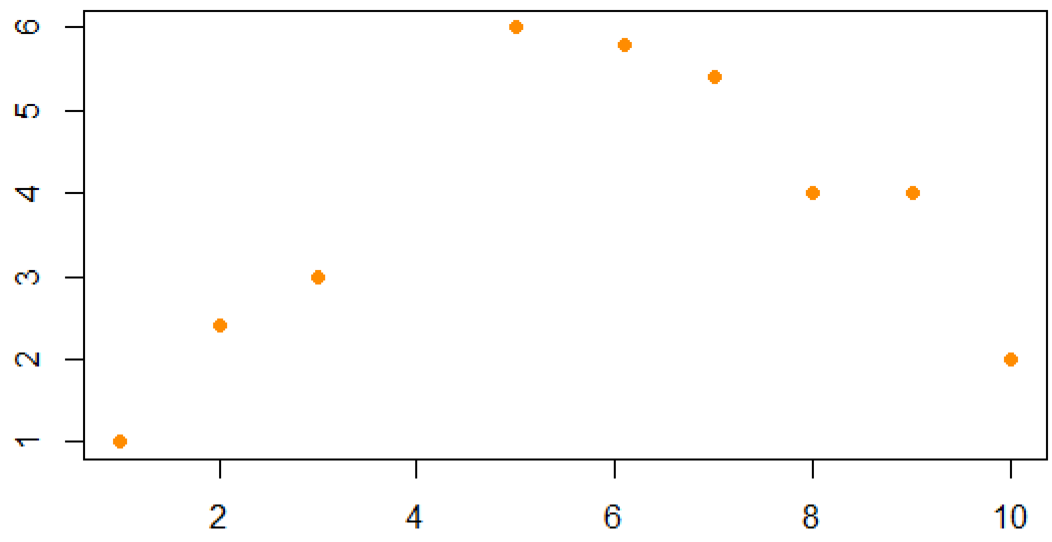
P skup točaka $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ takav da $x_1 < x_2 < ... < x_n$

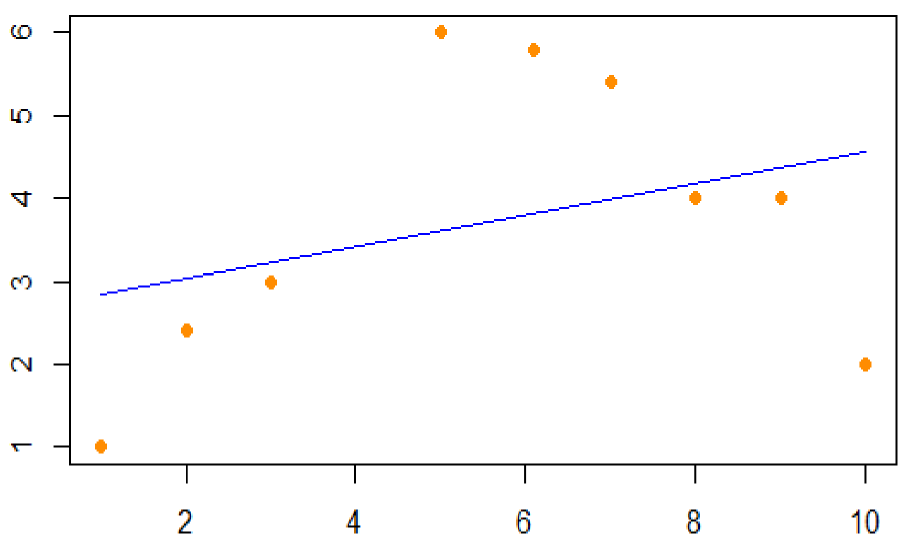


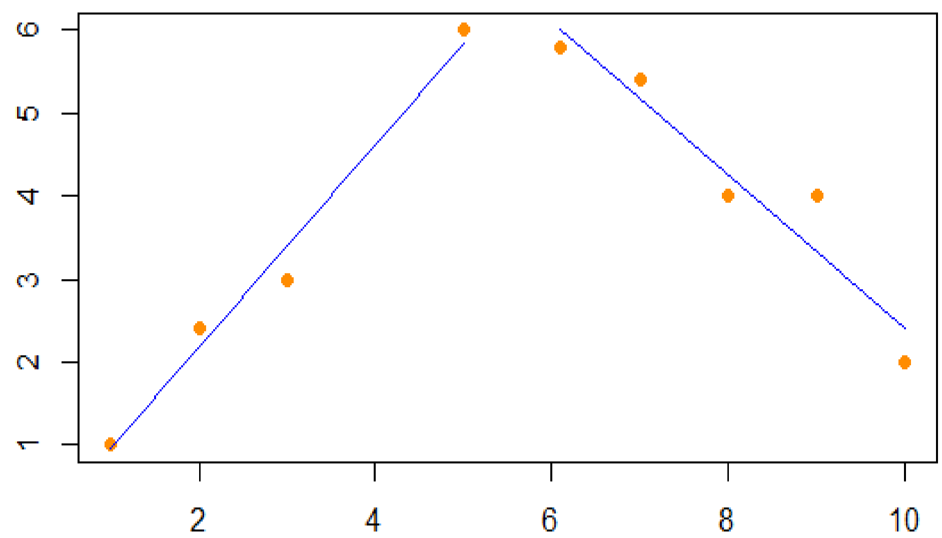
$$E(L, P) = \sum_i (y_i - ax_i - b)^2$$

$$a = \frac{n \sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)}{n \sum_i x_i^2 - (\sum_i x_i)^2} \qquad b = \frac{\sum_i y_i - a \sum_i x_i}{n}$$

ZAŠTO SEGMENTI







- Može se dogoditi da radimo s podacima za čiju aproksimaciju je prikladno uzeti **nekoliko** pravaca.
- U tom slučaju potrebno je nekako **podijeliti skup**.
    - Kasnije uvodimo pojam **segmenta.**
- Kako smisleno podijeliti skup **bez vizualizacije** podataka?

FORMULACIJA PROBLEMA

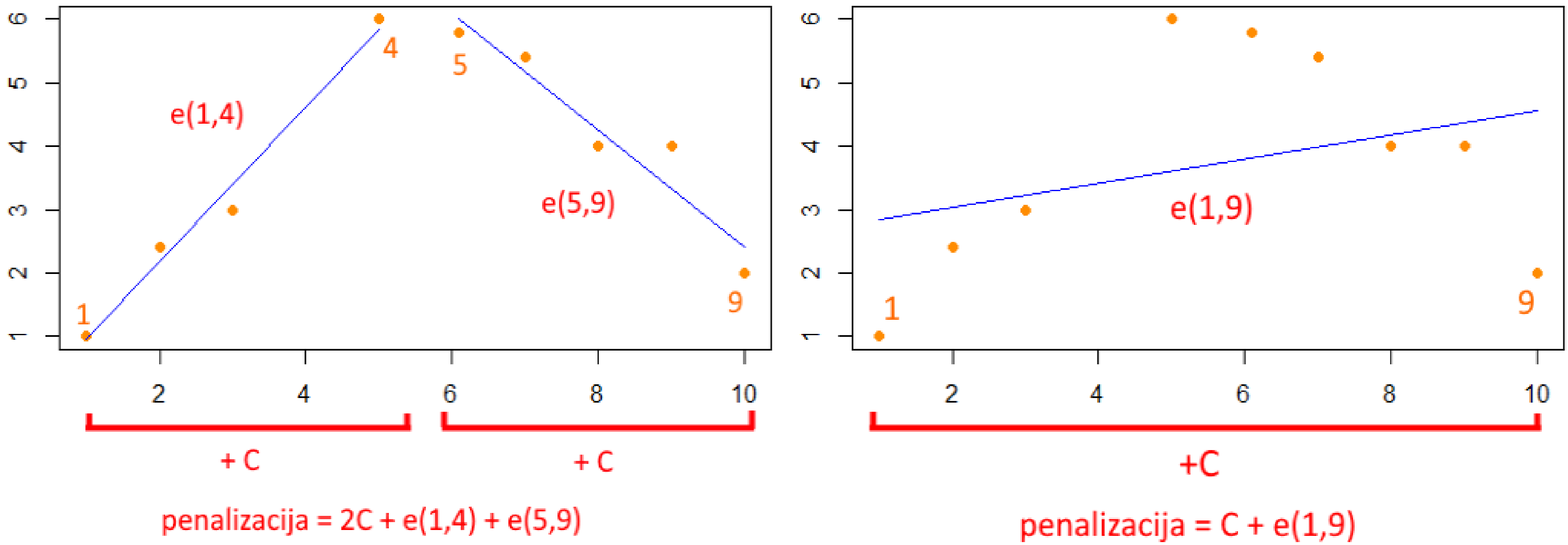
Segment u P je skup oblika $\{p_i, p_{i+1}, ..., p_j\}$ gdje $1 \leq i \leq j \leq n$.

- Tražimo **particiju** skupa P čiji elementi su isključivo segmenti u P.

Penalizacija particije F se definira kao suma sljedećih izraza.

(i) Broj segmenata u F pomnožen s konstantom $C > 0$.

(ii) Za svaki segment $S \in F$ vrijednost $E(L, S)$ gdje je L pravac dobiven *metodom najmanjih kvadrata*.

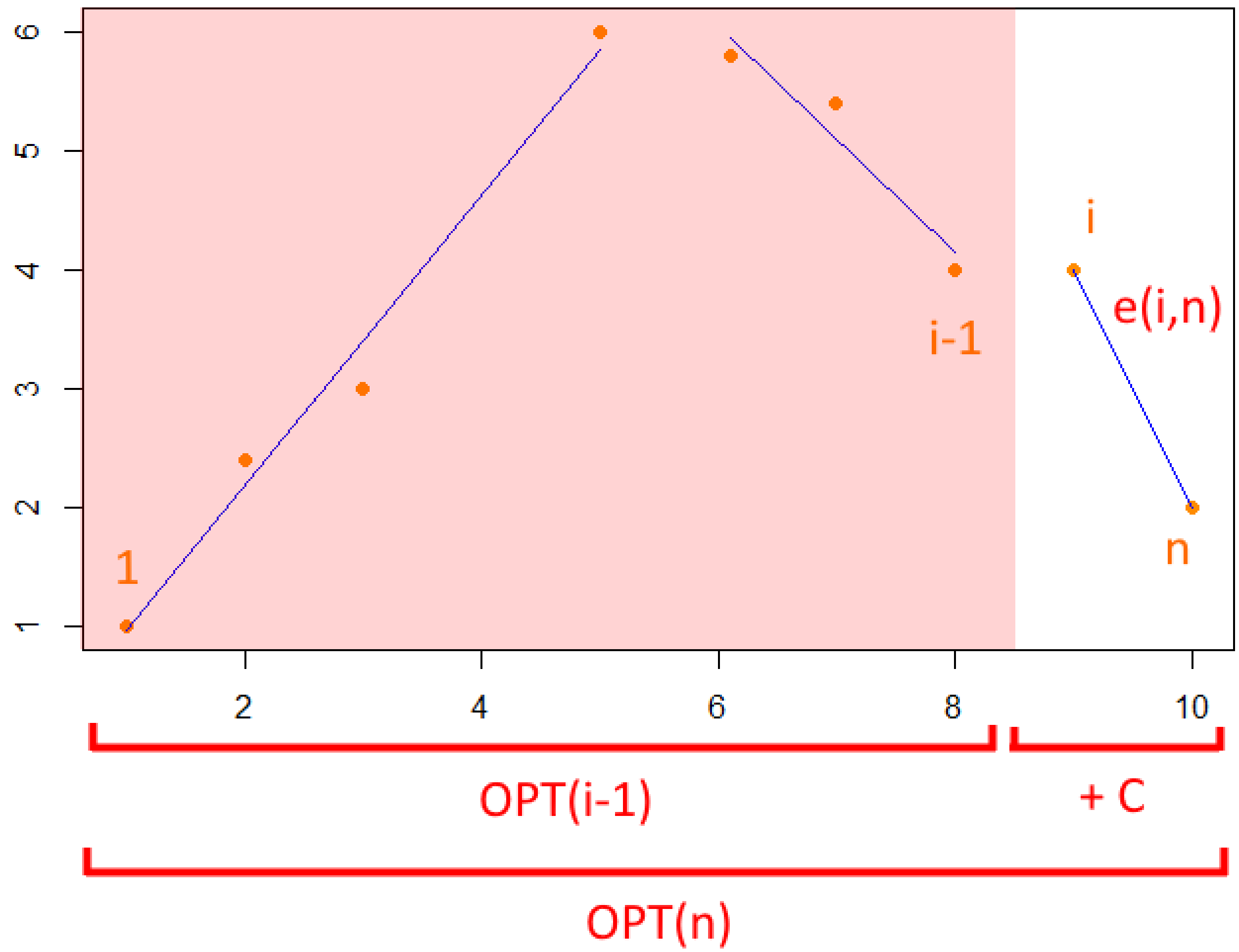


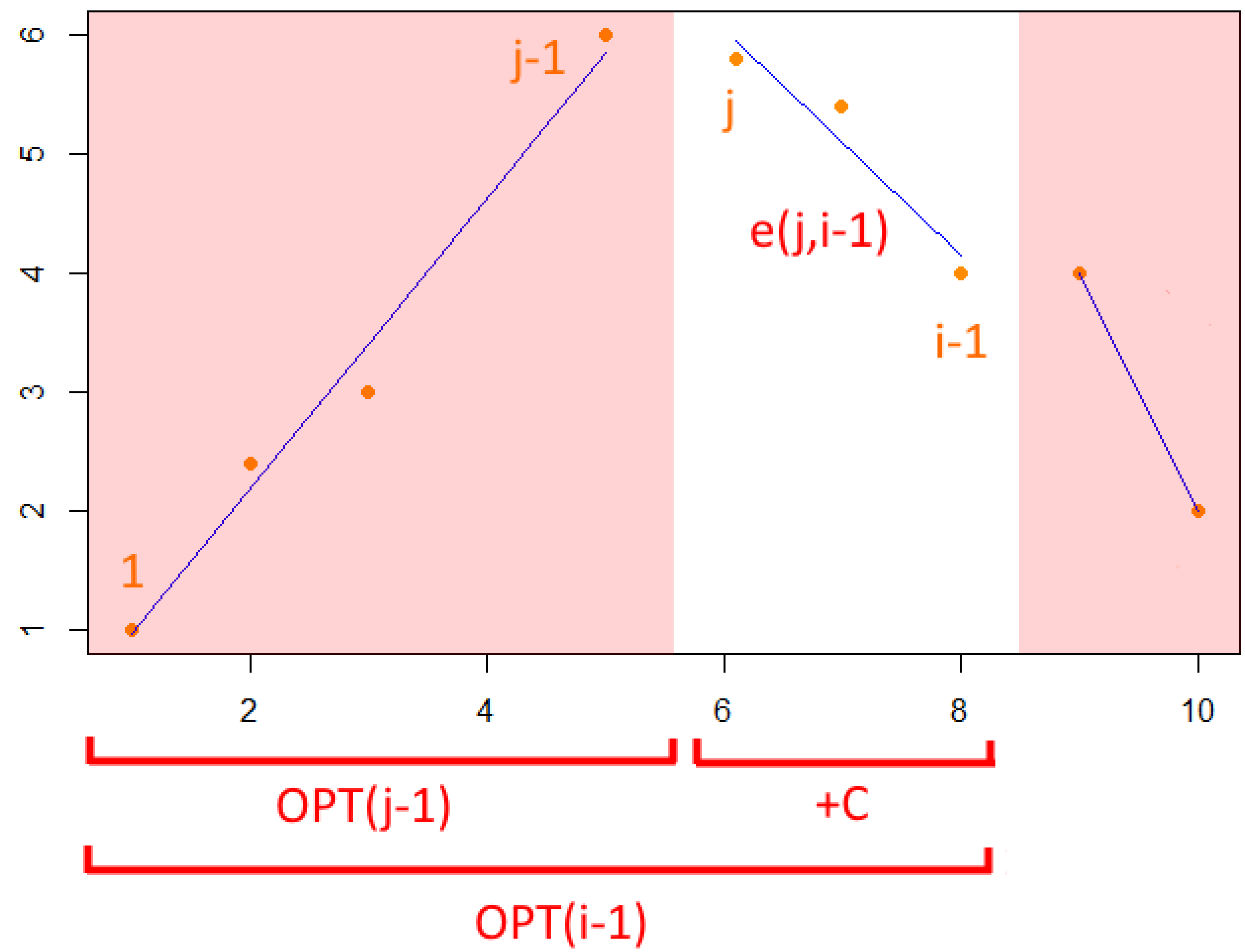
- $e(i,j) = E(L, \{p_i, \ldots, p_j\})$ ... (minimalna) greška aproksimacije pravcem
- Povečanjem **broja segmenata** povečava se suma konstanti C, ali se smanjuje zbroj **grešaka** aproksimacije.
    - Za dovoljno veliki C će penalizacija u prvom slučaju nadmašiti vrijednost penalizacije u drugom slučaju iako je e(1,9) jako velik broj.
    - C modelira cijenu dodavanja segmenata u particiju.
- Cilj algoritma je za dani C pronaći particiju skupa P **najmanje penalizacije.**

PRINCIP OPTIMALNOSTI

- Pretpostavimo da znamo optimalnu particiju za P i njenu penalizaciju **OPT(n).**

Slutnja 2.1.1. *Ako je zadnji segment optimalne particije* $\{p_i, ..., p_n\}$, *onda je vrijednost optimalnog rješenja jednaka* $OPT(n) = e_{i,n} + C + OPT(i-1)$





- Promatramo početni komad $p_1, \ldots, p_{i-1}$.
- Potrebno je odabrati indeks **j** tako da izraz **OPT(i-1) = OPT(j-1) + C + e(j , i-1)** ima minimalniu vrijednost. U suprotnom vrijednost izraza **OPT(n)** u kojem se javlja **OPT(i-1)** neće biti minimalna.

Optimalno rješenje podproblema na $\{p_1, ..., p_j\}$ *dano je s*

$$OPT(j) = \min_{1 \le i \le j}(e_{i,j} + C + OPT(i-1))$$

uz početni uvjet $OPT(0) = 0$.

$$OPT(j) = \min \begin{cases} e_{1,j} + C + OPT(0), \\ e_{2,j} + C + OPT(1), \\ \qquad \vdots \\ e_{j,j} + C + OPT(j-1) \end{cases}$$

ALGORITAM

```
procedure SMNK(P, n, C)
    Polje OPT[0...n]
    OPT[0] ← 0
    Matricu E[1...n][1...n] inicijaliziraj na (0).
    for svi parovi i ≤ j do
        E[i][j] ← e_{i,j}
    end for
    for 1 ≤ j ≤ n do
        Računaj OPT[j] po rekurziji OPT(j) = min{...}
    end for
return OPT
end procedure
```

$$OPT(j) = \min \begin{cases} e_{1,j} + C + OPT(0), \\ e_{2,j} + C + OPT(1), \\ \vdots \\ e_{j,j} + C + OPT(j-1) \end{cases}$$

```
procedure IZVADI SEGMENTE(OPT, j, C)
    if j = 0 then
        Gotovo.
    else
        Pronađi i koji minimizira e_{i,j} + C + OPT[i − 1].
        return {p_i, ..., p_j}
        return IZVADI SEGMENTE(OPT, i − 1, C)
    end if
end procedure
```

```
procedure MAIN
    P, n, C    zadani.
    OPT ← SMNK(P, n, C)
    particija ← IZVADI SEGMENTE(OPT, n, C)
end procedure
```

```r
segmentedLeastSquares <- function(pX, pY, n, C) {
  # OPT[k] = value of optimal solution for p1, p2,..., pk-1
  # our goal is OPT[n+1]
  OPT <- c()
  OPT[1] = 0
  E <- matrix(0, ncol=n, nrow=n)

  for (i in 1:n) {
    for (j in i:n) {
      if (j - i >= 1) E[i,j] <- minError(pX, pY, i, j)
    }
  }

  for (j in 1:n) {

    candidates <- c()
    for (i in 1:j) candidates[i] <- E[i,j] + C + OPT[i]

    OPT[j + 1] <- min(candidates)
  }

  partition <- findSegments(n, C, OPT, E)
  return(partition)
}
```

u R-u indeksi vektora počinju od 1 !!! pomak u OPT

inicijalizacija matrice u koju spremamo e(i,j)

1:n je oznaka za (1,2,...n)

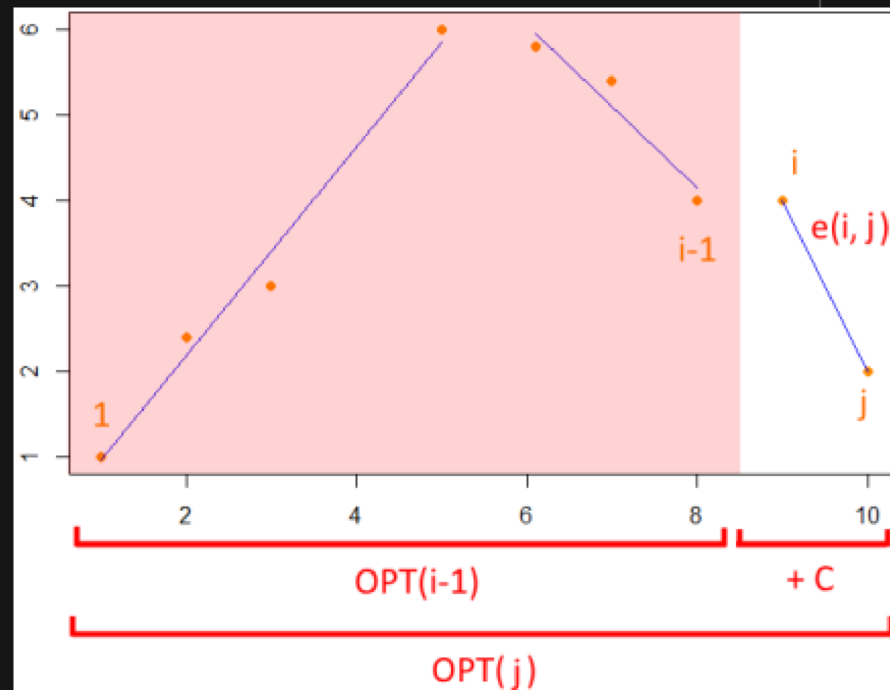
za svaki par i <= j računamo e(i,j)

$$OPT(j) = \min \begin{cases} e_{1,j} + C + OPT(0), \\ e_{2,j} + C + OPT(1), \\ \vdots \\ e_{j,j} + C + OPT(j-1) \end{cases}$$

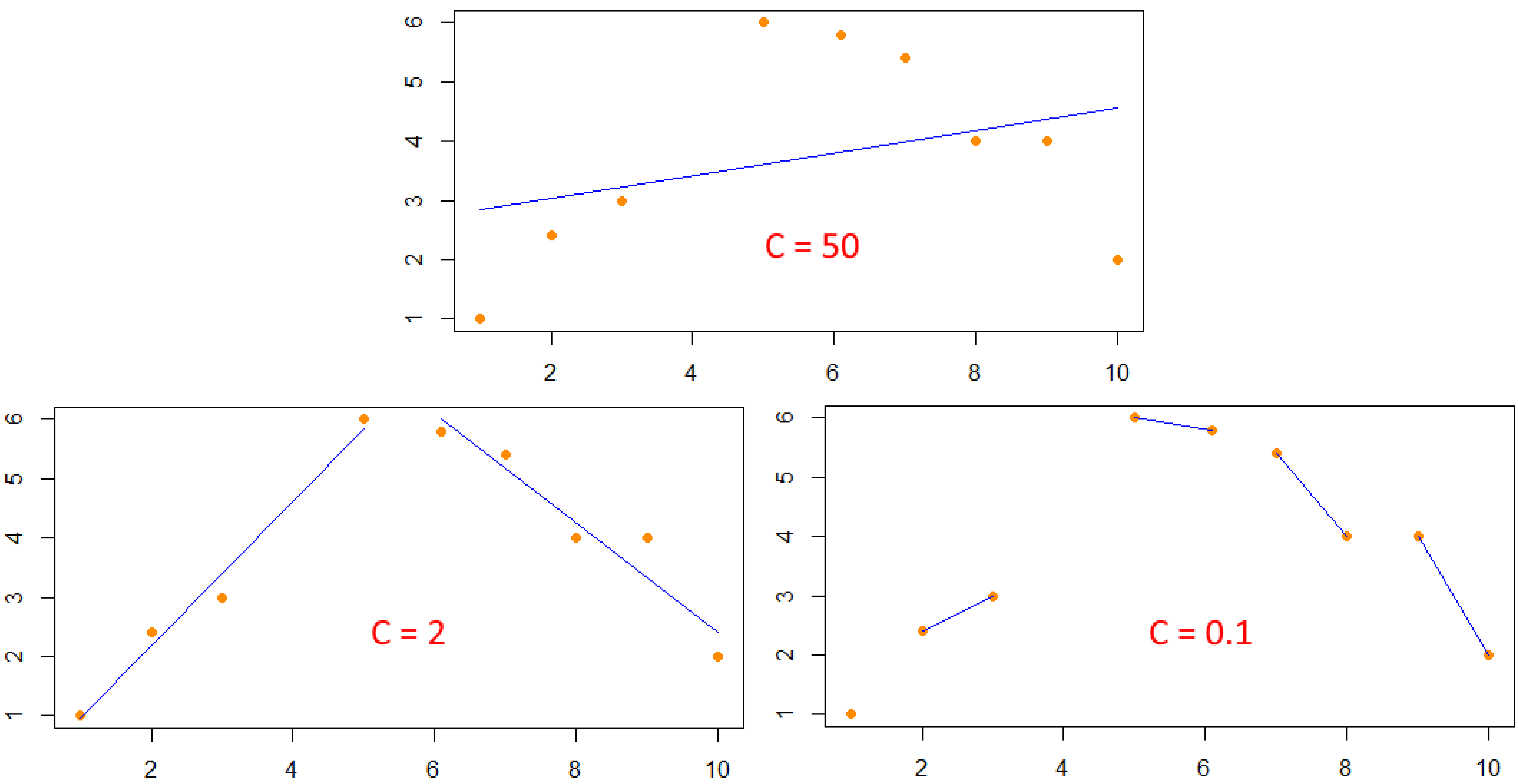
u OPT se nalaze minimizirane penalizacije početnih komada u P, tražimo particiju najmanje sume penalizacija

```r
findSegments <- function(j, C, OPT, E) {
  min <- .Machine$double.xmax
  index <- NULL

  for (i in 1:j) {
    value <- E[i, j] + C + OPT[i]
    if(value < min) {
      min <- value
      optIndex <- i
    }
  }

  s2 <- c(optIndex, j)
  if(optIndex==1) return(s2)

  s1 <- findSegments(optIndex-1, C, OPT, E)
  return( c(s1, s2) )
}
```

- biramo i t.d. OPT(j) bude minimalan



- ponovimo poziv za komad lijevo od izabranog i


C = 50
C = 2
C = 0.1

SLOŽENOST

- $\mathcal{O}$(n) - računanje pravca L(a,b) za E(L,S)
- $\mathcal{O}$(n) - računanje E(L,S)
- $\mathcal{O}(n^2)$ - broj parova e(i,j)
- $\mathcal{O}(n^3)$ - računanje svih potrebnih e(i,j)

$$OPT(j) = \min \underbrace{\begin{cases} e_{1,j} + C + OPT(0), \\ e_{2,j} + C + OPT(1), \\ \vdots \\ e_{j,j} + C + OPT(j-1) \end{cases}}_{\text{j puta}}$$

$1 \leq j \leq n$

- $\mathcal{O}(n^2)$ - OPT popunjen
- U najgorem slućaju FIND SEGMENTS se poziva n/2 puta (kao za C = 0.1), a traje $\mathcal{O}$(n).
    - $\mathcal{O}(n^2)$ - iščitavanje segmenata iz OPT

- $\mathcal{O}(n^2)$ za algoritam nakon što imamo sve e(i,j)

LITERATURA

Jon Kleinberg i Eva Tardos, *Algorithm design*, Addison Wesley, 2006.