Coding session: Local Similarities in Patient Trajectories
The Smith-Waterman algorithm

Juan Miguel García Gómez & Ramón Pardo
BDSLab-ITACA, Dpto. de Física Aplicada
UPV

4 de junio de 2024

Índice

1. Objetive

Dynamic programming is the ideal framework for understanding sequence alignment algorithms. The Smith-Waterman algorithm obtains the optimal local alignments between two sequences.

In this practice we will implement the Smith-Waterman algorithm in the R programming language. Then, we will apply it to compare patient trajectories to find patients with similar clinical behaviours in some parts of their clinical pathways. The similarity of patients' trajectories to the trajectories of historic patients may help understanding their potential outcomes.

2. Material

- A sequence with a patient trajectory is composed by 1 to m elements with the syntaxis $c+s+d$, where c is the total cholecterol level in mg/dL, s is the systolic blood preasure in $mmHg$ and d is the diastolic blood preasure in $mmHg$. Each element represents a set of measures acquired together and the elements in a sequence follow a temporal order. Example: 150.0+100.0+70.0 153.0+100.0+70.0 250.0+160.0+110.0
- R $> 3{,}5{,}0$ (http://www.r-project.org)
- Template with the R code: clinicalPathwaysAligment_by_DynamicProgramming_TODO.R

3. Evaluation

The students must submit their implementation of the algorithm and the testing script after solving the next exercises.

4. Exercise 1. Implementing the Smith-Waterman algorithm

4.1. Objetive

Complete the general loop of the algorithm in the SmithWaterman function:

4.2. Note

- The variable `trellis` is a matrix of m rows and n columns where m is lenght(S)+1 y n es length(R)+1
- The `delta` matrix (or function) is the score matrix with the score of substituing symbol x by y, including the deletion of x (sustituing "x"by $-$) y the insertion of y (sustituing $-$ by y)
- The matrices backi and backj indicates the coordenates of the previous node of the editing sequence to allow recovering the local aligment

4.3. To Do

1. Complete the code to calculate the score of an insertion, deletion or Match/Susbtitution to achieve node i, j in the general loop of the SmithWaterman function

```
#General loop
for (i in 2:m)
{
for (j in 2:n)
{
  #deletion of u[i]
costes["B"] <-  #TODO EXERCISE 1

#insertion of v[j]
costes["I"] <-  #TODO EXERCISE 1

#sustitution or match u[i]==v[j]
costes["MoS"] <-  #TODO EXERCISE 1

#cost of the operation
trellis[i,j] <- decisionf(costes)

#preparing the backtracking
operacion <- names(which(costes==trellis[i,j])[1])

if (operacion=="B")
{backi[i,j] <- i-1; backj[i,j] <- j;}
else if (operacion=="I")
{backi[i,j] <- i; backj[i,j] <- j-1;}
else if (operacion=="MoS")
{backi[i,j] <- i-1; backj[i,j] <- j-1;}
else if (operacion=="Ini")
{backi[i,j] <- 1; backj[i,j] <- 1;}
else warning("Not recognized operation in Trellis")

} #for j
} #for i
```

5. Exercise 2. Implementing function cholesterol_level()

5.1. Objetive

Return high, borderline or normal depending the level of total cholesterol according to the clinical guideline.

5.2. Notes

- Total cholesterol levels less than 200 milligrams per deciliter (mg/dL) are considered desirable for adults. A reading between 200 and 239 mg/dL is considered borderline high and a reading of 240 mg/dL and above is considered high. LDL cholesterol levels should be less than 100 mg/dL.

5.3. To Do

1. Complete the code in the cholesterol_level function:

```
cholesterol_level <- function(CHO_TOTAL)
{
  #return "high", "borderline" or "normal" depending
  the level of total cholesterol according to the next guideline:
  #Total cholesterol levels less than 200 milligrams per
  deciliter (mg/dL) are considered desirable for adults.
  A reading between 200 and 239 mg/dL is considered borderline high
  and a reading of 240 mg/dL and above is considered high.

  #TODO EXERCISE 2
}
```

6. Exercise 3. Implementing function blood_preasure_level()

6.1. Objetive

Return high, prehypertension or normal, depending the levels of systolic and diastolic blood preasures accordingto the clinical guideline.

6.2. Notes

- Normal systolic: less than 120 mm Hg diastolic: less than 80 mm Hg
- At Risk (prehypertension) systolic: 120-139 mm Hg diastolic: 80-89 mm Hg
- High Blood Pressure (hypertension) systolic: 140 mm Hg or higher diastolic: 90 mm Hg or higher

6.3. To Do

1. Complete the code in the blood_preasure_level function:

```
blood_preasure_level <- function(BP_systolic,BP_diastolic)
{
  #return "high", "prehypertension" or "normal",
  depending the levels of systolic and diastolic
  blood preasures according to the next guideline:
  # Normal systolic: less than 120 mm Hg diastolic: less than 80 mm Hg
  # At Risk (prehypertension) systolic:
```

```
  120-139 mm Hg diastolic: 80-89 mm Hg
  # High Blood Pressure (hypertension) systolic:
  140 mm Hg or higher diastolic: 90 mm Hg or higher

  #TODO EXERCISE 3

}
```

7. Exercise 4. Implementing the delta score function

7.1. Objetive

Return the score of the delta function by adding the scores for cholesterol and blood preasure delta levels

7.2. Notes

- delta function splits the elements from ui and vj in three tokens each: cholecteron, systolic preasure and dyastolic preasure
- By calling cholesterol_level() function transforms the quantitative values to the corresponding cholecterol levels
- By calling blood_preasure_level() function transforms the quantitative values in the corresponding blood preasure levels
- delta_CHO_TOTAL is the scoring matrix to compare cholecterol levels
- delta_BLOOD_PREASURE is the scoring matrix to compare blood preasure levels
- High Blood Pressure (hypertension) systolic: 140 mm Hg or higher diastolic: 90 mm Hg or higher

7.3. To Do

1. Complete the code in the blood_preasure_level function:

```
#delta matrix for cholecterol levels
match <- 1
jump1 <- -1
jump2 <- -2
gap <- -1 #insert/delete
delta_CHO_TOTAL <-
matrix(c(match,jump1,jump2,gap,
jump1,match,jump1,gap,
jump2,jump1,match,gap,
gap,gap,gap,Inf),
dimnames=list(c("normal","borderline","high","-"),
c("normal","borderline","high","-")),
```

```
nrow=4,ncol=4,byrow=TRUE)

#delta matrix for blood preasure levels
match <- 1
jump1 <- -1
jump2 <- -2
gap <- -1 #insert/delete
delta_BLOOD_PREASURE <-
matrix(c(match,jump1,jump2,gap,
jump1,match,jump1,gap,
jump2,jump1,match,gap,
gap,gap,gap,Inf),
dimnames=list(c("normal","prehypertension","high","-"),
c("normal","prehypertension","high","-")),
nrow=4,ncol=4,byrow=TRUE)

delta <- function(ui,vj)
{
  ui_elements <- strsplit(ui,"\\+")[[1]]
  eli <- ui_elements[1]
  if (eli != "-"){
    ui_CHO_TOTAL <- as.numeric(eli)
    ui_BP_systolic <- as.numeric(ui_elements[2])
    ui_BP_diastolic <- as.numeric(ui_elements[3])
    ui_cholesterol_level <- cholesterol_level(ui_CHO_TOTAL)
    ui_blood_preasure_level <-
    blood_preasure_level(ui_BP_systolic,ui_BP_diastolic)
  }
  else{
    ui_cholesterol_level <- "-"
    ui_blood_preasure_level <- "-"
  }

  vj_elements <- strsplit(vj,"\\+")[[1]]
  elj <- vj_elements[1]
  if (elj != "-"){
    vj_CHO_TOTAL <- as.numeric(eslj)
    vj_BP_systolic <- as.numeric(vj_elements[2])
    vj_BP_diastolic <- as.numeric(vj_elements[3])
    vj_cholesterol_level <- cholesterol_level(vj_CHO_TOTAL)
    vj_blood_preasure_level <-
    blood_preasure_level(vj_BP_systolic,vj_BP_diastolic)
  }
  else{
    vj_cholesterol_level <- "-"
    vj_blood_preasure_level <- "-"
  }
  # return the score of the delta function by adding the
  delta functions for cholesterol and blood preasure levels
```

```
  return(#TODO EXERCISE 4)

}
```

8. Exercise 5. Testing the algorithm with patient trajectories

8.1. Objetive

Compare three new clinical pathways with different conditions to test the algorithm implemented in previous exercises.

8.2. Notes

- Check the systaxis of a sequence with a patient trajectory in section 2
- Call the function SmithWaterman to compare two patients' trajectories by using the delta function

8.3. To Do

1. Complete the script to test your algorithm with your own realistic patients:

```
# TEST THE ALGORITHM
# TODO EXERCICE 5: prepare three new clinical pathways
with different conditions to test the algorithm

seq1 <- "80.0+110.0+70.0 80.0+110.0+75.0
80.0+110.0+70.0 80.0+110.0+70.0
80.0+110.0+70.0 80.0+110.0+70.0" #sin problemas de cardiopatia
seq2 <- "80.0+110.0+70.0 80.0+110.0+75.0
90.0+150.0+95.0 90.0+145.0+95.0 90.0+145.0+95.0
90.0+145.0+95.0" #con problemas de
sistole-diastole a partir de 4a visita
seq3 <- "80.0+110.0+70.0 80.0+110.0+75.0
80.0+110.0+75.0 90.0+150.0+95.0 90.0+145.0+95.0
90.0+145.0+95.0" #con problemas de
sistole-diastole a partir de 5a visita

print(seq1)
print(seq1)
t11sw <- SmithWaterman(seq1,seq1,delta)

print(seq1)
print(seq2)
t12sw <- SmithWaterman(seq1,seq2,delta)
```

```
print(seq1)
print(seq3)
t13sw <- SmithWaterman(seq1,seq3,delta)

print(seq2)
print(seq3)
t23sw <- SmithWaterman(seq2,seq3,delta)

#seq 4 <- TODO EXERCICE 5
#seq 5 <- TODO EXERCICE 5
#seq 6 <- TODO EXERCICE 5

# print(seq4) #TODO EXERCICE 5
# print(seq5) #TODO EXERCICE 5
# t45sw <- SmithWaterman(seq4,seq5,delta) #TODO EXERCICE 5

# print(seq4) #TODO EXERCICE 5
# print(seq6) #TODO EXERCICE 5
# t46sw <- SmithWaterman(seq4,seq6,delta) #TODO EXERCICE 5

# print(seq5) #TODO EXERCICE 5
# print(seq6) #TODO EXERCICE 5
# t56sw <- SmithWaterman(seq5,seq6,delta) #TODO EXERCICE 5
```