

R alapok, Rejtett Markov modell gyakorlat

Valószínűségi következtető és döntéstámogató rendszerek

R alapok

Az alábbiakban rövid (kivonatos) áttekintést adunk az R programnyelvbe és statisztikai szoftverbe. (Forrás: <http://www.tutorialspoint.com/r/>)

Alapvető adattípusok

Logikai érték

```
v <- TRUE
print(class(v))
## [1] "logical"

v <- F
F & F
## [1] FALSE

F | T
## [1] TRUE
```

Szám

```
v <- 23.5
print(class(v))
## [1] "numeric"

5*6
## [1] 30

5^6
## [1] 15625
```

Egész szám

```
v <- 2L
print(class(v))
## [1] "integer"
```

```
v + 2*v
## [1] 6
```

Komplex szám

```
v <- 2+5i
print(class(v))
## [1] "complex"
sqrt(-1+0i)
## [1] 0+1i
```

Karakter

```
v <- "TRUE"
print(class(v))
## [1] "character"
paste( v, ":", v )
## [1] "TRUE : TRUE"
```

Értékadás

```
# Assignment using equal operator.
var.1 = c(0,1,2,3)

# Assignment using leftward operator.
var.2 <- c("learn", "R")

# Assignment using rightward operator.
c(TRUE,1) -> var.3
```

Magasabb rendű adattípusok

Vektorok

```
# Create a vector.
apple <- c('red', 'green', "yellow")
print(apple)
## [1] "red"    "green"  "yellow"

# Get the class of the vector.
print(class(apple))
## [1] "character"
```

Vektorok létrehozása

```
# Creating a sequence from 5 to 13.
v <- 5:13
print(v)

## [1] 5 6 7 8 9 10 11 12 13

# Creating a sequence from 6.6 to 12.6.
v <- 6.6:12.6
print(v)

## [1] 6.6 7.6 8.6 9.6 10.6 11.6 12.6

# Create vector with elements from 5 to 9 incrementing by 0.4.
print(seq(5, 9, by = 0.4))

## [1] 5.0 5.4 5.8 6.2 6.6 7.0 7.4 7.8 8.2 8.6 9.0
```

Vektorok elemeinek elérése (indexelés)

```
# Accessing vector elements using position.
t <- c("Sun", "Mon", "Tue", "Wed", "Thurs", "Fri", "Sat")
u <- t[c(2,3,6)]
print(u)

## [1] "Mon" "Tue" "Fri"

# Accessing vector elements using logical indexing.
v <- t[c(TRUE,FALSE,FALSE,FALSE,FALSE,TRUE,FALSE)]
print(v)

## [1] "Sun" "Fri"

# Accessing vector elements using negative indexing.
x <- t[c(-2,-5)]
print(x)

## [1] "Sun" "Tue" "Wed" "Fri" "Sat"
```

Vektor aritmetika

```
# Create two vectors.
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11,0,8,1,2)

# Vector addition.
add.result <- v1+v2
print(add.result)

## [1] 7 19 4 13 1 13
```

```

# Vector subtraction.
sub.result <- v1-v2
print(sub.result)

## [1] -1 -3 4 -3 -1 9

# Vector multiplication.
multi.result <- v1*v2
print(multi.result)

## [1] 12 88 0 40 0 22

# Vector division.
divi.result <- v1/v2
print(divi.result)

## [1] 0.7500000 0.7272727          Inf 0.6250000 0.0000000 5.5000000

```

Listák

```

# Create a list.
list1 <- list( c(2,5,3), 21.3, sin)

# Print the list.
print(list1)

## [[1]]
## [1] 2 5 3
##
## [[2]]
## [1] 21.3
##
## [[3]]
## function (x) .Primitive("sin")

```

Listák elemeinek elnevezése

```

# Create a list containing a vector, a matrix and a list.
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
  list("green",12.3))

# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")

# Show the list.
print(list_data)

## $`1st Quarter`
## [1] "Jan" "Feb" "Mar"
##
## $A_Matrix
##      [,1] [,2] [,3]

```

```
## [1,] 3 5 -2
## [2,] 9 1 8
##
## $`A Inner list`
## $`A Inner list`[[1]]
## [1] "green"
##
## $`A Inner list`[[2]]
## [1] 12.3
```

Listák elemeinek elérése

```
# Access the first element of the list.
print(list_data[1])

## $`1st Quarter`
## [1] "Jan" "Feb" "Mar"

# Access the third element. As it is also a list, all its elements will be
printed.
print(list_data[3])

## $`A Inner list`
## $`A Inner list`[[1]]
## [1] "green"
##
## $`A Inner list`[[2]]
## [1] 12.3

# Access the list element using the name of the element.
print(list_data$A_Matrix)

##      [,1] [,2] [,3]
## [1,] 3 5 -2
## [2,] 9 1 8
```

Listák összefűzése

```
# Create two lists.
list1 <- list(1,2,3)
list2 <- list("Sun", "Mon", "Tue")

# Merge the two lists.
merged.list <- c(list1, list2)

# Print the merged list.
print(merged.list)

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

```
## [1] 2
##
## [[3]]
## [1] 3
##
## [[4]]
## [1] "Sun"
##
## [[5]]
## [1] "Mon"
##
## [[6]]
## [1] "Tue"
```

Mátrixok

```
# Create a matrix.
M = matrix(c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
print(M)

##      [,1] [,2] [,3]
## [1,] "a"  "a"  "b"
## [2,] "c"  "b"  "a"
```

Mátrixok létrehozása (byrow: T vs. F)

```
# Elements are arranged sequentially by row.
M <- matrix(c(3:14), nrow = 4, byrow = TRUE)
print(M)

##      [,1] [,2] [,3]
## [1,]    3    4    5
## [2,]    6    7    8
## [3,]    9   10   11
## [4,]   12   13   14

# Elements are arranged sequentially by column.
N <- matrix(c(3:14), nrow = 4, byrow = FALSE)
print(N)

##      [,1] [,2] [,3]
## [1,]    3    7   11
## [2,]    4    8   12
## [3,]    5    9   13
## [4,]    6   10   14

# Define the column and row names.
rownames = c("row1", "row2", "row3", "row4")
colnames = c("col1", "col2", "col3")

P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames,
```

```
colnames))
print(P)

##      col1 col2 col3
## row1     3     4     5
## row2     6     7     8
## row3     9    10    11
## row4    12    13    14
```

Mátrixok elemeinek elérése (indexelés)

```
# Access the element at 3rd column and 1st row.
print(P[1,3])

## [1] 5

# Access the element at 2nd column and 4th row.
print(P[4,2])

## [1] 13

# Access only the 2nd row.
print(P[2,])

## col1 col2 col3
##     6     7     8

# Access only the 3rd column.
print(P[,3])

## row1 row2 row3 row4
##     5     8    11    14
```

Mátrix-műveletek

```
# Create two 2x3 matrices.
matrix1 <- matrix(c(3, 9, -1, 4, 2, 6), nrow = 2)
print(matrix1)

##      [,1] [,2] [,3]
## [1,]    3  -1    2
## [2,]    9   4    6

matrix2 <- matrix(c(5, 2, 0, 9, 3, 4), nrow = 2)
print(matrix2)

##      [,1] [,2] [,3]
## [1,]    5   0    3
## [2,]    2   9    4

# Transpose of matrix
result <- t(matrix1)
print(result)
```

```

##      [,1] [,2]
## [1,]   3   9
## [2,]  -1   4
## [3,]   2   6

# Add the matrices.
result <- matrix1 + matrix2
print(result)

##      [,1] [,2] [,3]
## [1,]   8  -1   5
## [2,]  11  13  10

# Subtract the matrices
result <- matrix1 - matrix2
print(result)

##      [,1] [,2] [,3]
## [1,]  -2  -1  -1
## [2,]   7  -5   2

# Multiply the matrices.
result <- matrix1 * matrix2
print(result)

##      [,1] [,2] [,3]
## [1,]  15   0   6
## [2,]  18  36  24

# Divide the matrices
result <- matrix1 / matrix2
print(result)

##      [,1]      [,2]      [,3]
## [1,]  0.6      -Inf  0.6666667
## [2,]  4.5  0.4444444  1.5000000

# Multiply the matrices by matrix multiplication
result <- matrix1 %*% t(matrix2)
print(result)

##      [,1] [,2]
## [1,]  21   5
## [2,]  63  78

result <- t(matrix1) %*% matrix2
print(result)

##      [,1] [,2] [,3]
## [1,]  33  81  45
## [2,]   3  36  13
## [3,]  22  54  30

```


Faktorok

```
# Create a vector.
apple_colors <- c('green','green','yellow','red','red','red','green')

# Create a factor object.
factor_apple <- factor(apple_colors)

# Print the factor.
print(factor_apple)

## [1] green green yellow red red red green
## Levels: green red yellow

print(nlevels(factor_apple))

## [1] 3
```

Adat táblák

```
# Create the data frame.
emp.data <- data.frame(
  emp_id = c(1:5),
  emp_name = c("Rick", "Dan", "Michelle", "Ryan", "Gary"),
  salary = c(623.3, 515.2, 611.0, 729.0, 843.25),

  start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",
    "2015-03-27")),
  stringsAsFactors = FALSE
)

# Print the data frame.
print(emp.data)

##   emp_id emp_name salary start_date
## 1     1     Rick 623.30 2012-01-01
## 2     2      Dan 515.20 2013-09-23
## 3     3 Michelle 611.00 2014-11-15
## 4     4     Ryan 729.00 2014-05-11
## 5     5     Gary 843.25 2015-03-27
```

Adat tábla elemeinek elérése (indexelés)

```
# Extract Specific columns.
result <- data.frame(emp.data$emp_name, emp.data$salary)
print(result)

##   emp.data.emp_name emp.data.salary
## 1             Rick           623.30
## 2              Dan           515.20
## 3          Michelle           611.00
```

```
## 4          Ryan          729.00
## 5          Gary          843.25

# Extract first two rows.
result <- emp.data[1:2,]
print(result)

##   emp_id emp_name salary start_date
## 1      1    Rick  623.3 2012-01-01
## 2      2     Dan  515.2 2013-09-23

# Extract 3rd and 5th row with 2nd and 4th column.
result <- emp.data[c(3,5),c(2,4)]
print(result)

##   emp_name start_date
## 3 Michelle 2014-11-15
## 5      Gary 2015-03-27
```

Új oszlop hozzáadása

```
# Add the "dept" column.
emp.data$dept <- c("IT","Operations","IT","HR","Finance")
v <- emp.data
print(v)

##   emp_id emp_name salary start_date      dept
## 1      1    Rick  623.30 2012-01-01        IT
## 2      2     Dan  515.20 2013-09-23 Operations
## 3      3 Michelle 611.00 2014-11-15        IT
## 4      4    Ryan  729.00 2014-05-11        HR
## 5      5    Gary  843.25 2015-03-27      Finance
```

Új sorok hozzáadása

```
# Create the second data frame
emp.newdata <- data.frame(
  emp_id = c(6:8),
  emp_name = c("Rasmi","Pranab","Tusar"),
  salary = c(578.0,722.5,632.8),
  start_date = as.Date(c("2013-05-21","2013-07-30","2014-06-17")),
  dept = c("IT","Operations","Finance"),
  stringsAsFactors = FALSE
)

# Bind the two data frames.
emp.finaldata <- rbind(emp.data,emp.newdata)
print(emp.finaldata)

##   emp_id emp_name salary start_date      dept
## 1      1    Rick  623.30 2012-01-01        IT
## 2      2     Dan  515.20 2013-09-23 Operations
```

```
## 3      3 Michelle 611.00 2014-11-15      IT
## 4      4      Ryan 729.00 2014-05-11      HR
## 5      5      Gary 843.25 2015-03-27    Finance
## 6      6      Rasmi 578.00 2013-05-21      IT
## 7      7      Pranab 722.50 2013-07-30 Operations
## 8      8      Tusar 632.80 2014-06-17    Fianance
```

Függvények

```
# define a simple function
myFirstFun <- function( n )
{
  n*n # compute the square of integer n
  # or return( n*n )
}
# define a value
k <- 10
# call the function with that value
myFirstFun( k )
## [1] 100
```

Vezérlési szerkezetek

```
# For Loop, if
m=20;
for (k in 1:m)
{
  if (!k %% 2)
    next
  print(k)
}

## [1] 1
## [1] 3
## [1] 5
## [1] 7
## [1] 9
## [1] 11
## [1] 13
## [1] 15
## [1] 17
## [1] 19

# next, break
# while loop
response <- sample(40:45, size=1)
while( response != 42 )
{
  print( "Sorry, the answer to whatever the question MUST be 42" )
}
```

```
response <- sample(40:45, size=1)
}
## [1] "Sorry, the answer to whatever the question MUST be 42"
## [1] "Sorry, the answer to whatever the question MUST be 42"
## [1] "Sorry, the answer to whatever the question MUST be 42"
## [1] "Sorry, the answer to whatever the question MUST be 42"
## [1] "Sorry, the answer to whatever the question MUST be 42"
# if() { } else if() {} else {}
```

Csomagok

Csomagok letöltése

```
#install.packages("HMM")
#install.packages("ggplot2")
```

Csomagok betöltése

```
library(HMM)
library(ggplot2)
```

Rejtett Markov modell - példa

Avagy "Hogyan tudjuk kiszűrni a hamis pénzérmét a feldobások eredménye alapján?".

Tegyük fel, hogy egy kaszinóban "fej vagy írás" játékot játszunk. A krupié azonban nem becsületes; időnként, amikor nem vesszük észre, a valódi pénzérmét egy cinkeltre cseréli. A valódi pénzérme feldobása esetén fele-fele arányban kapunk fejet vagy írást, ám a cinkelt pénzérme feldobása után 0.8 valószínűséggel lesz fej az eredmény. A krupié a valódi pénzérmét 0.3 valószínűséggel cseréli le minden dobás után a cinkeltre, míg a cinkeltet 0.2 valószínűséggel cseréli vissza valódira. A játékot egyébként nagy (0.9) valószínűséggel a valódi pénzérmével kezdjük.

Modellezzük a játékot egy rejtett Markov modell segítségével!

```
library(HMM)
```

```
hmm <- initHMM( States = c( "Valodi", "Nem valódi" ),  
               Symbols = c( "Fej", "Írás" ),  
               startProbs = c( 0.9, 0.1 ),  
               transProbs = matrix( c( 0.7, 0.3,  
                                       0.2, 0.8 ), ncol = 2, byrow = T ),  
               emissionProbs = matrix( c( 0.5, 0.5,  
                                          0.8, 0.2 ), ncol = 2, byrow = T )  
)
```

Nyomtassuk ki a modell leírását a képernyőre!

```
print(hmm)
```

```
## $States  
## [1] "Valodi"      "Nem valódi"  
##  
## $Symbols  
## [1] "Fej"  "Írás"  
##  
## $startProbs  
##      Valodi Nem valódi  
##      0.9      0.1  
##  
## $transProbs  
##           to  
## from      Valodi Nem valódi  
## Valodi      0.7      0.3  
## Nem valódi  0.2      0.8  
##  
## $emissionProbs  
##           symbols  
## states    Fej  Írás  
## Valodi    0.5  0.5  
## Nem valódi 0.8  0.2
```

Szimuláljunk egy 100 hosszú dobássorozatot, majd írjuk ki az eredményeket!

```
nSim <- 100

simulation <- simHMM( hmm, nSim )

simulation

## $states
## [1] "Valodi"      "Valodi"      "Valodi"      "Valodi"      "Valodi"
## [6] "Valodi"      "Valodi"      "Valodi"      "Nem valódi"  "Valodi"
## [11] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Valodi"
## [16] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Valodi"
## [21] "Valodi"      "Nem valódi"  "Nem valódi"  "Valodi"      "Valodi"
## [26] "Valodi"      "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
## [31] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Valodi"      "Valodi"
## [36] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
## [41] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
## [46] "Nem valódi"  "Nem valódi"  "Valodi"      "Valodi"      "Valodi"
## [51] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
## [56] "Valodi"      "Valodi"      "Valodi"      "Nem valódi"  "Nem valódi"
## [61] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
## [66] "Nem valódi"  "Valodi"      "Nem valódi"  "Nem valódi"  "Nem valódi"
## [71] "Nem valódi"  "Valodi"      "Valodi"      "Nem valódi"  "Nem valódi"
## [76] "Nem valódi"  "Nem valódi"  "Valodi"      "Valodi"      "Valodi"
## [81] "Valodi"      "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
## [86] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
## [91] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
## [96] "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"  "Nem valódi"
##
## $observation
## [1] "Fej" "Fej" "Iras" "Iras" "Iras" "Iras" "Fej" "Iras" "Iras" "Fej"
## [11] "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Iras"
## [21] "Fej" "Fej" "Fej" "Iras" "Iras" "Iras" "Fej" "Fej" "Fej" "Fej"
## [31] "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Iras" "Fej" "Fej"
## [41] "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Iras" "Iras" "Fej"
## [51] "Iras" "Fej" "Fej" "Fej" "Fej" "Fej" "Iras" "Fej" "Fej" "Fej"
## [61] "Fej" "Fej" "Fej" "Fej" "Fej" "Iras" "Iras" "Iras" "Fej" "Fej"
## [71] "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej"
## [81] "Iras" "Iras" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej"
## [91] "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Fej" "Iras" "Fej" "Fej"
```

Hogyan működik a simHMM metódus?

```
simHMM

## function (hmm, length)
## {
##     hmm$transProbs[is.na(hmm$transProbs)] = 0
##     hmm$emissionProbs[is.na(hmm$emissionProbs)] = 0
##     states = c()
```

```

##      emission = c()
##      states = c(states, sample(hmm$States, 1, prob = hmm$startProbs))
##      for (i in 2:length) {
##          state = sample(hmm$States, 1, prob = hmm$transProbs[states[i -
##              1], ])
##          states = c(states, state)
##      }
##      for (i in 1:length) {
##          emi = sample(hmm$Symbols, 1, prob = hmm$emissionProbs[states[i],
##              ])
##          emission = c(emission, emi)
##      }
##      return(list(states = states, observation = emission))
##  }

```

Keressük meg a legvalószínűbb útvonalat (dobássorozatot) a Viterbi algoritmus segítségével, majd írjuk ki az eredményeket!

```

vit = viterbi( hmm, simulation$observation )
vit
##      [1] "Valodi"      "Valodi"      "Valodi"      "Valodi"      "Valodi"
##      [6] "Valodi"      "Valodi"      "Valodi"      "Valodi"      "Nem valódi"
##     [11] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [16] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [21] "Nem valódi" "Nem valódi" "Nem valódi" "Valodi"      "Valodi"
##     [26] "Valodi"      "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [31] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [36] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [41] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [46] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [51] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [56] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [61] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [66] "Valodi"      "Valodi"      "Valodi"      "Nem valódi" "Nem valódi"
##     [71] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [76] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [81] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [86] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [91] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"
##     [96] "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi" "Nem valódi"

```

Számítsuk ki az egyes állapotok poszterior valószínűségét minden időpillanatban! Emlékeztető: simítás (forward-backward algoritmus).

```

f = forward( hmm, simulation$observation )
b = backward( hmm, simulation$observation )

logProbObservations = log( sum( exp( f[,nSim] ) ) )
posterior = exp( ( f + b ) - logProbObservations )

```

Ábrázoljuk és értelmezzük az eredményeket! Ehhez először töltsük be a ggplot2 csomagot, majd készítsünk egy data.frame-et, amely a dobás indexe mellett tartalmazza a szimulált állapotokat és szimulált megfigyeléseket, a Viterbi algoritmussal meghatározott legvalószínűbb útvonalat, illetve a Valódi pénzérme állapot poszterior valószínűségét.

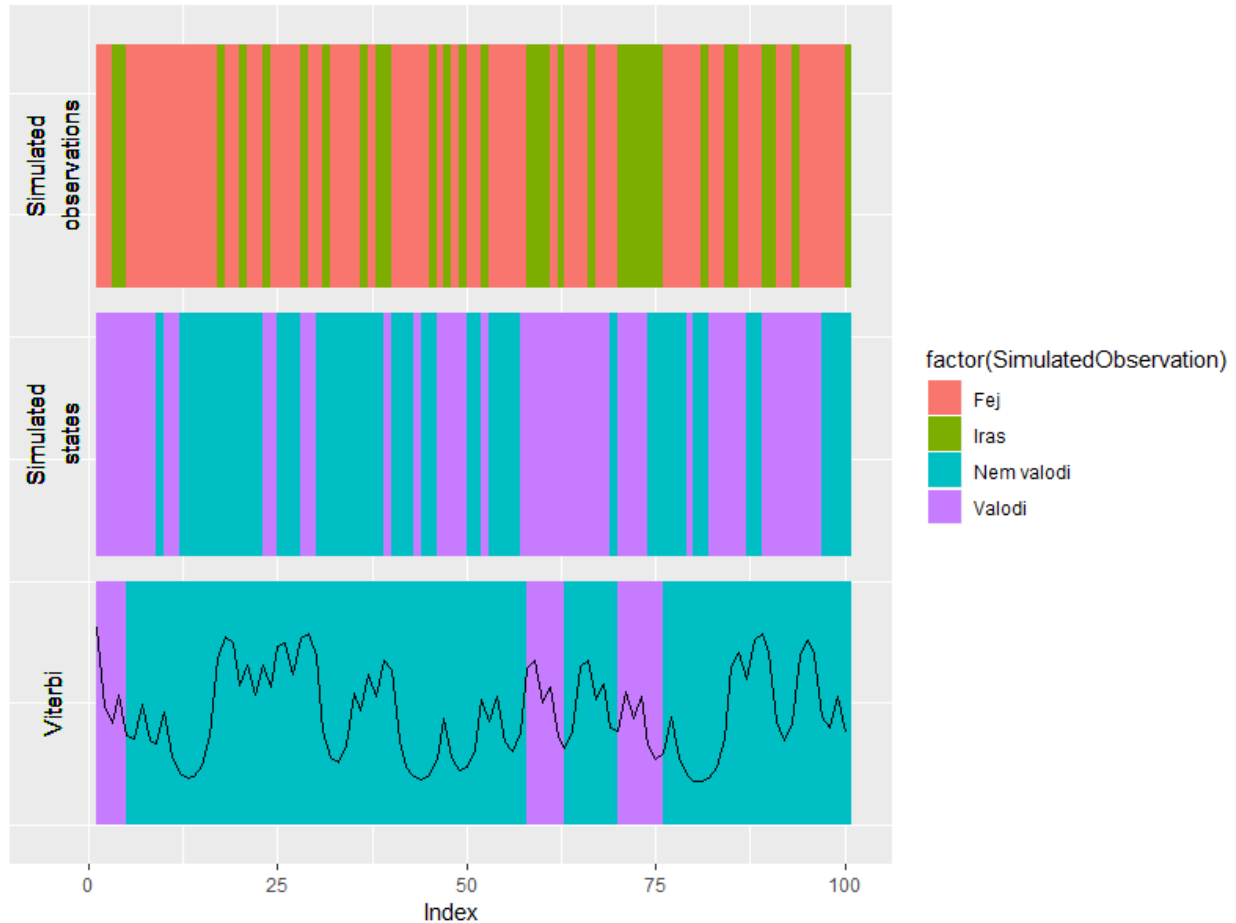
```
library(ggplot2)

df <- data.frame( Index = 1:nSim,
                  SimulatedStates = simulation$states,
                  SimulatedObservation = simulation$observation,
                  ViterbiStates = vit,
                  PosteriorOfHonest = posterior[1,] )

head(df)

##   Index SimulatedStates SimulatedObservation ViterbiStates
## 1     1           Valodi                   Fej           Valodi
## 2     2           Valodi                   Fej           Valodi
## 3     3           Valodi                   Iras           Valodi
## 4     4           Valodi                   Iras           Valodi
## 5     5           Valodi                   Iras           Valodi
## 6     6           Valodi                   Iras           Valodi
##   PosteriorOfHonest
## 1           0.8670509
## 2           0.6903766
## 3           0.8133898
## 4           0.8499013
## 5           0.8437979
## 6           0.7877434

ggplot( df ) +
  geom_rect( aes( xmin=Index, xmax=Index+1, ymin=2.2, ymax=3.2,
                 fill=factor(SimulatedObservation) ) ) +
  geom_rect( aes( xmin=Index, xmax=Index+1, ymin=1.1, ymax=2.1,
                 fill=factor(SimulatedStates) ) ) +
  geom_rect( aes( xmin=Index, xmax=Index+1, ymin=0, ymax=1,
                 fill=factor(ViterbiStates) ) ) +
  geom_line( aes( x=Index, y=PosteriorOfHonest ) ) +
  geom_text( aes( label = "Viterbi", x=-5, y=0.5, angle=90) ) +
  geom_text( aes( label = "Simulated\nstates", x=-5, y=1.6, angle=90) ) +
  geom_text( aes( label = "Simulated\nobservations", x=-5, y=2.7, angle=90) ) +
  labs( y="" ) +
  theme( axis.text.y = element_blank(), axis.ticks.y = element_blank(),
         axis.title.y = element_blank() )
```

Feladatok:

1. Végezzük el a szimulációt hosszabb dobássorozatokra is.
2. Vizsgáljuk meg a modell paramétereinek hatását az eredményekre (állapotátmenet-valószínűségek, kibocsátási valószínűség-eloszlás, prior állapot-valószínűségek).
3. Egészítsük ki az ábrát, hogy ne csak a Valódi pénzérme állapot poszterior valószínűségét, hanem az adott időpontban a legvalószínűbb állapotot is megjelenítse! Figyeljük meg, hogy ez hogyan különbözik a Viterbi útvonaltól, illetve a szimulált állapotoktól. Melyik közelíti jobban a valóságot? Számszerűsítsük a különbségeket (viterbi vs. szimulált) \Leftrightarrow (legvalószínűbb állapotok sorozata vs. szimulált)!
4. Módosítsuk a modellt, hogy három pénzérme feldobását modellezze, amelyből kettő hamis (az egyik gyakrabban eredményez fejet, a másik pedig írást). Módosítsuk az ábrát a modellnek megfelelően.
5. (* Készíts egy rejtett Markov modellt egy általad választott problémára. Vizsgáld meg és demonstráld a modell működését.)