R ladies hands-on HMM - answers

Emmeke Aarts

Exercise 1

In this first exercise we will simulate data from a hidden Markov model, and subsequently analyse it. We will assume that the observations are generated by a normal distribution, for which the parameters depend on which state is active. Note that the used code can easily be adapted to generate and analyse data with more than 2 states, or another conditional distribution, like Poisson or categorical.

a)

Simulate a 2-state normal HMM with 100 observations using the code below. Start with a model that is easy to estimate for the HMM algorithm: conditional distributions that do not or only partly overlap and high self-transitions. For example,

- mu equals 3 and 10
- sd equals 1.5 and 3
- self transition probabilities of .8 for both states (and hence, a probability of .2 to switch to the other state)

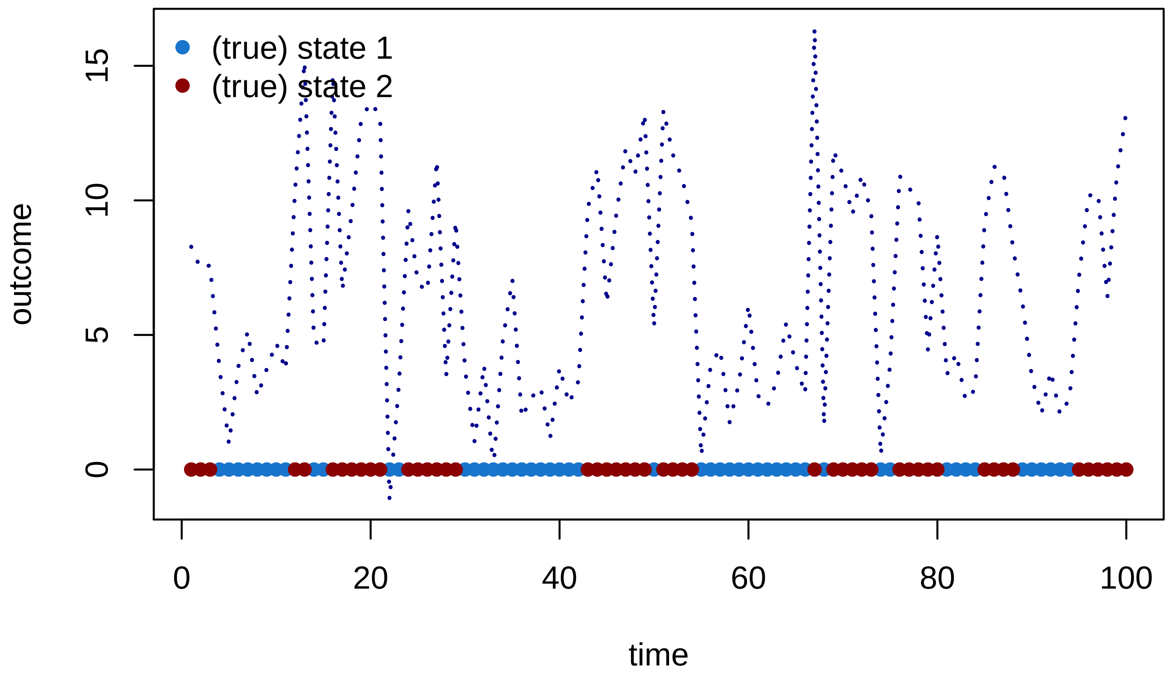
```r
norm.HMM.Generate <- function(n, m, mu, sd, gamma1, delta=NULL){
  if(m != length(gamma1[,1]) | m != length(gamma1[1,])){
   stop("The number of states given by m does not match the number of rows
        and/or colums of the transition probability matrix gamma")
  }
  state <- numeric(n)
      # first attribute the first observation in the sequence
      if (is.null(delta)) {
        state[1] <- sample(1:m, 1, prob = solve(t(diag(m) - gamma1 + 1), rep(1, m)))
      } else {
            state[1] <- sample(1:m, 1, prob = delta)
      }
    # conditional on the first observation, sample the following observations
      for (i in 2:n){
        state[i] <- sample(1:m, 1, prob = gamma1[state[i-1],])
      }
      x <- rnorm(n, mean = mu[state], sd = sd[state])
      return(list(state = state, seq.x = x))
    }

set.seed(23142)
sample.data <- norm.HMM.Generate(n = 100, m = 2, mu = c(3, 10), sd = c(1.5, 3),
               gamma1 =   matrix(c(.8, .2, .2, .8), byrow = T, ncol = 2))
```

b)

Plot the data over time. A line plot gives the most intuitive visualization of the outcome here.

```r
plot(sample.data$seq.x, type = "l", xlab = "time", ylab = "outcome", lwd = 2,
     lty = 3, col = "darkblue")
points(x = c(1:100)[sample.data$state == 1], y = rep(0, sum(sample.data$state == 1)),
       col = "dodgerblue3", pch = 16)
points(x = c(1:100)[sample.data$state == 2], y = rep(0, sum(sample.data$state == 2)),
       col = "darkred", pch = 16)
legend("topleft", pch = 16, col = c("dodgerblue3", "darkred"),
       legend = c("(true) state 1", "(true) state 2"), bty = "n")
```



c)

Analyse the simulated data applying the HMM algorithm from the `depmixS4` package.

```
library(depmixS4)
```

```
## Loading required package: nnet

## Loading required package: MASS

## Loading required package: Rsolnp
```

```
# ?depmix
model1 <- depmix(sample.data$seq.x ~ 1, nstates = 2, family = gaussian(), ntimes = c(100))
out.model1 <- fit(model1)
```

```
## iteration 0 logLik: -282.5615
## iteration 5 logLik: -282.517
## iteration 10 logLik: -282.4595
## iteration 15 logLik: -282.1865
## iteration 20 logLik: -277.9548
## iteration 25 logLik: -257.3868
## iteration 30 logLik: -257.2573
## iteration 35 logLik: -257.2558
## converged at iteration 39 with logLik: -257.2558
```

d)

Inspect conditional distributions and compare to true values.

```r
summary(out.model1)
```

```
## Initial state probabilties model
## pr1 pr2
##   1   0
##
## Transition matrix
##         toS1  toS2
## fromS1 0.779 0.221
## fromS2 0.222 0.778
##
## Response parameters
## Resp 1 : gaussian
##     Re1.(Intercept) Re1.sd
## St1          10.005  2.691
## St2           3.129  1.546
```

e)

Inspect the transition matrix and compare to true values

```r
# see output under 1d)
```

f)

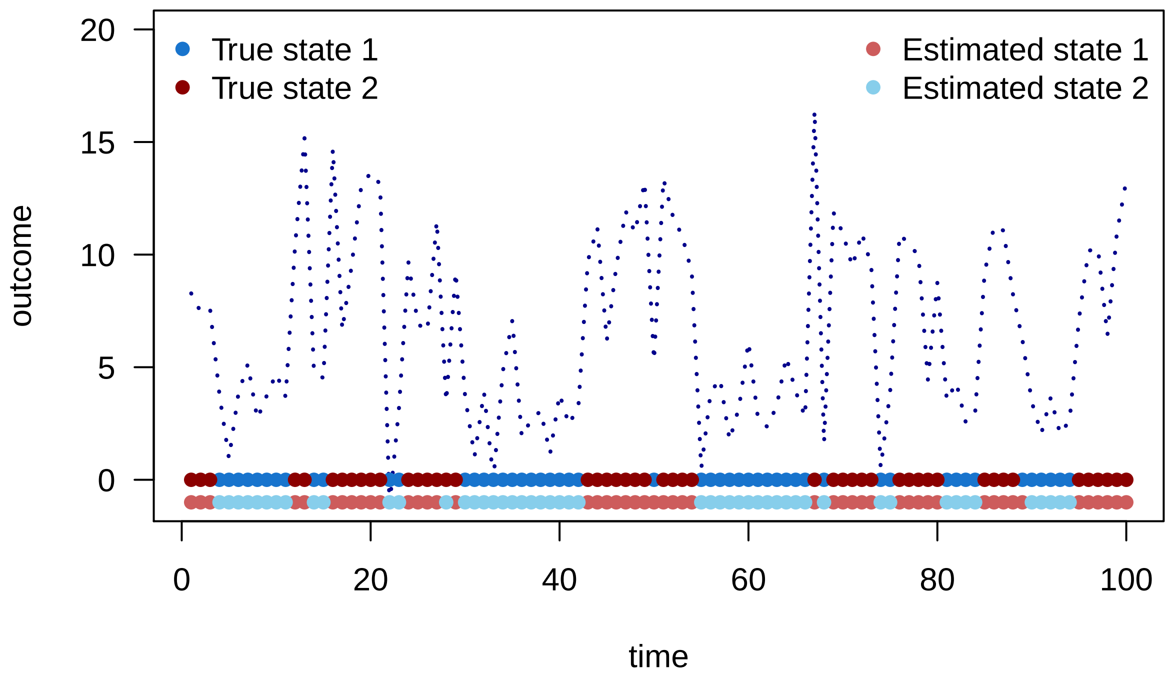
Use the Viterbi algorithm from the `depmixS4` package to recover the most likely state sequence

```r
# ?posterior
viterbi.states <- posterior(out.model1)[,1]
```

g)

Add the estimated and the true states to the plot that depicts the observations over time (that you made in 1a)

```r
plot(sample.data$seq.x, type = "l", xlab = "time", ylab = "outcome", lwd = 2,
     lty = 3, col = "darkblue", ylim = c(-1, 20), las = 1)
points(x = c(1:100)[sample.data$state == 1],
       y = rep(0, sum(sample.data$state == 1)), col = "dodgerblue3", pch = 16)
points(x = c(1:100)[sample.data$state == 2],
       y = rep(0, sum(sample.data$state == 2)), col = "darkred", pch = 16)
legend("topleft", pch = 16, col = c("dodgerblue3", "darkred"),
       legend = c("True state 1", "True state 2"), bty = "n")
points(x = c(1:100)[viterbi.states == 1],
       y = rep(-1, sum(viterbi.states == 1)), col = "indianred", pch = 16)
points(x = c(1:100)[viterbi.states == 2],
       y = rep(-1, sum(viterbi.states == 2)), col = "skyblue", pch = 16)
legend("topright", pch = 16, col = c("indianred", "skyblue"),
       legend = c("Estimated state 1", "Estimated state 2"), bty = "n")
```



h)

Compare the plotted estimated and true states to each other, and to the simulated observations over time

```
# we see that the estimated and true states are almost identical
```

i)

Play around with simulating the data and running the HMM: does it recover the hidden states correctly each time? How about if you make the simulated data sequence longer or shorter? And if you change the conditional distribution or the transition matrix?

```
# something you can do for yourselves
```

Exercise 2

In this exercise, we will use example data from the book *Hidden Markov Models for Time Series - An Introduction Using R* (2009) by W. Zucchini and I.L. MacDonald. The data concerns the number of major earthquakes in the world from 1900-2006, and is assumed to be Poisson distributed.

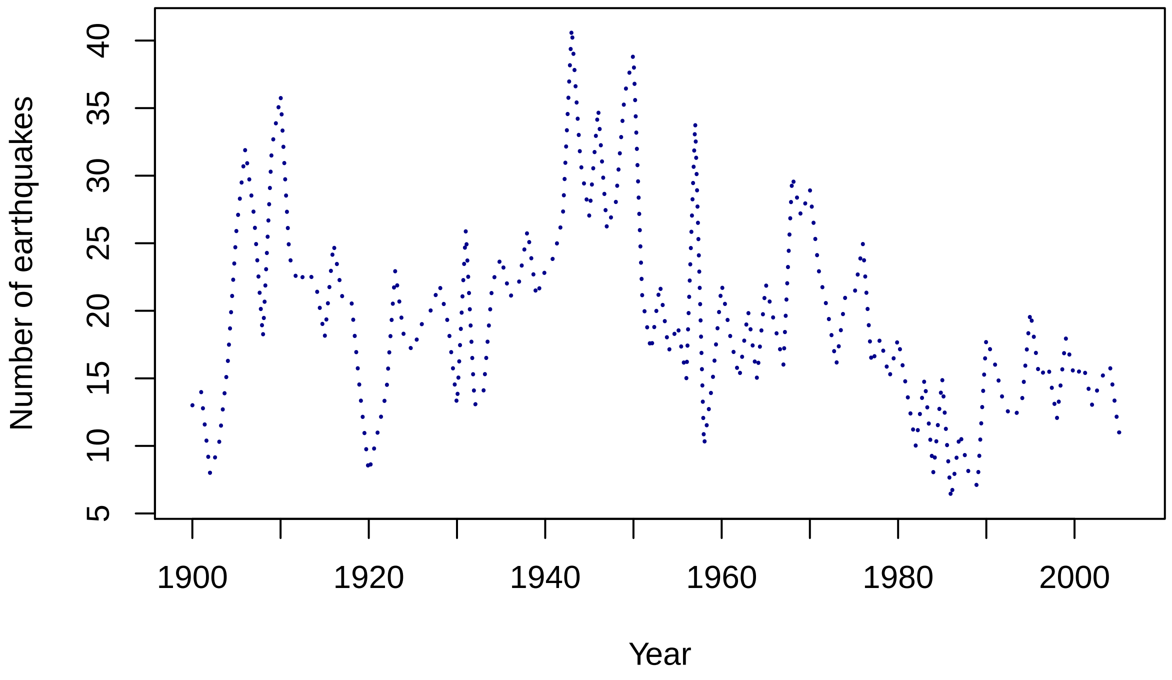
a)

The data is as follows:

```
earthq.data <- c(13, 14, 8, 10, 16, 26, 32, 27, 18, 32, 36, 24, 22, 23, 22, 18, 25,
                 21, 21, 14, 8, 11, 14, 23, 18, 17, 19, 20, 22, 19, 13, 26, 13, 14,
                 22, 24, 21, 22, 26, 21, 23, 24, 27, 41, 31, 27, 35, 26, 28, 36,
                 39, 21, 17, 22, 17, 19, 15, 34, 10, 15, 22, 18, 15, 20, 15, 22,
                 19, 16, 30, 27, 29, 23, 20, 16, 21, 21, 25, 16, 18, 15, 18, 14,
                 10, 15, 8, 15, 6, 11, 8, 7, 18, 16, 13, 12, 13, 20, 15, 16, 12,
                 18, 15, 16, 13, 15, 16, 11, 11)
```

Plot the data over time.

```r
plot(earthq.data, type = "l", xlab = "Year", ylab = "Number of earthquakes",
     lwd = 2, lty = 3, col = "darkblue", xaxt = 'n')
axis(1, at = seq(1,105,10), labels = seq(1900,2005,10))
```



b)

Fit a 2 state, 3 state and 4 state HMM using the `depmixS4` package. Note that depmix returns the intercept for the Poisson distribution instead of lambda. To get lambda, take the exponent of the value under intercept.

```r
set.seed(234137)
library(depmixS4)
# 1 state model
earth1.m1 <- depmix(earthq.data ~ 1, nstates = 1, family = poisson(),
                    ntimes = length(earthq.data))
out.earth1 <- fit(earth1.m1)
```

```
## iteration 0 logLik: -391.9189
## converged at iteration 1 with logLik: -391.9189
```

```r
# 2 state model
earth1.m2 <- depmix(earthq.data ~ 1, nstates = 2, family = poisson(),
                    ntimes = length(earthq.data))
out.earth2 <- fit(earth1.m2)
```

```
## iteration 0 logLik: -380.9367
## iteration 5 logLik: -342.3084
## iteration 10 logLik: -341.8944
## iteration 15 logLik: -341.8799
## iteration 20 logLik: -341.8788
## iteration 25 logLik: -341.8787
## converged at iteration 26 with logLik: -341.8787
```

```r
# 3 state model
earth1.m3 <- depmix(earthq.data ~ 1, nstates = 3, family = poisson(),
                    ntimes = length(earthq.data))
out.earth3 <- fit(earth1.m3)
```

```
## iteration 0 logLik: -364.2395
## iteration 5 logLik: -336.4032
## iteration 10 logLik: -328.9572
## iteration 15 logLik: -328.5311
## iteration 20 logLik: -328.5275
## converged at iteration 23 with logLik: -328.5275
```

```
# 4 state model
earth1.m4 <- depmix(earthq.data ~ 1, nstates = 4, family = poisson(),
                    ntimes = length(earthq.data))
out.earth4 <- fit(earth1.m4)
```

```
## iteration 0 logLik: -383.4405
## iteration 5 logLik: -334.4925
## iteration 10 logLik: -328.7399
## iteration 15 logLik: -328.3981
## iteration 20 logLik: -328.3478
## iteration 25 logLik: -328.3118
## iteration 30 logLik: -328.2851
## iteration 35 logLik: -328.2645
## iteration 40 logLik: -328.247
## iteration 45 logLik: -328.2311
## iteration 50 logLik: -328.2161
## iteration 55 logLik: -328.2021
## iteration 60 logLik: -328.1893
## iteration 65 logLik: -328.1779
## iteration 70 logLik: -328.1676
## iteration 75 logLik: -328.1583
## iteration 80 logLik: -328.1496
## iteration 85 logLik: -328.1412
## iteration 90 logLik: -328.1326
## iteration 95 logLik: -328.1235
## iteration 100 logLik: -328.1136
## iteration 105 logLik: -328.1026
## iteration 110 logLik: -328.0899
## iteration 115 logLik: -328.075
## iteration 120 logLik: -328.0573
## iteration 125 logLik: -328.0359
## iteration 130 logLik: -328.0095
## iteration 135 logLik: -327.9765
## iteration 140 logLik: -327.9344
## iteration 145 logLik: -327.88
## iteration 150 logLik: -327.8111
## iteration 155 logLik: -327.7332
## iteration 160 logLik: -327.665
## iteration 165 logLik: -327.6204
## iteration 170 logLik: -327.5963
## iteration 175 logLik: -327.5837
## iteration 180 logLik: -327.5769
## iteration 185 logLik: -327.5728
## iteration 190 logLik: -327.5703
## iteration 195 logLik: -327.5687
## iteration 200 logLik: -327.5676
## iteration 205 logLik: -327.5668
## iteration 210 logLik: -327.5663
## iteration 215 logLik: -327.5659
## iteration 220 logLik: -327.5657
## iteration 225 logLik: -327.5655
## iteration 230 logLik: -327.5654
## iteration 235 logLik: -327.5653
## iteration 240 logLik: -327.5653
## iteration 245 logLik: -327.5652
## iteration 250 logLik: -327.5652
## iteration 255 logLik: -327.5652
## converged at iteration 256 with logLik: -327.5652
```

c)

Compare the different models on their conditional distributions

```
# results 1 state model
summary(out.earth1)
```

```
## Initial state probabilties model
## pr1
##   1
##
## Transition matrix
##        toS1
## fromS1    1
##
## Response parameters
## Resp 1 : poisson
##     Re1.(Intercept)
## St1           2.963
```

```
round(exp(2.963),2) # lambda state 1
```

```
## [1] 19.36
```

```
summary(out.earth2)
```

```
## Initial state probabilties model
## pr1 pr2
##   1   0
##
## Transition matrix
##         toS1  toS2
## fromS1 0.928 0.072
## fromS2 0.119 0.881
##
## Response parameters
## Resp 1 : poisson
##     Re1.(Intercept)
## St1           2.736
## St2           3.259
```

```
round(c(exp(2.736), exp(3.259)),2) # lambda state 1 and 2
```

```
## [1] 15.43 26.02
```

```r
summary(out.earth3)
```

```
## Initial state probabilties model
## pr1 pr2 pr3
##   0   1   0
##
## Transition matrix
##         toS1  toS2  toS3
## fromS1 0.810 0.000 0.190
## fromS2 0.029 0.939 0.032
## fromS3 0.053 0.040 0.906
##
## Response parameters
## Resp 1 : poisson
##     Re1.(Intercept)
## St1           3.392
## St2           2.575
## St3           2.981
```

```r
round(c(exp(3.392), exp(2.575), exp(2.981)), 2) # lambda state 1, 2, and 3
```

```
## [1] 29.73 13.13 19.71
```

```r
summary(out.earth4)
```

```
## Initial state probabilties model
## pr1 pr2 pr3 pr4
##   0   0   0   1
##
## Transition matrix
##         toS1  toS2  toS3  toS4
## fromS1 0.000 0.594 0.406 0.000
## fromS2 0.100 0.856 0.000 0.044
## fromS3 0.310 0.000 0.690 0.000
## fromS4 0.059 0.000 0.000 0.941
##
## Response parameters
## Resp 1 : poisson
##     Re1.(Intercept)
## St1           3.173
## St2           2.966
## St3           3.432
## St4           2.574
```

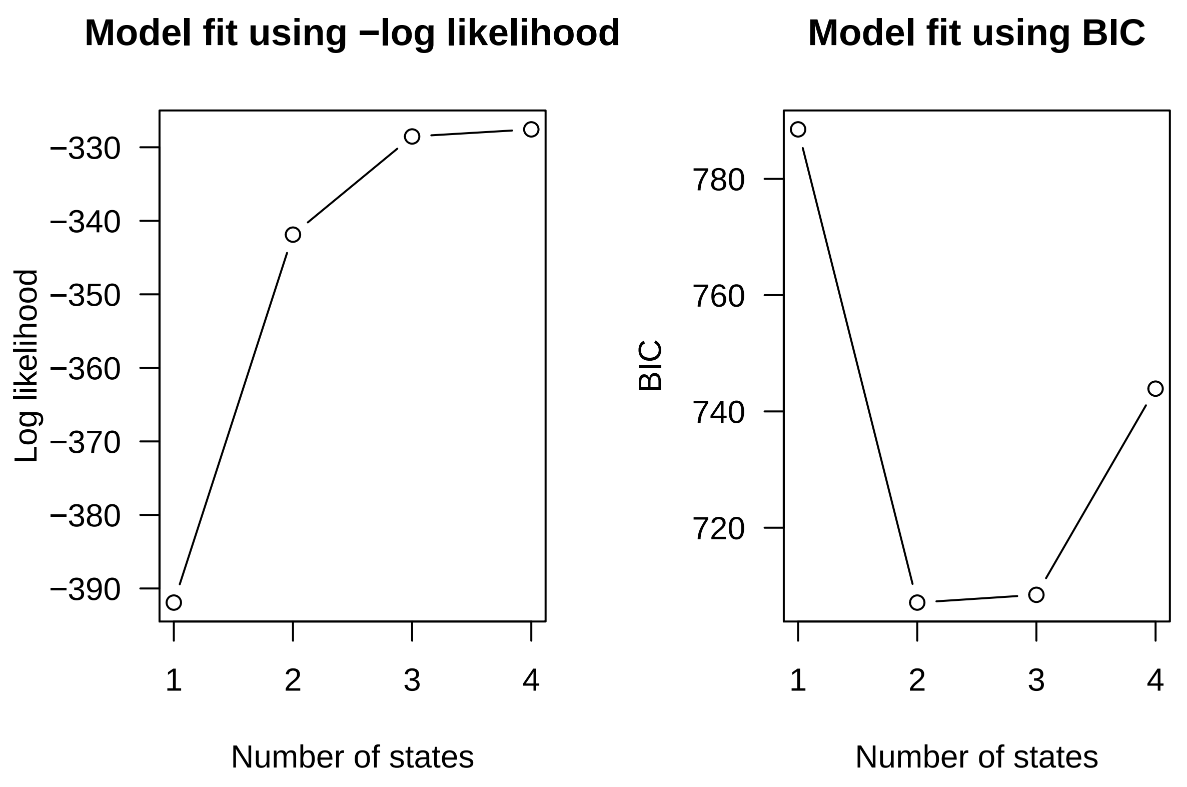
```r
round(c(exp(3.390), exp(2.420), exp(2.980), exp(2.625)),2) # lambda state 1, 2, 3, and 4
```

```
## [1] 29.67 11.25 19.69 13.80
```

d)

Which model best fits the data?

```r
par(mfrow = c(1,2))
plot(c(logLik(out.earth1), logLik(out.earth2), logLik(out.earth3), logLik(out.earth4)),
     type = "b", ylab = "Log likelihood", xlab = "Number of states",
     main = "Model fit using -log likelihood", xaxt = 'n', las = 1)
axis(1, at = c(1:4))
plot(c(BIC(out.earth1), BIC(out.earth2), BIC(out.earth3), BIC(out.earth4)),
     type = "b", ylab = "BIC", xlab = "Number of states",
     main = "Model fit using BIC", xaxt = 'n', las = 1)
axis(1, at = c(1:4))
```

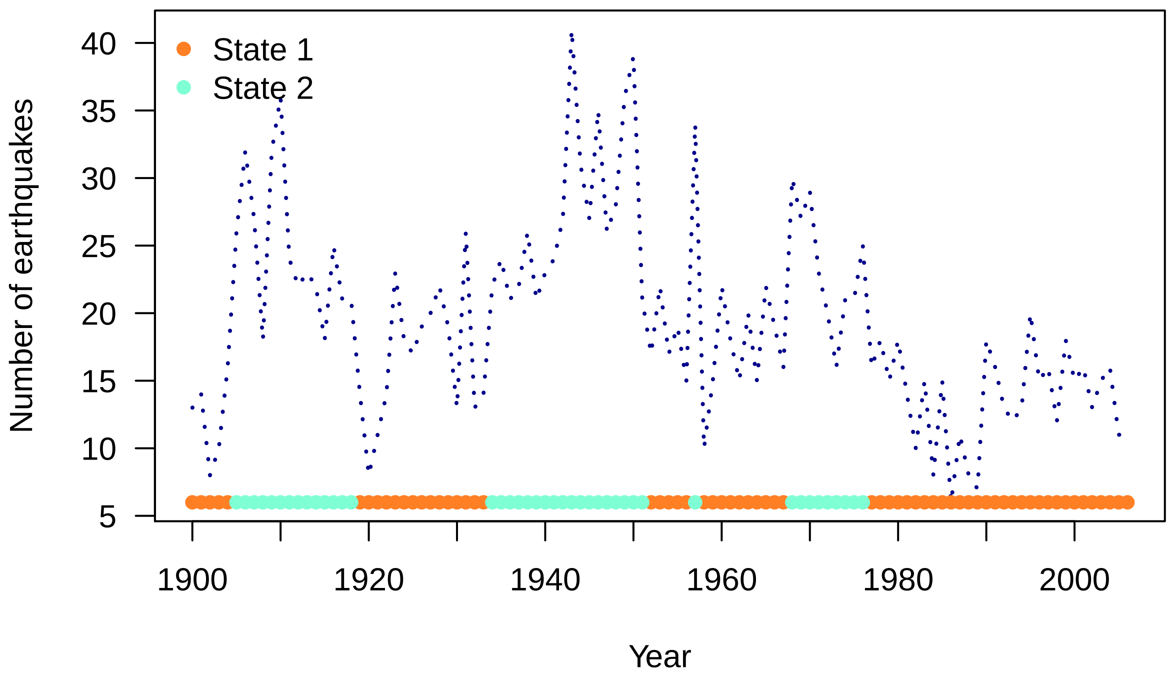


```
# according to the log likelihood, a model with 3 states would be best.
# However, when penalizing for the number of parameters estimated by obtaining the BIC,
# a model with 2 states appears best.
```

e)

Obtain the most likely state sequence over time using the best fitting model, and add it to the plot that contains the data over time.

```
states.earthq <- posterior(out.earth2)[,1]
plot(earthq.data, type = "l", xlab = "Year", ylab = "Number of earthquakes",
     lwd = 2, lty = 3, col = "darkblue", xaxt = 'n', las = 1)
axis(1, at = seq(1,105,10), labels = seq(1900,2005,10))
points(x = c(1:length(earthq.data))[states.earthq == 1],
       y = rep(6, sum(states.earthq == 1)), col = "chocolate1", pch = 16)
points(x = c(1:length(earthq.data))[states.earthq == 2],
       y = rep(6, sum(states.earthq == 2)), col = "aquamarine1", pch = 16)
legend("topleft", pch = 16, col = c("chocolate1", "aquamarine1"),
       legend = c("State 1", "State 2"), bty = "n")
```



Exercise 3

Next, we will use a fictional example inspired on a collaboration with dr. William Hale. We have the data on non-verbal communication between a therapist and its patient. For both, we collected:

- looking behaviour: the person either does or does not look at the other person
- vocalizing behaviour: the person is speaking, back channelling, or silent.

The data is in the file patient-therapist.csv.

a)

Plot the data over time.

```r
dyad.dat <- read.csv2(file = "patient-therapist.csv")
head(dyad.dat)
```

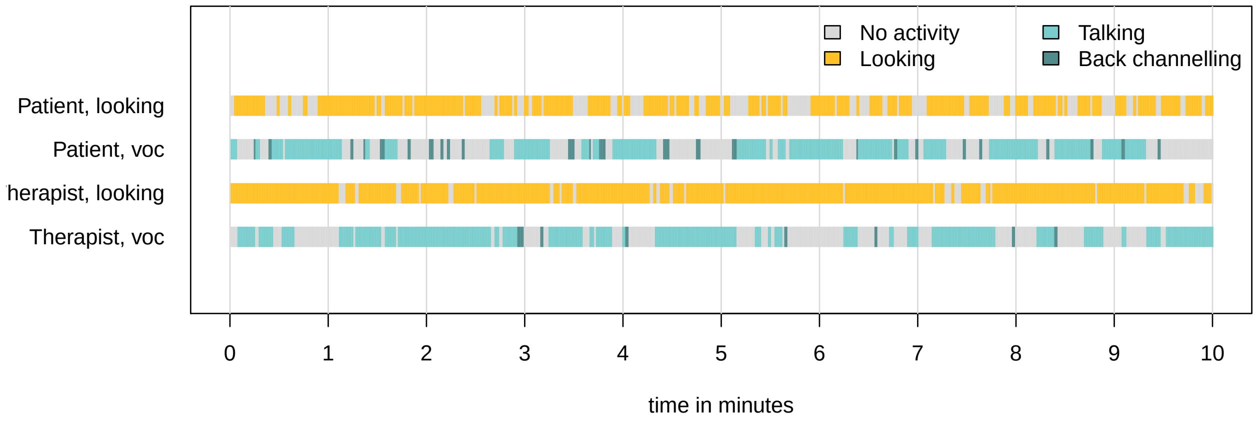
```
##   Looking_pat Vocalizing_pat Looking_ther Vocalizing_ther
## 1     Look_no      Voc_speak    Look_look          Voc_no
## 2     Look_no      Voc_speak    Look_look          Voc_no
## 3   Look_look      Voc_speak    Look_look          Voc_no
## 4   Look_look      Voc_speak    Look_look          Voc_no
## 5   Look_look         Voc_no    Look_look       Voc_speak
## 6   Look_look         Voc_no    Look_look       Voc_speak
```

```r
# make sure that the levels of the factor are in the correct order
dyad.dat$Vocalizing_pat <- factor(dyad.dat$Vocalizing_pat,
                                  levels = c("Voc_speak", "Voc_bch",  "Voc_no"))
dyad.dat$Looking_pat <- factor(dyad.dat$Looking_pat ,
                               levels = c("Look_look", "Look_no"))
dyad.dat$Vocalizing_ther <- factor(dyad.dat$Vocalizing_ther,
                                   levels = c("Voc_speak", "Voc_bch",  "Voc_no"))
dyad.dat$Looking_ther <- factor(dyad.dat$Looking_ther ,
                                    levels = c("Look_look", "Look_no"))

# plotting the data
Voc.col <- c("darkslategray3", "darkslategray4", "gray85")
Look.col <- c("goldenrod1", "gray85")

par(mar = c(4.1, 7.1, 4.1, 2.1))
plot(x = 1, xlim = c(0,600), ylim = c(0.5,7), type = "n", las = 1,
     xlab = "time in minutes", xaxt = "n", yaxt = "n", ylab = "")
axis(2, at = seq(5,2), tick = FALSE,
     labels = c("Patient, looking", "Patient, voc", "Therapist, looking",
                "Therapist, voc"), las = 1)
axis(1, at = seq(0,600,60), tick = TRUE, las = 1, labels = FALSE)
axis(1, at = seq(0,600,60), tick = FALSE, las = 1, labels = seq(0,10,1))
abline(v = seq(0,600,60), col = "gray85")

for(i in 1:2){
  points(x = c(1:600)[unclass(dyad.dat$Looking_pat) == i],
         y = rep(5, sum(unclass(dyad.dat$Looking_pat) == i)),pch = "|",
         col = Look.col[i])
}
for(i in 1:3){
  points(x = c(1:600)[unclass(dyad.dat$Vocalizing_pat) == i],
         y = rep(4, sum(unclass(dyad.dat$Vocalizing_pat) == i)),pch = "|",
         col = Voc.col[i])
}
for(i in 1:2){
  points(x = c(1:600)[unclass(dyad.dat$Looking_ther) == i],
         y = rep(3, sum(unclass(dyad.dat$Looking_ther) == i)),pch = "|",
         col = Look.col[i])
}
for(i in 1:3){
  points(x = c(1:600)[unclass(dyad.dat$Vocalizing_ther) == i],
         y = rep(2, sum(unclass(dyad.dat$Vocalizing_ther) == i)),pch = "|",
         col = Voc.col[i])
}
legend("topright", bty = "n", fill = c( "gray85",  Look.col[-2], Voc.col[-3]),
       legend = c("No activity", "Looking", "Talking", "Back channelling"), ncol = 2)
```



b)

Fit a 1 state, 2 state, 3 state, and 4 state HMM

```r
library(depmixS4)
set.seed(2314)

# 1 state model
Mdyad.s1 <- depmix(list(Looking_pat~1, Vocalizing_pat~1,
                        Looking_ther~1, Vocalizing_ther~1),
                   data = dyad.dat, nstates = 1,
                   family=list(multinomial("identity"), multinomial("identity"),
                               multinomial("identity"), multinomial("identity")))
fit.Mdyad.s1<- fit(Mdyad.s1)
```

```
## iteration 0 logLik: -2432.662
## converged at iteration 2 with logLik: -2432.662
```

```r
# 2 state model
Mdyad.s2 <- depmix(list(Looking_pat~1, Vocalizing_pat~1,
                        Looking_ther~1, Vocalizing_ther~1),
                   data = dyad.dat, nstates = 2,
                   family=list(multinomial("identity"), multinomial("identity"),
                               multinomial("identity"), multinomial("identity")))
fit.Mdyad.s2<- fit(Mdyad.s2)
```

```
## iteration 0 logLik: -2426.774
## iteration 5 logLik: -1855.215
## iteration 10 logLik: -1840.914
## iteration 15 logLik: -1835.782
## iteration 20 logLik: -1825.635
## iteration 25 logLik: -1824.882
## iteration 30 logLik: -1824.42
## iteration 35 logLik: -1824.375
## converged at iteration 40 with logLik: -1824.373
```

```r
# 3 state model
Mdyad.s3 <- depmix(list(Looking_pat~1, Vocalizing_pat~1,
                        Looking_ther~1, Vocalizing_ther~1),
                   data = dyad.dat, nstates = 3,
                   family=list(multinomial("identity"), multinomial("identity"),
                               multinomial("identity"), multinomial("identity")))
fit.Mdyad.s3<- fit(Mdyad.s3)
```

```
## iteration 0 logLik: -2423.551
## iteration 5 logLik: -1833.718
## iteration 10 logLik: -1727.923
## iteration 15 logLik: -1718.016
## iteration 20 logLik: -1717.86
## iteration 25 logLik: -1717.854
## iteration 30 logLik: -1717.852
## iteration 35 logLik: -1717.851
## iteration 40 logLik: -1717.85
## iteration 45 logLik: -1717.85
## iteration 50 logLik: -1717.85
## converged at iteration 51 with logLik: -1717.85
```

```
# 4 state model
Mdyad.s4 <- depmix(list(Looking_pat~1, Vocalizing_pat~1,
                        Looking_ther~1, Vocalizing_ther~1),
                   data = dyad.dat, nstates = 4,
                   family=list(multinomial("identity"), multinomial("identity"),
                               multinomial("identity"), multinomial("identity")))
fit.Mdyad.s4<- fit(Mdyad.s4)
```

```
## iteration 0 logLik: -2430.992
## iteration 5 logLik: -1908.168
## iteration 10 logLik: -1767.788
## iteration 15 logLik: -1730.855
## iteration 20 logLik: -1702.803
## iteration 25 logLik: -1696.956
## iteration 30 logLik: -1694.743
## iteration 35 logLik: -1686.585
## iteration 40 logLik: -1679.861
## iteration 45 logLik: -1679.106
## iteration 50 logLik: -1678.977
## iteration 55 logLik: -1678.953
## iteration 60 logLik: -1678.949
## iteration 65 logLik: -1678.948
## converged at iteration 70 with logLik: -1678.948
```

c)

Inspect the composition of the states in the different models, and give a theoretical interpretation to them. Do each of the states make sense?

```
# results 1 state model
summary(fit.Mdyad.s1)
```

```
## Initial state probabilties model
## pr1
##   1
##
## Transition matrix
##        toS1
## fromS1    1
##
## Response parameters
## Resp 1 : multinomial
## Resp 2 : multinomial
## Resp 3 : multinomial
## Resp 4 : multinomial
##     Re1.Look_look Re1.Look_no Re2.Voc_speak Re2.Voc_bch Re2.Voc_no
## St1         0.694       0.306         0.378       0.082       0.54
##     Re3.Look_look Re3.Look_no Re4.Voc_speak Re4.Voc_bch Re4.Voc_no
## St1         0.842       0.158         0.636       0.024       0.34
```

```
# Here, we just get the proportions per variable of each behavior in the dataset.
# Note that the first set of paremeters (looking and vocalizing) are for the patient,
# the second set corresponds to the therapist.

# results 2 state model
```

```
summary(fit.Mdyad.s2)
```

```
## Initial state probabilties model
## pr1 pr2
##   0   1
##
## Transition matrix
##         toS1  toS2
## fromS1 0.959 0.041
## fromS2 0.069 0.931
##
## Response parameters
## Resp 1 : multinomial
## Resp 2 : multinomial
## Resp 3 : multinomial
## Resp 4 : multinomial
##     Re1.Look_look Re1.Look_no Re2.Voc_speak Re2.Voc_bch Re2.Voc_no
## St1         0.750       0.250         0.000       0.133      0.867
## St2         0.604       0.396         0.993       0.000      0.007
##     Re3.Look_look Re3.Look_no Re4.Voc_speak Re4.Voc_bch Re4.Voc_no
## St1         0.779       0.221         0.861       0.004      0.135
## St2         0.944       0.056         0.268       0.058      0.675
```

```
# In state number 1, the patient is silent and the therapist speaks,
# in state number 2 this is reversed.

# results 3 state model
summary(fit.Mdyad.s3)
```

```
## Initial state probabilties model
## pr1 pr2 pr3
##   0   1   0
##
## Transition matrix
##         toS1  toS2  toS3
## fromS1 0.837 0.067 0.096
## fromS2 0.080 0.914 0.005
## fromS3 0.044 0.006 0.950
##
## Response parameters
## Resp 1 : multinomial
## Resp 2 : multinomial
## Resp 3 : multinomial
## Resp 4 : multinomial
##     Re1.Look_look Re1.Look_no Re2.Voc_speak Re2.Voc_bch Re2.Voc_no
## St1         0.631       0.369         0.534       0.108      0.358
## St2         0.550       0.450         1.000       0.000      0.000
## St3         0.795       0.205         0.000       0.109      0.891
##     Re3.Look_look Re3.Look_no Re4.Voc_speak Re4.Voc_bch Re4.Voc_no
## St1         0.812       0.188         0.527       0.023      0.449
## St2         1.000       0.000         0.000       0.071      0.929
## St3         0.781       0.219         0.996       0.002      0.002
```

```
# In state number 1, both the patient and the therapist are generally speaking.
# In state number 2, the patient speaks and the therapist is silent,
```

```r
# in state number 3, this is reversed: the therapist speaks and the patient is silent.

# results 4 state model
summary(fit.Mdyad.s4)
```

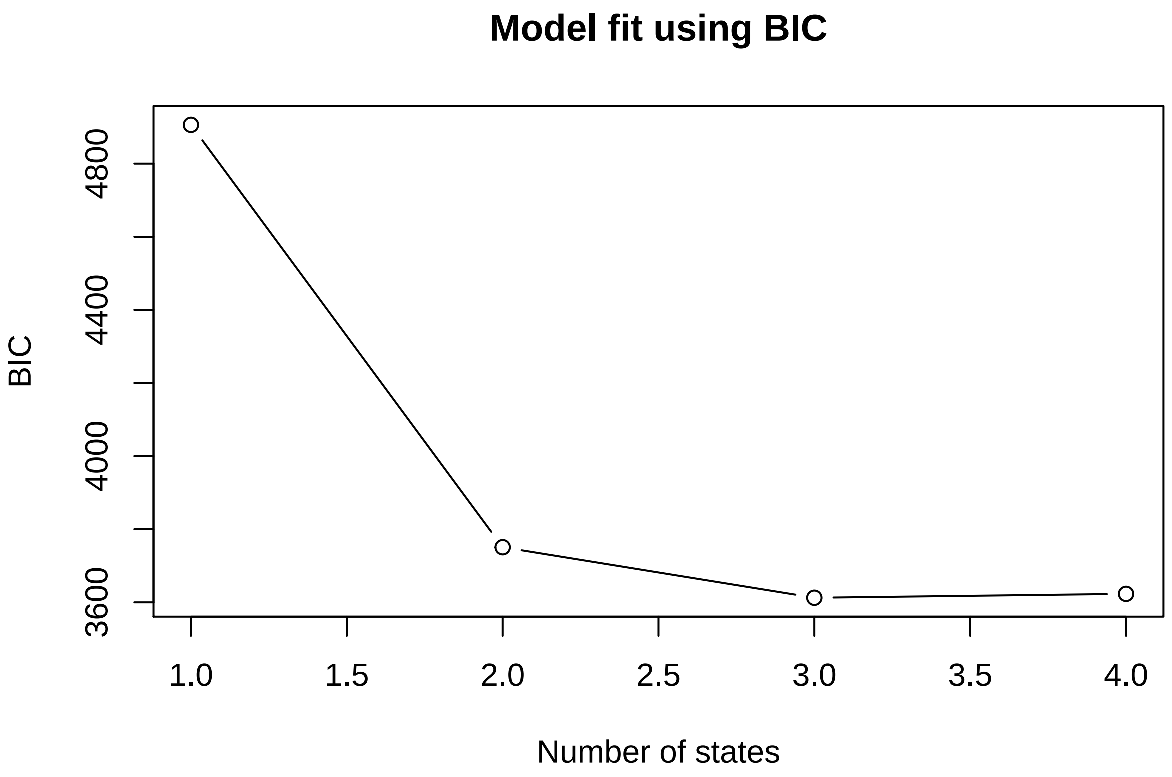
```
## Initial state probabilties model
## pr1 pr2 pr3 pr4
##   1   0   0   0
##
## Transition matrix
##         toS1  toS2  toS3  toS4
## fromS1 0.938 0.016 0.018 0.028
## fromS2 0.138 0.555 0.118 0.188
## fromS3 0.009 0.000 0.812 0.179
## fromS4 0.035 0.076 0.034 0.855
##
## Response parameters
## Resp 1 : multinomial
## Resp 2 : multinomial
## Resp 3 : multinomial
## Resp 4 : multinomial
##     Re1.Look_look Re1.Look_no Re2.Voc_speak Re2.Voc_bch Re2.Voc_no
## St1         0.586       0.414         0.895       0.013      0.092
## St2         0.727       0.273         0.000       1.000      0.000
## St3         0.754       0.246         0.048       0.000      0.952
## St4         0.787       0.213         0.000       0.000      1.000
##     Re3.Look_look Re3.Look_no Re4.Voc_speak Re4.Voc_bch Re4.Voc_no
## St1         0.963       0.037         0.239       0.056      0.704
## St2         0.957       0.043         0.862       0.000      0.138
## St3         0.181       0.819         0.738       0.008      0.254
## St4         0.951       0.049         1.000       0.000      0.000
```

```r
# In state number 1, the patient speaks and the therapist is (mostly) silent,
# In state number 2, the therapist speaks and the patient is back channelling.
# In both state number 3 and 4, the therapist speaks and the patient is silent.
# The difference is that in state number 4 the therapist is looking at the patient,
# while in state number 3 he/she is not.
```

d)

Which model would you choose to represent the data?

```r
plot(c(BIC(fit.Mdyad.s1), BIC(fit.Mdyad.s2), BIC(fit.Mdyad.s3), BIC(fit.Mdyad.s4)),
     type = "b", ylab = "BIC", xlab = "Number of states", main = "Model fit using BIC")
```



```
# According to the BIC, a model with 3 or 4 states fit the data equally well and hence
# adivizes to use 3 states. When looking at the composition of the states however,
# one could argue that the distinction between the therapist looking and not looking
# at the patient while speaking is  theoretically interesting, just as a state that
# represents back channelling of the patient.
# Therefore, one could also choose to use a 4 state model for theoretical reasons.
```

e)

Obtain the most likely state sequence over time, and add it to the plot that contains the data over time

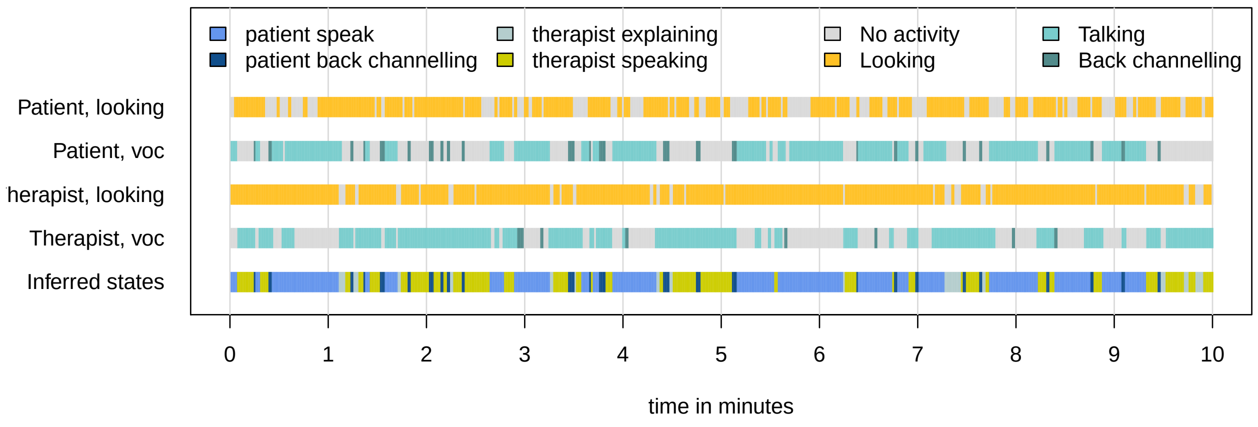
```
states.dyad <- posterior(fit.Mdyad.s4)[,1]

Voc.col <- c("darkslategray3", "darkslategray4", "gray85")
Look.col <- c("goldenrod1", "gray85")
col.states <- c("cornflowerblue", "dodgerblue4", "lightcyan3", "yellow3")

par(mar = c(4.1, 7.1, 4.1, 2.1))
plot(x = 1, xlim = c(0,600), ylim = c(0.5,7), type = "n", las = 1,
     xlab = "time in minutes", xaxt = "n", yaxt = "n", ylab = "")
axis(2, at = seq(5,1), tick = FALSE,
     labels = c("Patient, looking", "Patient, voc", "Therapist, looking",
                "Therapist, voc", "Inferred states"), las = 1)
axis(1, at = seq(0,600,60), tick = TRUE, las = 1, labels = FALSE)
axis(1, at = seq(0,600,60), tick = FALSE, las = 1, labels = seq(0,10,1))
abline(v = seq(0,600,60), col = "gray85")

for(i in 1:2){
  points(x = c(1:600)[unclass(dyad.dat$Looking_pat) == i],
         y = rep(5, sum(unclass(dyad.dat$Looking_pat) == i)),pch = "|",
         col = Look.col[i])
}
for(i in 1:3){
  points(x = c(1:600)[unclass(dyad.dat$Vocalizing_pat) == i],
         y = rep(4, sum(unclass(dyad.dat$Vocalizing_pat) == i)),pch = "|",
         col = Voc.col[i])
}
for(i in 1:2){
  points(x = c(1:600)[unclass(dyad.dat$Looking_ther) == i],
         y = rep(3, sum(unclass(dyad.dat$Looking_ther) == i)),pch = "|",
         col = Look.col[i])
}
for(i in 1:3){
  points(x = c(1:600)[unclass(dyad.dat$Vocalizing_ther) == i],
         y = rep(2, sum(unclass(dyad.dat$Vocalizing_ther) == i)),pch = "|",
         col = Voc.col[i])
}
legend("topright", bty = "n", fill = c( "gray85",  Look.col[-2], Voc.col[-3]),
       legend = c("No activity", "Looking", "Talking", "Back channelling"),
       ncol = 2)

for(i in 1:4){
    points(x = c(1:600)[states.dyad == i], y = rep(1, sum(states.dyad == i)),
           col = col.states[i], pch = "|")
}
legend("topleft", legend = c("patient speak", "patient back channelling",
                             "therapist explaining", "therapist speaking"),
       fill = col.states, ncol = 2, bty = "n")
```



```
# The state in which the thereapist is speaking but is not looking at the patient,
# is coined 'therapist explaining'.
```