Markov Hidden Model

Silvana Alvarez

2025-05-17

```
if (!require(depmixS4)) install.packages("depmixS4")
if (!require(R2OpenBUGS)) install.packages("R2OpenBUGS")

library(depmixS4)
library(ggplot2)
library(dplyr)
library(R2OpenBUGS)
library(lubridate)


df <- read.csv("seattle-weather.csv")
```

Introduction

This document compares the Frequentist and Bayesian approaches in setting up a Hidden Markov Models (HMM) set up to forecast weather, or more specifically wether it will rain on a given day or not. The kaggle dataset "weather prediction" (https://www.kaggle.com/datasets/ananthr1/weather-prediction?resource=download) contains daily weather observations. The dataset includes the following input variables:

- `precipitation`: Amount of precipitation recorded.
- `temp_max`: Maximum temperature of the day.
- `temp_min`: Minimum temperature of the day.
- `wind`: Wind speed.

and has **weather condition** as target, which is categorized into:

- `drizzle`
- `rain`
- `sun`
- `snow`
- `fog`

beyond that, we generate a new variable `temp_range` which is the difference between the maximum and minimum temperature. We then scale the continious variables before proceeding with the analysis.

```
# Create additional features
df$temp_range <- df$temp_max - df$temp_min

# Convert date column to Date type
```

```r
df$date <- as.Date(df$date)

# Scale continuous variables
df$temp_max <- scale(df$temp_max)
df$temp_min <- scale(df$temp_min)
df$wind     <- scale(df$wind)
df$precipitation <- scale(df$precipitation)
df$temp_range <- scale(df$temp_range)
```

Frequentist Approach

We begin with the frequentist approach, by estimating a transition matrix containing the probabilities to move between states. Using the frequentist approach, we can estimate the parameters of the model using the EM algorithm. We repeat this process for 2-5 states and store the resulting AIC/BIC values. According to both AIC and BIC, the 3 state model is the best fit. However, with the goal of predicting rain (true/false) we will proceed none the less with the simpler 2 state model.

QUESTION: What Mean ?

```r
aic_vals <- c()
bic_vals <- c()
loglik_vals <- c()
models <- list()

for (k in 2:5) {
  model_k <- depmix(
    list(temp_max ~ 1, temp_min ~ 1, wind ~ 1, precipitation ~ 1),
    data = df,
    nstates = k,
    family = list(gaussian(), gaussian(), gaussian(), gaussian())
  )
  fit_k <- fit(model_k, verbose = FALSE)
  models[[k]] <- fit_k

  aic_vals[k] <- AIC(fit_k)
  bic_vals[k] <- BIC(fit_k)
  loglik_vals[k] <- logLik(fit_k)
}
```

```
## Warning in em.depmix(object = object, maxit = emcontrol$maxit, tol =
## emcontrol$tol, : Log likelihood decreased on iteration 13 from 20628.5641578465
## to 18741.2412855796
```

```
## Warning in em.depmix(object = object, maxit = emcontrol$maxit, tol =
## emcontrol$tol, : Log likelihood decreased on iteration 10 from 20282.7628556911
## to 18946.3354186447
```

```
## Warning in em.depmix(object = object, maxit = emcontrol$maxit, tol =
## emcontrol$tol, : Log likelihood decreased on iteration 11 from 19376.7221099977
## to 18946.3328047558
```

```
## Warning in em.depmix(object = object, maxit = emcontrol$maxit, tol =
## emcontrol$tol, : Log likelihood decreased on iteration 11 from 23176.0952821394
## to 23036.9310471871
```

```
data.frame(k = 2:5, AIC = aic_vals[2:5], BIC = bic_vals[2:5], LogLik = loglik_vals[2:5])
```

```
##   k       AIC       BIC   LogLik
## 1 2 -37444.48 -37344.03 18741.24
## 2 3 -37828.67 -37659.49 18946.34
## 3 4 -37798.67 -37550.18 18946.33
## 4 5 -45945.86 -45607.50 23036.93
```

By printing the summary of the chosen 2 state model we can find as general caracteristics of the two states, that state 1 shows negative precipitation mean, positive temperature range, giving hints towards this state representing sunny/foggy weather. Meanwhile state 2 shows high precipitation mean and low temp range, which appears to cover rainy weather well. Based on the weather distribution between the states we thus label them as:

- State 1: Dry weather (sunny, foggy, driztle)
- State 2: Wet weather (rainy, snow)

```
set.seed(123)

# Define HMM with 2 states
hmm_model <- depmix(
  list(temp_max ~ 1,
       temp_min ~ 1,
       wind ~ 1,
       precipitation ~ 1,
       temp_range ~ 1),
  data = df,
  nstates = 2,
  family = list(gaussian(), gaussian(), gaussian(), gaussian(), gaussian())
)

# Fit the model
hmm_fit <- fit(hmm_model, verbose = FALSE)
```

```
## Warning in em.depmix(object = object, maxit = emcontrol$maxit, tol =
## emcontrol$tol, : Log likelihood decreased on iteration 11 from 19161.7254711651
## to 16761.5781319448
```

```
# View model parameters
summary(hmm_fit)
```

```
## Initial state probabilities model
## pr1 pr2
##   1   0
##
## Transition matrix
##         toS1  toS2
```

```
## fromS1 0.756 0.244
## fromS2 0.327 0.673
##
## Response parameters
## Resp 1 : gaussian
## Resp 2 : gaussian
## Resp 3 : gaussian
## Resp 4 : gaussian
## Resp 5 : gaussian
##     Re1.(Intercept) Re1.sd Re2.(Intercept) Re2.sd Re3.(Intercept) Re3.sd
## St1           0.348  1.054           0.140  1.090          -0.256  0.816
## St2          -0.469  0.684          -0.188  0.827           0.345  1.113
##     Re4.(Intercept) Re4.sd Re5.(Intercept) Re5.sd
## St1          -0.453  0.000           0.487  0.956
## St2           0.610  1.302          -0.655  0.605
```

```r
# Predict hidden states
df$state_HMM <- posterior(hmm_fit)$state
```

```
## Warning in .local(object, ...): Argument 'type' not specified and will default
## to 'viterbi'. This default may change in future releases of depmixS4. Please
## see ?posterior for alternative options.
```

```r
# Compare with original weather labels
table(Predicted = df$state_HMM, Actual = df$weather)
```

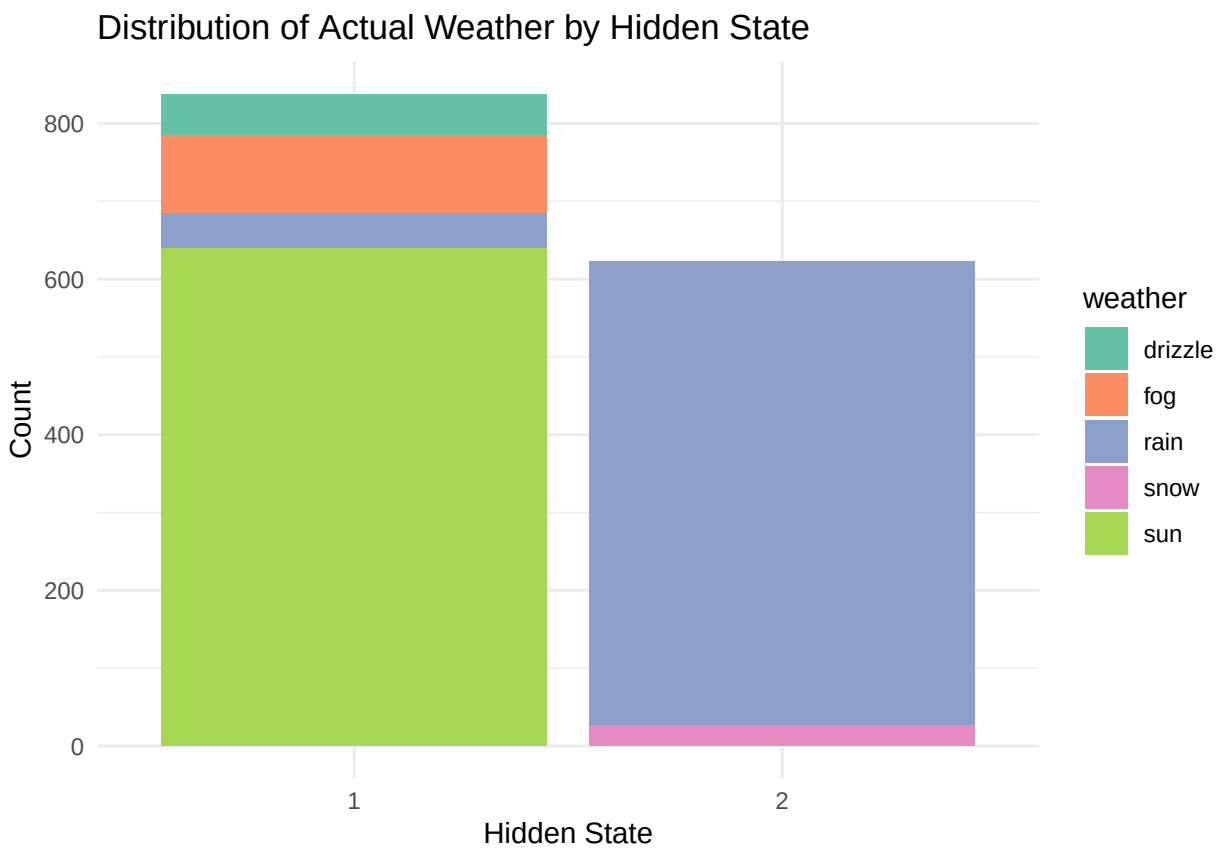
```
##          Actual
## Predicted drizzle fog rain snow sun
##         1      53 101   44    0 640
##         2       0   0  597   26   0
```

```r
# Compare with original weather labels
table(Predicted = df$state_HMM, Actual = df$weather)
```

```
##          Actual
## Predicted drizzle fog rain snow sun
##         1      53 101   44    0 640
##         2       0   0  597   26   0
```

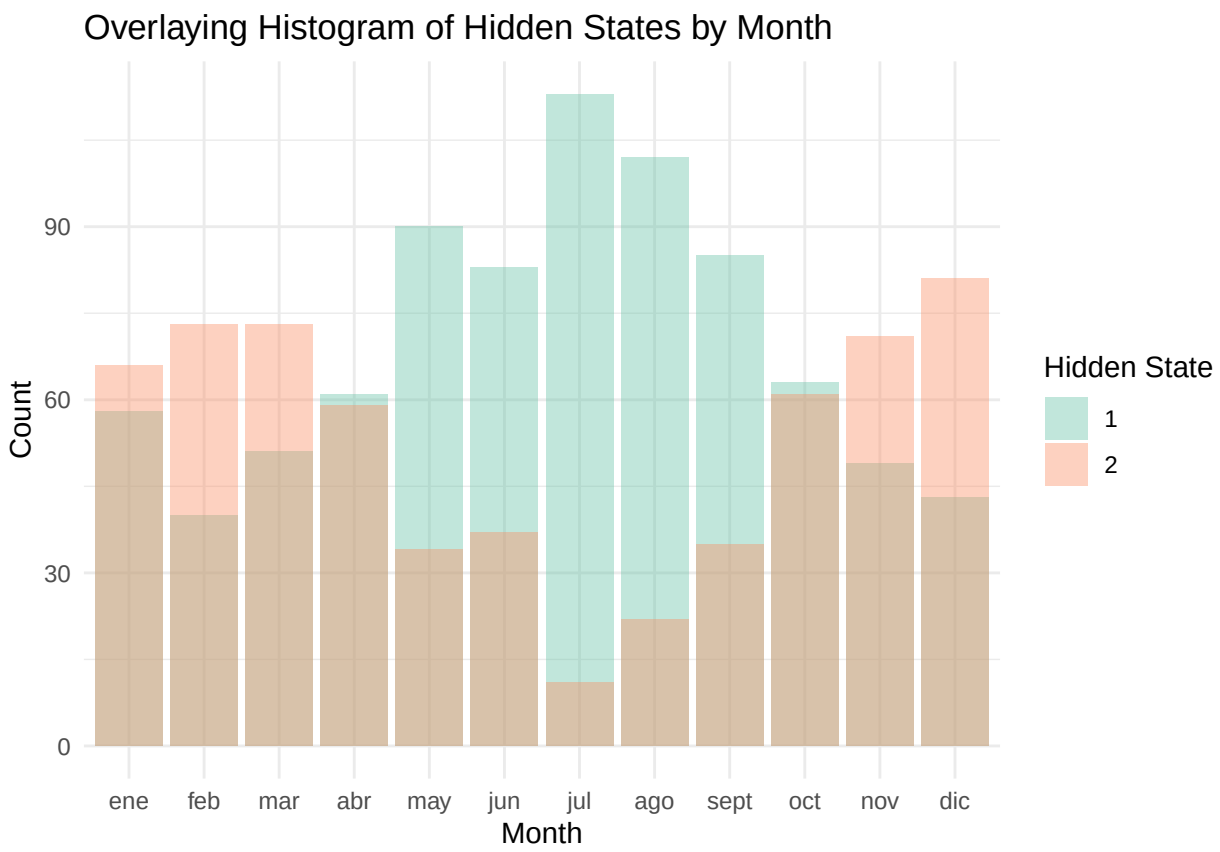
These assumptions can be verified by plotting the predicted hidden states against the actual weather labels. There we see that our seperation of the states is indeed fitting.

```r
# Count of actual weather types per hidden state
df %>%
  count(state_HMM, weather) %>%
  ggplot(aes(x = factor(state_HMM), y = n, fill = weather)) +
  geom_bar(stat = "identity", position = "stack") +
  labs(title = "Distribution of Actual Weather by Hidden State",
       x = "Hidden State", y = "Count") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2")
```



We can further validate by looking at the distribution of the states over the months of the year. Here we see the conventional weather patterns, with "better" weather in the summer months (State 1) and "worse" weather in the winter months (State 2).

```
# Extract month
df <- df %>%
  mutate(month = month(date, label = TRUE, abbr = TRUE))  # e.g., Jan, Feb


# Plot: histogram of states by month
ggplot(df, aes(x = month, fill = factor(state_HMM))) +
  geom_bar(position = "identity", alpha = 0.4) +
  labs(title = "Overlaying Histogram of Hidden States by Month",
       x = "Month", y = "Count", fill = "Hidden State") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2")
```



By defining this split as "true states", we can then score the model by comparing the predicted states with the true states. We can see that the model is able to predict the weather with an accuracy of 97%.

```
# score
df$state_true <- ifelse(df$weather %in% c("drizzle", "fog", "sun"), 1,
                        ifelse(df$weather %in% c("rain", "snow"), 2, NA))

conf_matrix_HMM <- table(Predicted = df$state_HMM, Actual = df$state_true)
print(conf_matrix_HMM)
```

```
##           Actual
## Predicted   1   2
##         1 794  44
##         2   0 623
```

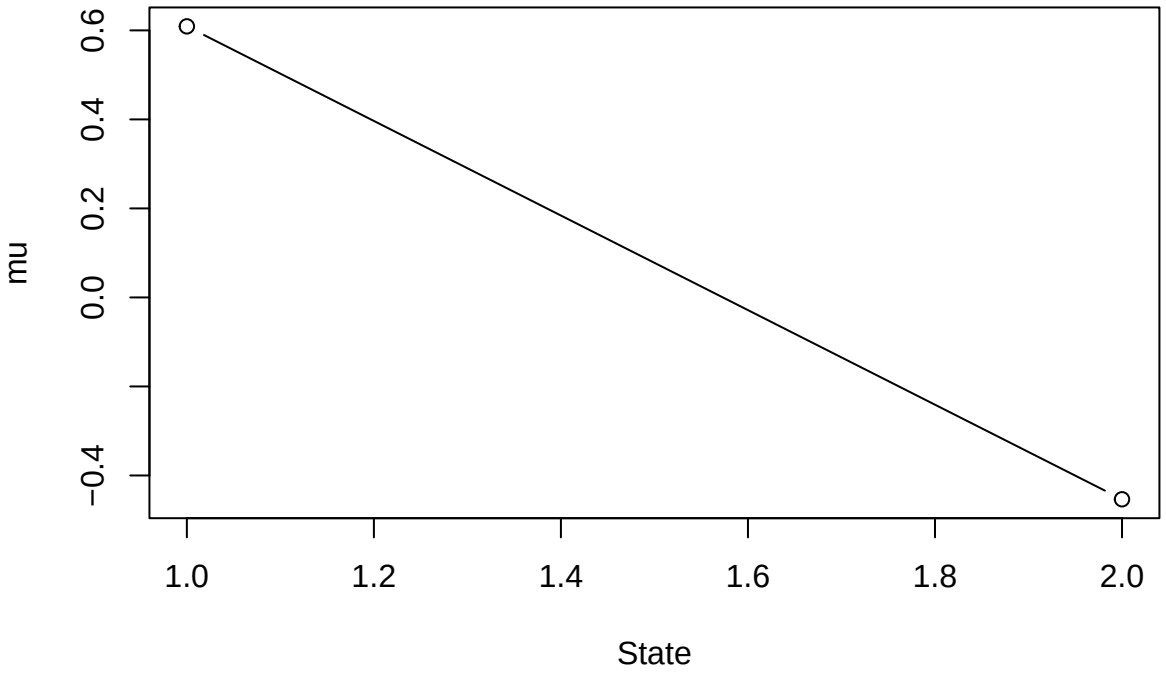
```
accuracy_HMM <- sum(diag(conf_matrix_HMM)) / sum(conf_matrix_HMM)
accuracy_HMM
```

```
## [1] 0.9698836
```

Bayesian Approach

An alternative to the Frequentist approach is the Bayesian approach to yield richer uncertainty information (e.g., credible intervals). Here, we create a transition matrix using MCMC to obtain the posterior distribution of parameters, with the mean as the posterior mean.

```r
y <- as.numeric(df$precipitation) # standardize for numerical stability
n <- length(y)
k <- 2  # number of hidden states

# Gaussian HMM - Bugs Model Function
bugsHMM <- function(){
  s[1] ~ dcat(P0[])
  y[1] ~ dnorm(mu[s[1]], tau[s[1]])

  for (j in 2:n){
    s[j] ~ dcat(P.mat[s[j-1],])
    y[j] ~ dnorm(mu[s[j]], tau[s[j]])
  }

  for(i in 1:k){
    alpha[i] <- 1
    P0[i] <- 1/k
    P.mat[i,1:k] ~ ddirich(alpha[])

    mu[i] ~ dnorm(0, 0.01)
    tau[i] ~ dgamma(0.001, 0.001)
  }
}

# Data and Initial Values
data <- list("k" = k, "n" = n, "y" = y)

inits <- function() {
  list(mu = rnorm(k), tau = rgamma(k, 1, 1))
}

params <- c("mu", "tau", "P.mat", "s")

# Run the model
res <- bugs(data, inits, parameters.to.save = params,
            model.file = bugsHMM,
            n.iter = 10000, n.burnin = 2000, n.chains = 3,
            codaPkg = FALSE,
            OpenBUGS.pgm = "C:/Program Files (x86)/OpenBUGS/OpenBUGS323/OpenBUGS.exe")

# Posterior Analysis
cc <- res$sims.list
mu <- cc$mu
plot(apply(mu, 2, mean), type = 'b', main = "Posterior Means for mu", ylab = "mu", xlab = "State")
```



With the model set up and validated, we proceed to predicting the states.

```
## Predicted states
s <- cc$s
ps <- matrix(0, nrow=n, ncol=k)
maxstate <- rep(0, n)

for (i in 1:n){
  for (j in 1:k){
    ps[i,j] <- sum(s[,i]==j)
  }
  maxstate[i] <- which.max(ps[i,])
}
df$state_bayesian <- maxstate
```

And then again score the resulting predictions, yielding a accuracy of XXXXXXXXX

```
# score for Bayesian
conf_matrix_HMM <- table(Predicted = df$state_bayesian, Actual = df$state_true)
print(conf_matrix_HMM)
```

```
##          Actual
## Predicted   1   2
##         1   0 623
##         2 794  44
```

```
accuracy_HMM <- sum(diag(conf_matrix_HMM)) / sum(conf_matrix_HMM)
accuracy_HMM
```

```
## [1] 0.03011636
```

Conclusion

Comparing both the Frequentist HMM and the Bayesian HMM approach, we see the XXXX approach outperform the other.