Transition Matrices

Transition matrix from one location for early summer subseason fire risk:

$$P = \begin{array}{c} \\ \text{Nil} \\ \text{Low} \\ \text{Moderate} \\ \text{High} \\ \text{Extreme} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Nil} & \text{Low} & \text{Moderate} & \text{High} & \text{Extreme} \end{array} \\ \begin{pmatrix} 0.575 & 0.118 & 0.172 & 0.109 & 0.026 \\ 0.453 & 0.243 & 0.148 & 0.123 & 0.033 \\ 0.104 & 0.343 & 0.367 & 0.167 & 0.019 \\ 0.015 & 0.066 & 0.318 & 0.505 & 0.096 \\ 0.000 & 0.060 & 0.149 & 0.567 & 0.224 \end{pmatrix} \end{array} \tag{1}$$

```
weather_P = matrix(c(0.575,0.118 , 0.172 , 0.109 , 0.026,
0.453 , 0.243 , 0.148 , 0.123 , 0.033,
0.104 , 0.343 , 0.367 , 0.167 , 0.019 ,
0.015 , 0.066 , 0.318 , 0.505 , 0.096 ,
0.000 , 0.060 , 0.149 , 0.567 , 0.224),ncol=5,nrow=5,byrow=T)
```

Code for computing P^n from textbook

```
##### Matrix powers ###############################
# matrixpower(mat,k) mat^k
#
matrixpower = function(X, n){

  if(dim(X)[1] != dim(X)[2]){
     throw("Dimensions of the matrix do not match: ", dim(X))
  }
  if (n==0){return(diag(dim(X)[1]))}
  else if(n >0) {return(X %*% matrixpower(X, (n-1)))}
  else {return(matrixpower(solve(X), -n))}
}
```

For n = 3

```
matrixpower(weather_P,3)
```

```
##           [,1]      [,2]      [,3]      [,4]       [,5]
## [1,] 0.3317973 0.1762260 0.2353411 0.2111096 0.04552595
## [2,] 0.3263579 0.1753868 0.2352439 0.2160688 0.04694271
## [3,] 0.2830784 0.1922351 0.2466504 0.2293466 0.04868947
## [4,] 0.1579034 0.1832159 0.2798370 0.3123858 0.06665790
## [5,] 0.1177433 0.1654309 0.2858074 0.3532858 0.07773251
```

For n = 10

```
matrixpower(weather_P,10)
```

```
##           [,1]      [,2]      [,3]      [,4]       [,5]
## [1,] 0.2643504 0.1812413 0.2518115 0.2491008 0.05349592
## [2,] 0.2642635 0.1812455 0.2518332 0.2491513 0.05350655
## [3,] 0.2640283 0.1812567 0.2518919 0.2492878 0.05353532
## [4,] 0.2625915 0.1813257 0.2522504 0.2501214 0.05371100
## [5,] 0.2618765 0.1813600 0.2524288 0.2505362 0.05379840
```

Simulating from the chain (Code from textbook)

```
###### Simulate discrete-time Markov chain ########################
# Simulates n steps of a Markov chain
# markov(init,mat,n,states)
# Generates X0, ..., Xn for a Markov chain with initiial
#  distribution init and transition matrix mat
# Labels can be a character vector of states; default is 1, .... k

markov <- function(init,mat,n,labels) {
    if (missing(labels)) labels <- 1:length(init)
simlist <- numeric(n+1)
states <- 1:length(init)
simlist[1] <- sample(states,1,prob=init)
for (i in 2:(n+1))
    { simlist[i] <- sample(states,1,prob=mat[simlist[i-1],]) }
labels[simlist]
}
#####################################################
```

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------- tidyverse 1
```

```
## v ggplot2 3.1.0     v purrr   0.3.0
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

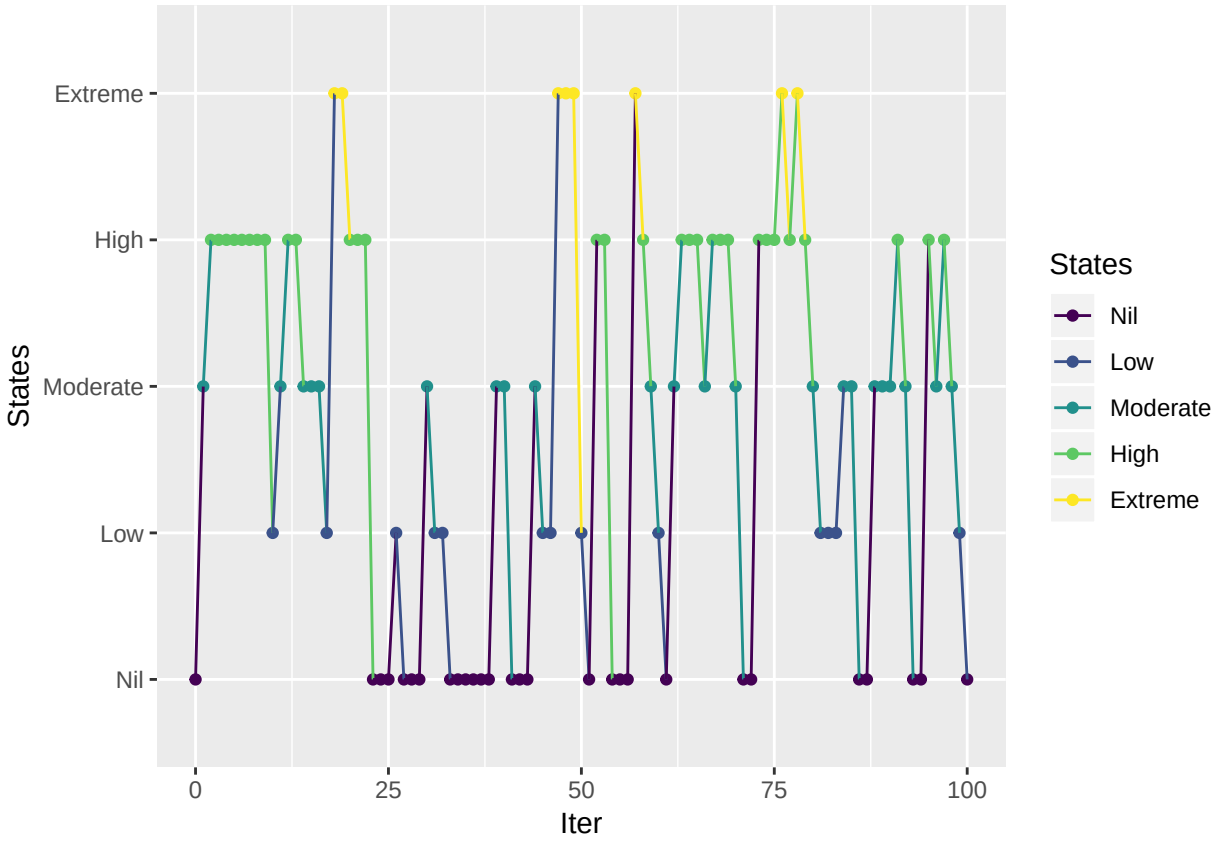
```
## -- Conflicts ---------------------------------------------------------------------- tidyverse_conflic
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
###@ Define initial distribution

init_prob = c(0.4,0.25,0.2,0.1) ## First four states
init_prob = c(init_prob, 1 - sum(init_prob)) ## Forces sum to one

iter=100
simulated_states = markov(init_prob,weather_P,iter,
                          labels=c("Nil","Low","Moderate","High","Extreme"))

simulated_states  = ordered(simulated_states,
                            c("Nil","Low","Moderate","High","Extreme"))

sim_df = data_frame(Iter=0:iter, States = simulated_states)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
```

```
## This warning is displayed once per session.
```

```
ggplot(sim_df,aes(x=Iter,y=States,group=1)) +
  geom_point(aes(colour=States)) +geom_line(aes(colour=States))
```



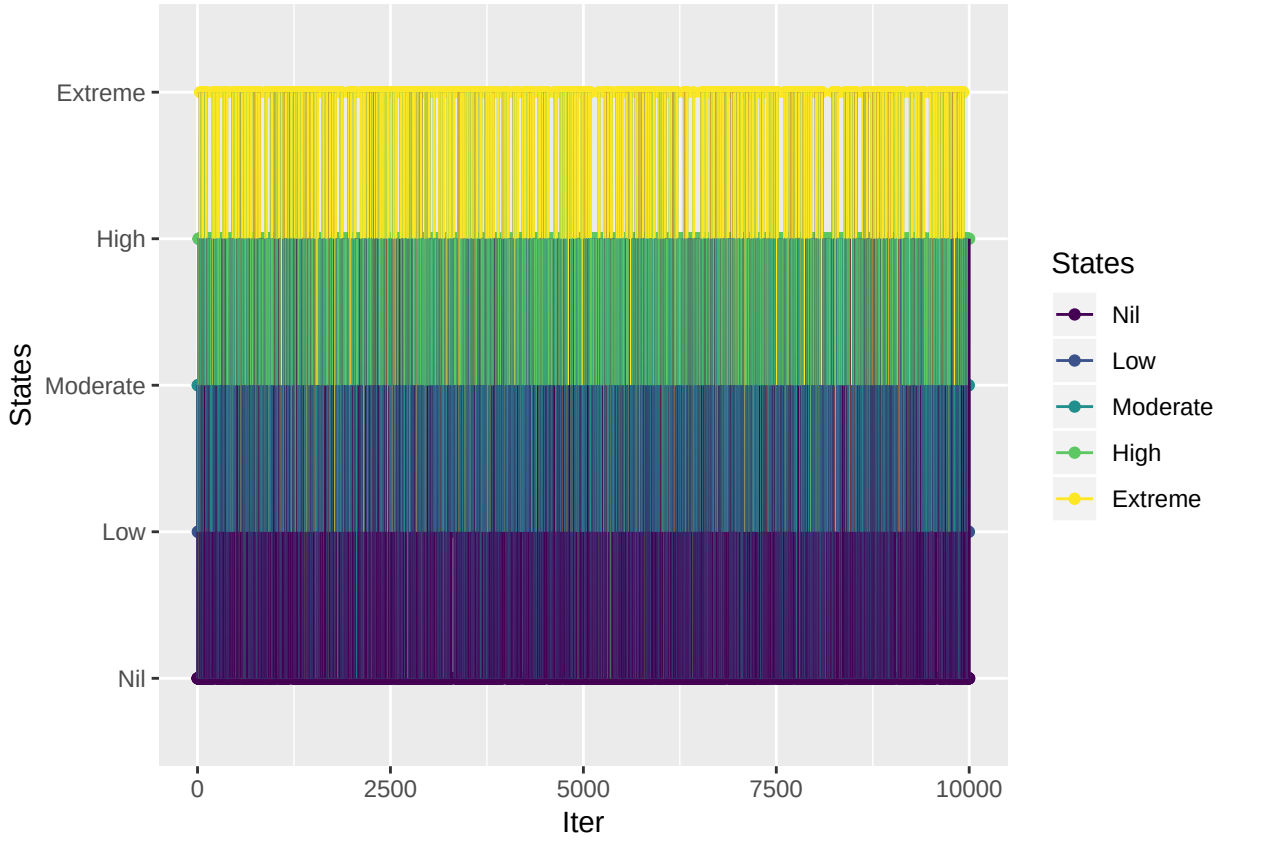

```
sim_df %>% group_by(States) %>% summarise(Counts=n())
```

```
## # A tibble: 5 x 2
##   States   Counts
##   <ord>     <int>
## 1 Nil          28
## 2 Low          13
## 3 Moderate     22
## 4 High         30
## 5 Extreme       8
```

Larger number of iterations

```
init_prob = c(0.4,0.25,0.2,0.1) ## First four states
init_prob = c(init_prob, 1 - sum(init_prob)) ## Forces sum to one

iter=10000
simulated_states = markov(init_prob,weather_P,iter,
                          labels=c("Nil","Low","Moderate","High","Extreme"))

sim_df = data_frame(Iter=0:iter, States =
                      ordered(simulated_states,
```

```
                                c("Nil","Low","Moderate","High","Extreme")))

ggplot(sim_df,aes(x=Iter,y=States,group=1)) +
  geom_point(aes(colour=States)) +geom_line(aes(colour=States))
```




```
sim_df %>% group_by(States) %>% summarise(Counts=n()) %>%
    mutate(Proportion=Counts/sum(Counts))
```

```
## # A tibble: 5 x 3
##   States   Counts Proportion
##   <ord>     <int>      <dbl>
## 1 Nil        2512     0.251
## 2 Low        1782     0.178
## 3 Moderate   2505     0.250
## 4 High       2618     0.262
## 5 Extreme     584     0.0584
```