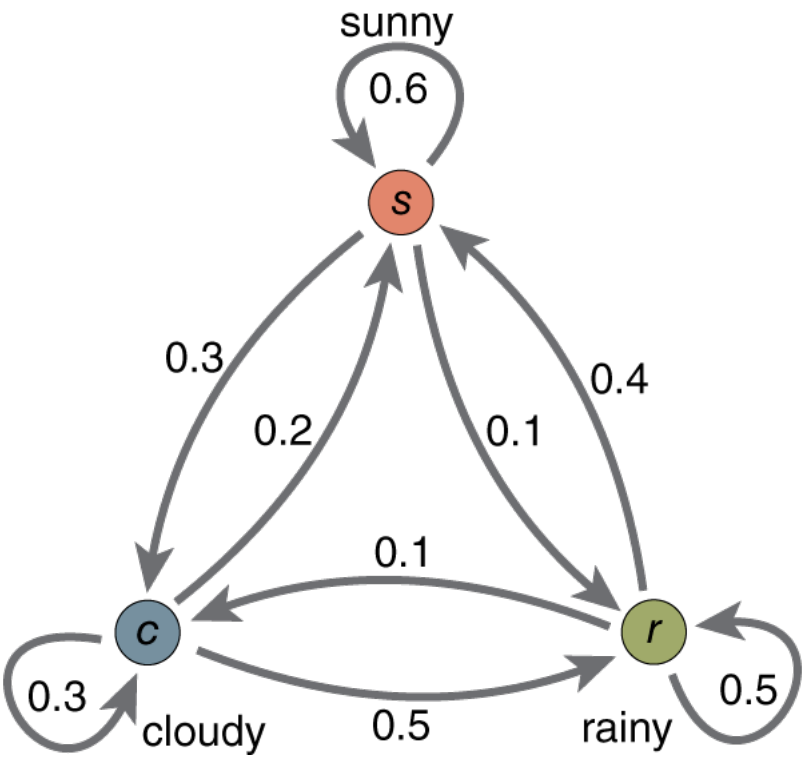
Stat133 Post #2

Alan Chuang

11/24/2017

Introduction to Markov Chains using the Package `markovchain`



A sample Markov Chain.

Introduction

Markov Chains are extremely powerful tools for computing and working with probabilities of events. Formally, a Markov Chain is a random process where we make use of previous states to predict the probabilities of future states. While this may seem like a lot, Markov Chains are not too difficult to fully grasp after some practice. In this tutorial, we'll mainly be working with the package `markovchain`. For an extensive introduction to this package, see here.

You can install it by running the following code in RStudio:

```
install.packages("markovchain")
```

Make sure you have all of the dependencies installed, you can do this by selected the "Install Dependencies" option from RStudio. The dependencies include: `igraph`, `Matrix`, `matlab`, `expm`, `stats4`, `parallel`, `Rcpp (≥ 0.11.5)`, `RcppParallel`, `utils`, `stats`, and `grDevices`. Installing the most recent version of the `markovchain` package will suffice for the purposes of this post.

Make sure to load the package before we get started!

```
library(markovchain)
```

```
## Package:  markovchain
## Version:  0.6.9.8-1
## Date:     2017-08-15
## BugReport: http://github.com/spedygiorgio/markovchain/issues
```

Let's work through an instructive example on how Markov Chains work, and how we can use the `markovchain` package to do some computations.

You're an Uber Driver!

An Uber-Special First Example!

Let's say you are an Uber Driver in Berkeley. To simplify things, there are three locations you can drive to and from: Northside, Southside, and Downtown. If you are Downtown, 30% of the time your next ride goes Northside, 30% of the time it goes Southside, and 40% of the time it stays Downtown. If you are Northside, 40% of the time it goes Southside, 20% of the time it goes Downtown, and 40% of the time it stays Northside. Finally, if you are Southside, 50% of the time it goes Downtown, 30% of the time it goes Northside, and 20% of the time it stays Southside.

Firstly, we can model the Markov Chain with what is called a transition matrix. This is a simple matrix that encodes the probabilities from moving from one state to the next state. In our case, it has the probabilities from going from one location of Berkeley to another location in Berkeley.

```
driveStates <- c( "Downtown", "Northside", "Southside")
driveTransition <- matrix(c(0.4, 0.3, 0.3,
                            + 0.2, 0.4, 0.4,
                            + 0.5, 0.3, 0.2),
                            byrow = TRUE, nrow = 3, dimnames = list(driveStates, driveStates))
driveTransition
```

```
##           Downtown Northside Southside
## Downtown       0.4       0.3       0.3
## Northside      0.2       0.4       0.4
## Southside      0.5       0.3       0.2
```

Let's take a moment to try to decipher this matrix. If we interpret the rows as meaning "From" and the columns as meaning "To", we can see tell the probabilities of driving to a new location from a previous location. For example, we can see that the probability of going Southside from Downtown is 0.3, and the probability of going Downtown from Northside is 0.2.

Now, we can begin to model this using a Markov Chain! We use the function `new` to create a Markov Chain object.

```
#Initialize the Markov Chain
uberDrive = new("markovchain", states = driveStates, byrow = TRUE, transitionMatrix = driveTransition, name = "Dri
ving Destinations")

#This gives us the transition matrix
uberDrive
```

```
## Driving Destinations
##  A  3 - dimensional discrete Markov Chain defined by the following states:
##  Downtown, Northside, Southside
##  The transition matrix  (by rows)  is defined as follows:
##           Downtown Northside Southside
## Downtown       0.4       0.3       0.3
## Northside      0.2       0.4       0.4
## Southside      0.5       0.3       0.2
```

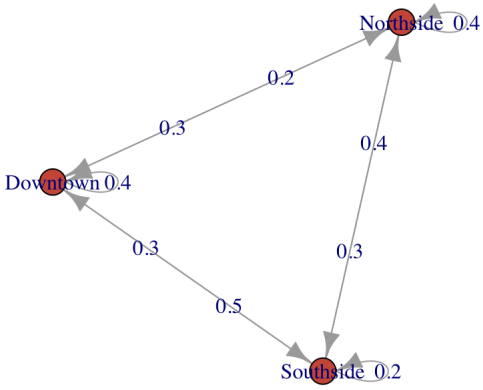
Now, if we want to see the probabilities of going to different places from a specific location, we can access this by rows. For example, to see the probability of going places from Northside, we can do the following.

```
#Probabilities that we drive Downtown, Northside, and Southside given that we are currently Northside
uberDrive[2]
```

```
##  Downtown Northside Southside
##       0.2       0.4       0.4
```

We can also draw a convenient diagram of our Markov Chain! This is done using the function `plot`.

```
plot(uberDrive)
```



There are ways to create more aesthetically pleasing diagrams of Markov Chains, but aesthetics are not the main focus of this post. However, if you are interested, you can see how to do so here.

Now, let's try to ask and answer some interesting questions about the stated problem. Given that we are currently Downtown, what is the probability that we end up Northside after *TWO* trips?

The naive approach would be to figure out all the possible ways to end up Northside after two trips, and add up all of the probabilities. Let's do this first.

```
#Go Downtown -> Northside
dn <- 0.4 * 0.3
#Go Southside -> Northside
sn <- 0.3 * 0.3
#Go Northside -> Northside
nn <- 0.3 *  0.4

#Probability that we will end up Northside
dn + sn + nn
```

```
## [1] 0.33
```

However, Markov Chains make solving this problem (and more complex problems of the same nature) much easier. We can simply raise our state transition matrix to the nth power, where n is the number of states we traverse (number of trips). If this seems unintuitive, take a look at this resource to learn more of the theory behind Markov Chains.

Let's raise our matrix to the Second Power.

```
uberDrive^2
```

```
## Driving Destinations^2
##  A  3 - dimensional discrete Markov Chain defined by the following states:
##  Downtown, Northside, Southside
##  The transition matrix  (by rows)  is defined as follows:
##            Downtown Northside Southside
## Downtown      0.37      0.33      0.30
## Northside     0.36      0.34      0.30
## Southside     0.36      0.33      0.31
```

This matrix tells us that the probability that we end up Northside when starting Downtown after 2 trips is 0.33, which is the same result we got earlier!

We can do this for any number of trips.

```
uberDrive^4
```

```
## Driving Destinations^4
##  A  3 - dimensional discrete Markov Chain defined by the following states:
##  Downtown, Northside, Southside
##  The transition matrix  (by rows)  is defined as follows:
##            Downtown Northside Southside
## Downtown    0.3637    0.3333    0.3030
## Northside   0.3636    0.3334    0.3030
## Southside   0.3636    0.3333    0.3031
```

```
uberDrive^10
```

```
## Driving Destinations^10
##  A  3 - dimensional discrete Markov Chain defined by the following states:
##  Downtown, Northside, Southside
##  The transition matrix  (by rows)  is defined as follows:
##            Downtown Northside Southside
## Downtown  0.3636364 0.3333333 0.3030303
## Northside 0.3636364 0.3333333 0.3030303
## Southside 0.3636364 0.3333333 0.3030303
```

Notice that the distribution of the time we spend Downtown, Northside, and Southside gets more and more "stationary" the more trips we take. This is what is known as the *stationary distribution* of the Markov Chain.

Conveniently, there is a function for this included in the `markovchain` package.

```
steadyStates(uberDrive)
```

```
##       Downtown Northside Southside
## [1,] 0.3636364 0.3333333 0.3030303
```

This tells us, that after a reasonable number of trips, we expect to be Downtown roughly 36% of the time, Northside 33% of the time, and Southside 30% of the time.

Let's use this feature to answer another interesting question.

Say you have 9 other friends who are Uber Drivers in Berkeley. After driving all day, how many of you do you expect to end up Downtown? We can do this using the steady states! After driving all day, we can expect to have reached the *stationary distribution*. Thus, we can simply multiply by 10!

```
10 * steadyStates(uberDrive)
```

```
##      Downtown Northside Southside
## [1,] 3.636364  3.333333  3.030303
```

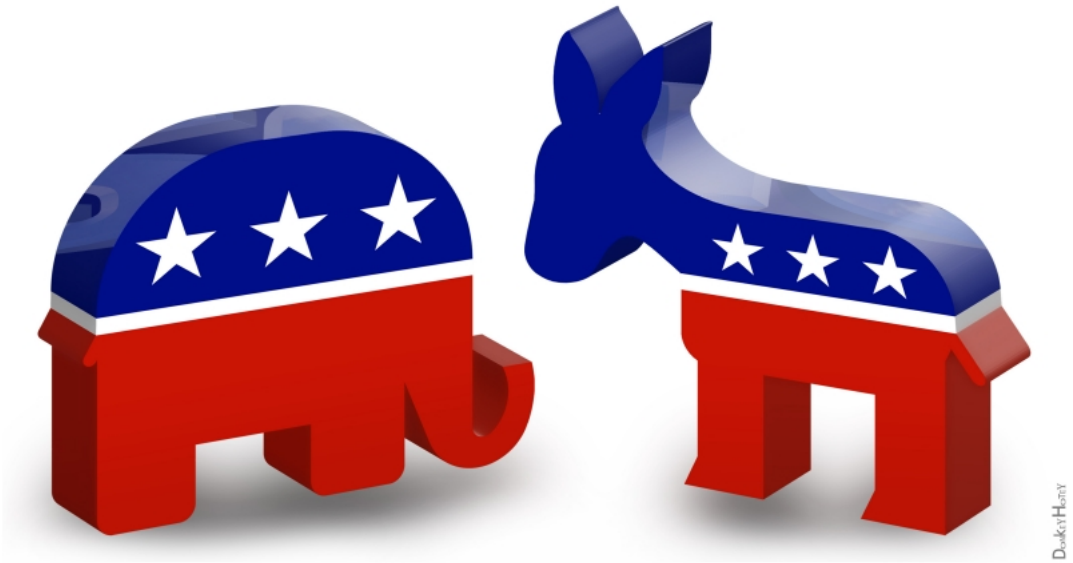
Thus, we expect 3.64 people to end up Downtown.

Now that we have established the basics of Markov Chains, here is some practice for you to try!

Practice Problem!

Let's say you are polling the next election. There are three parties: Democrats, Republicans, and Independents. Democrats have a 85% chance to stay Democrat, 10% chance to go Independent, and 5% chance to go Republican. Republicans have a 80% chance to stay Republican, 10% chance to go Independent, and 10% chance to go Democrat. Independents have a 40% chance to stay Independent, 30% to go Democrat, and 30% chance to go Republican. What are the states in this problem? What is the state transition matrix? Model this using a Markov Chain, draw a diagram, and find the stationary distribution.

Try this problem for yourself before taking a look at the solution that follows.



The two big parties in the United States.

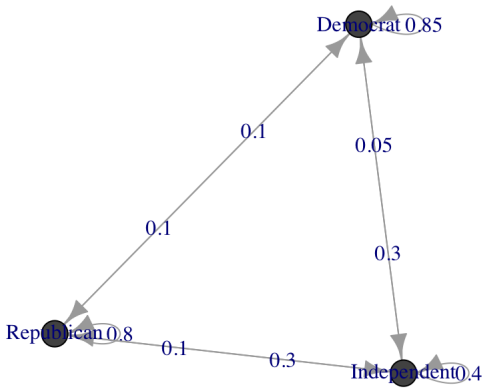
```
#Setting up the States and Transition Matrix
partyStates <- c( "Democrat", "Republican", "Independent")
partyTransition <- matrix(c(0.85, 0.1, 0.05,
                            + 0.1, 0.8, 0.1,
                            + 0.3, 0.3, 0.4),
                            byrow = TRUE, nrow = 3, dimnames = list(partyStates, partyStates))

partyTransition
```

```
##             Democrat Republican Independent
## Democrat        0.85        0.1        0.05
## Republican      0.10        0.8        0.10
## Independent     0.30        0.3        0.40
```

```
#Creating the Markov Chain
politicalParties = new("markovchain", states = partyStates, byrow = TRUE, transitionMatrix = partyTransition, name
= "Political Parties")

#Draw the Markov Chain
plot(politicalParties)
```




```
#Stationary Distribution
steadyStates(politicalParties)
```

```
##      Democrat Republican Independent
## [1,] 0.4864865  0.4054054   0.1081081
```

Looks like the Democrats are in good shape for the next election!

Take Home Message and Concluding Remarks

Markov Chains are great tools for working with probabilities that involve moving between states. We can use the package `markovchain` to work with Markov Chains in R, and make our computations quick and easy. The key functions are `plot` and `steadyStates`, and we need to create a vector of states and a transition matrix in order to make our Markov Chain. We can use our knowledge of previous states to evaluate the probabilities of being in future states.

Sources:

- https://cran.r-project.org/web/packages/markovchain/vignettes/an_introduction_to_markovchain_package.pdf
- http://blog.revolutionanalytics.com/2016/01/getting-started-with-markov-chains.html
- https://cran.r-project.org/web/packages/markovchain/vignettes/markovchainCrashIntro.pdf
- http://alexhwoods.com/markov-chains-in-r/
- http://www.mast.queensu.ca/~stat455/lecturenotes/set3.pdf
- https://www.r-bloggers.com/getting-started-with-markov-chains-part-2/
- http://en.proft.me/2014/04/14/how-simulate-markov-chain-r/
- http://statisticalrecipes.blogspot.com/2013/01/easy-introduction-to-markov-chains-in-r.html