MCMC and Covid Data

Lauren Bassett, William Berry, Adam Brewer, Javid Fathi, and Tian Qiu
Prepared for Dr. Judy Goldsmith, Computer Science, University of Kentucky,
November 21, 2020

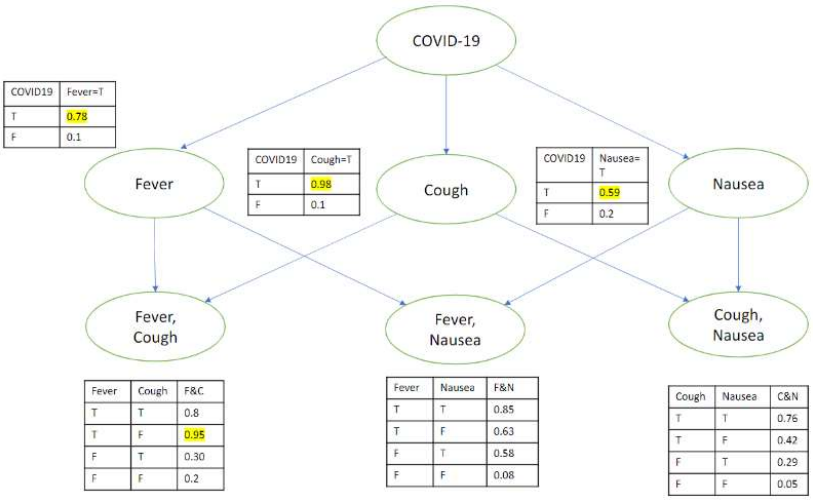
Project Abstract

In this project, we performed a Monte Carlo Markov Chain approximation of conditional probabilities using Gibbs Sampling on a simple Bayesian Network.

Additionally, we created conditional probability tables for each of the nodes in the Network, accounting for the Markov blanket of each node evaluated.

By representing states as bit-strings and using bitwise logic, we efficiently cycled through iterations of the sampling method and found values which converged on our expected probability within several thousand iterations.

Project Description:



We were provided with a simple Bayesian Network (pictured above) and asked to use MCMC with Gibbs Sampling to approximate the probability that COVID-19 (C_{19}) was True given that Nausea (N) was True at time (t = 1) and Fever & Nausea (F&N) was True at time (t = 2).

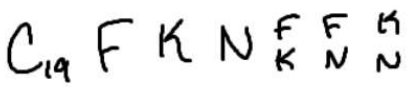
A state would be composed of a boolean configuration of the nodes, with each node set to either True or False. For the purposes of the Gibbs Sampling process, we assumed that Fever and Fever & Nausea would always be set to True.

Throughout the assignment, we will refer to each of the nodes using the following abbreviations:

COVID-19: C19, C, or C_{19}	Fever: F	Fever & Cough (t=2): F&K
	Cough: K	Fever & Nausea (t=2): F&N
	Nausea: N	Cough & Nausea (t=2): K&N

Process and Methodology:

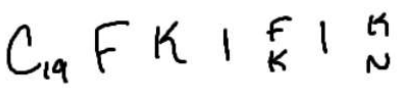
Let us imagine that every state is a binary string of the following configuration:



Where each bit is a node that can be set to True (1) or False (1).

This makes it apparent that there are 2^7 (128) unique states possible using these nodes.

We will also presume - when performing Gibbs Sampling - that the bits for Nausea and Fever & Nausea are both set to True (1):



This leaves us with 2^5 (32) unique states that can exist using this configuration - a relatively small subset to randomly traverse.

Thus, we chose to represent each state as an integer between 0 and 127, where the unsigned binary form of that integer represented the seven nodes - either set to True (1) or False (0).

Additionally, we decided to calculate our Conditional Probability Tables (CPTs) by calculating the probability of each discrete state - from 0 to 127 (0000000 to 111111).

As an example, let us say that we wanted to determine the probability that COVID-19 is True given that all other variables are set to True.

$P(C_{19} = 1 \mid F = K = N = F\&K = F\&N = K\&N = 1)$

There are only two possible states that can result from this configuration:

all variables are set to True (1111111), or

all variables but COVID-19 are set to True (0111111).

So, intuitively, the probability that COVID-19 is True conditioned on the other variables is equal to the ratio between the probability of the True configuration - P(1111111) - over the sum of the probabilities of both configurations - P(1111111) + P(0111111).

This means that:

$P(C_{19} = T \mid *) = P(A) / [P(A) + P(B)]$, where

$A = (C_{19} = 1, F = 1, K = 1, \ldots, K\&N = 1)$ and
$B = (C_{19} = 0, F = 1, K = 1, \ldots, K\&N = 1)$

Far easier than deriving the Baye's Theorem by-hand! We will use this state-based strategy regularly throughout our implementation.

Code Description: <*Code is submitted separately.*>

The code is broken into several easy-to-understand functions, with the main() driver performing most of the Gibbs Sampling process and determining what should be outputted to the user's terminal. Localized headers describe the purpose of each function for additional information.

When the program begins, the user indicates via standard input (Y/N, Enter means N by default) what options they would like to see in the program's data output, including, but not limited to:

1. Viewing the conditional probabilities of each node;
2. Approximate $P(C_{19} = 1 \mid F = F\&N = 1)$ using Gibbs Sampling and MCMC;
3. Generating and reviewing a table counting the directed transitions between any two states in the proceeding Gibbs Sampling; and
4. Reviewing the ratio that nodes where set to True in during a trial of the program.

The program will output an "live" approximation of $P(C_{19} = 1 \mid F = F\&N = 1)$ every 1000 iterations of the sampling cycle, which will attempt to "flip" the following variables during each iteration:

1. COVID-19 (C_{19}) 2. Fever (F) 3. Cough (K) 4. Fever & Cough (F&K) 5. Cough & Nausea	The initial state value (randomly) generated will guarantee that the evidence variables - N and F&N - will be set to True.

Conditional Probability Tables:

This chart shows the probabilities we calculated for COVID given each of the variables being in a state of either true or false. The first row, for example, holds Cough, Fever, and Nausea all to be true, and the probability of an individual with COVID given those symptoms is about 69.49%. The chart runs through all possible combinations of given symptoms and displays the probability conditioned on those symptoms - both True and False.

Conditional Probability Table for COVID-19 (C19):

Prob. of COVID-19	Given Var = True	Given Vars = False	Value
P(C19=T | *)	Cough, Fever, Nausea	None	0.694913373888282
P(C19=T | *)	Cough, Fever	Nausea	0.283519871467807
P(C19=T | *)	Cough, Nausea	Fever	0.066626719826692
P(C19=T | *)	Cough	Fever, Nausea	0.051705971378762
P(C19=T | *)	Fever, Nausea	Cough	0.005138443192400
P(C19=T | *)	Fever	Cough, Nausea	0.032798710876797
P(C19=T | *)	Nausea	Cough, Fever	0.007041384521273
P(C19=T | *)	None	Cough, Fever, Nausea	0.005525902888711

Conditional Probability Table for Fever (F):

Prob. of Fever	Given Var = True	Given Vars = False	Value
P(F = T | *)	C19, K, N, F&K, F&N	None	0.932686220721671
P(F = T | *)	C19, K, N, F&K	F&N	0.7715133531157271
P(F = T | *)	C19, K, N, F&N	F&K	0.597512617159336
P(F = T | *)	C19, K, N	F&K, F&N	0.578688676620606
P(F = T | *)	C19, K, F&K, F&N	N	0.986746987951807
P(F = T | *)	C19, K, F&K	N, F&N	0.877524974695242
P(F = T | *)	C19, K, F&N	N, F&K	0.823584282327334
P(F = T | *)	C19, K	N, F&K, F&N	0.702902786364290

P(F = T | *)	C19, N, F&K, F&N	K	0.961060164487236
P(F = T | *)	C19, N, F&K	K, F&N	0.791627536524767
P(F = T | *)	C19, N, F&N	K, F&K	0.613853101842485
P(F = T | *)	C19, N	K, F&K, F&N	0.587783514376534
P(F = T | *)	C19, F&K, F&N	K, N	0.985719451518785
P(F = T | *)	C19, F&K	K, N, F&N	0.880397155493659
P(F = T | *)	C19, F&N	K, N, F&K	0.825698200990384
P(F = T | *)	C19	K, N, F&K, F&N	0.705200118051954
P(F = T | *)	C, N, F&K, F&N	C19	0.302760463045414
P(F = T | *)	K, N, F&K	C19, F&N	0.278477747564996
P(F = T | *)	K, N, F&N	C19, F&K	0.184163345156707
P(F = T | *)	K, N	C19, F&K, F&N	0.153548207874534
P(F = T | *)	C, F&K, F&N	C19, N	0.588722406206941
P(F = T | *)	C, F&K	C19, N, F&N	0.287704810698520
P(F = T | *)	K, F&N	C19, N, F&K	0.361525915317116
P(F = T | *)	C	C19, N, F&K, F&N	0.137209435757911
P(F = T | *)	N, F&K, F&N	C19, K	0.421406038722972
P(F = T | *)	N, F&K	C19, K, F&N	0.337421754340810
P(F = T | *)	N, F&N	C19, K, F&K	0.145397939760735
P(F = T | *)	N	C19, K, F&K, F&N	0.106350759976262
P(F = T | *)	F&K, F&N	C19, K, N	0.655323236657391
P(F = T | *)	F&K	C19, K, N, F&N	0.338014950705213
P(F = T | *)	F&N	C19, K, N, F&K	0.301799317905679
P(F = T | *)	None	C19, K, N, F&N, F&K	0.104075438456815

Conditional Probability Table for Cough (K):

Prob. of Cough	Given Var = True	Given Vars = False	Value
P(K = T | *)	C19, F, N, F&K, K&N	None	0.9908372827804107
P(K = T | *)	C19, F, N, F&K	K&N	0.9331019092486983
P(K = T | *)	C19, F, N, K&N	F&K	0.9980569514237856
P(K = T | *)	C19, F, N	F&K, K&N	0.9636523266022827
P(K = T | *)	C19, F, F&K, K&N	N	0.9971232171516822
P(K = T | *)	C19, F, F&K	N, K&N	0.964861720178446
P(K = T | *)	C19, F, K&N	N, F&K	0.9977408751533201
P(K = T | *)	C19, F	N, F&K, K&N	0.9740405933941773
P(K = T | *)	C19, N, F&K, K&N	F	0.9948352626892253
P(K = T | *)	C19, N, F&K	F, K&N	0.9488339533401842
P(K = T | *)	C19, N, K&N	F, F&K	0.9946644787548793
P(K = T | *)	C19, N	F, F&K, K&N	0.9704260500577271
P(K = T | *)	C19, F&K, K&N	F, N	0.9967624101909742
P(K = T | *)	C19, F&K	F, N, K&N	0.9682909281469283
P(K = T | *)	C19, K&N	F, N, F&K	0.9961336706327898
P(K = T | *)	C19	F, N, F&K, K&N	0.9758533187701723
P(K = T | *)	F, N, F&K, K&N	C19	0.19692307692307692
P(K = T | *)	F, N, F&K	C19, K&N	0.1260178573283084
P(K = T | *)	F, N, K&N	C19, F&K	0.29761930122888963
P(K = T | *)	F, N	C19, F&K, K&N	0.17323669123099006
P(K = T | *)	F, F&K, K&N	C19, N	0.3721175900610576
P(K = T | *)	F, F&K	C19, N, K&N	0.11756223454930073
P(K = T | *)	F, K&N	C19, N, F&K	0.43887597494237274
P(K = T | *)	F	C19, N, F&K, K&N	0.142281805193554
P(K = T | *)	N, F&K, K&N	C19, F	0.2810767188678187

P(K = T | *)	N, F&K	C19, F, K&N	0.14283935848974613
P(K = T | *)	N, K&N	C19, F, F&K	0.24420186926026946
P(K = T | *)	N	C19, F, F&K, K&N	0.1135566574596537
P(K = T | *)	F&K, K&N	C19, F, N	0.4333008040711211
P(K = T | *)	F&K	C19, F, N, K&N	0.13624496805553493
P(K = T | *)	K&N	C19, F, N, F&K	0.37164187971354634
P(K = T | *)	None	C19, F, N, F&K, K&N	0.1063522649286373

Conditional Probability Table for Nausea (N):

Prob. of Nausea	**Given Var = True**	**Given Vars = False**	**Value**
P(N = T | *)	C19, F, K, F&N, K&N	None	0.7784308839808344
P(N = T | *)	C19, F, K, F&N	K&N	0.445490683781563
P(N = T | *)	C19, F, K, K&N	F&N	0.513537037885382
P(N = T | *)	C19, F, K	F&N, K&N	0.4092293102705076
P(N = T | *)	C19, F, F&N, K&N	K	0.9184401641932428
P(N = T | *)	C19, F, F&N	K, K&N	0.4571360889866402
P(N = T | *)	C19, F, K&N	K, F&N	0.5245376003544083
P(N = T | *)	C19, F	K, F&N, K&N	0.416179803344513
P(N = T | *)	C19, K, F&N K&N	F	0.9496947211591832
P(N = T | *)	C19, K, F&N	F, K&N	0.5768901551284408
P(N = T | *)	C19, K, K&N	F, F&N	0.6330793544074339
P(N = T | *)	C19, K	F, F&N, K&N	0.4735717864008921
P(N = T | *)	C19, F&N, K&N	F, K	0.9486924862117727
P(N = T | *)	C19, F&N	F, K, K&N	0.5820065894080375
P(N = T | *)	C19, K&N	F, K, F&N	0.6372687987531046
P(N = T | *)	C19	F, K, F&N, K&N	0.4772931433722782

P(N = T | *)	F, K, F&N, K&N	C19	0.37901900962215446
P(N = T | *)	F, K, F&N	C19, K&N	0.29790437458783775
P(N = T | *)	F, K, K&N	C19, F&N	0.3461606657647231
P(N = T | *)	F, K	C19, F&N, K&N	0.2591441484831795
P(N = T | *)	F, F&N, K&N	C19, K	0.5673200174476076
P(N = T | *)	F, F&N	C19, K, K&N	0.2587191032207444
P(N = T | *)	F, K&N	C19, K, F&N	0.4942716134189649
P(N = T | *)	F	C19, K, F&N, K&N	0.20851085831412647
P(N = T | *)	K, F&N, K&N	C19, F	0.6820063376971746
P(N = T | *)	K, F&N	C19, F, K&N	0.5345625897303659
P(N = T | *)	K, K&N	C19, F, F&N	0.3263277620248721
P(N = T | *)	K	C19, F, F&N, K&N	0.2157733549867045
P(N = T | *)	F&N, K&N	C19, F, K	0.8148941731076617
P(N = T | *)	F&N	C19, F, K, K&N	0.5298969828973237
P(N = T | *)	K&N	C19, F, K, F&N	0.494473897474254
P(N = T | *)	None	C19, F, K, F&N, K&N	0.2017416244486163

Conditional Probability for Fever&Cough (F&K)[1]*:*

Prob. of Fev&Cou	**Given Var = True**	**Given Vars = False**	**Value**
P(F&K = T | *)	Fever, Cough	None	0.800
P(F&K = T | *)	Fever	Cough	0.950
P(F&K = T | *)	Cough	Fever	0.300
P(F&K = T | *)	None	Fever, Cough	0.200

[1]: The three CPTs for (t=2) variables - Fever&Cough, Fever&Nausea, Cough&Nausea - were provided by Dr. Goldsmith in the initial Bayesian Network.

Conditional Probability Table for Fever&Nausea (F&N)[1]*:*

Prob. of Fev&Nau	Given Var = True	Given Vars = False	Value
P(F&N = T | *)	Fever, Nausea	None	0.850
P(F&N = T | *)	Fever	Nausea	0.630
P(F&N = T | *)	Nausea	Fever	0.580
P(F&N = T | *)	None	Fever, Nausea	0.080

Conditional Probability Table for Cough&Nausea (K&N)[1]*:*

Prob. of Cou&Nau	Given Var = True	Given Vars = False	Value
P(K&N = T | *)	Cough, Nausea	None	0.760
P(K&N = T | *)	Cough	Nausea	0.420
P(K&N = T | *)	Nausea	Cough	0.290
P(K&N = T | *)	None	Cough, Nausea	0.050

[1]: The three CPTs for (t=2) variables - Fever&Cough, Fever&Nausea, Cough&Nausea - were provided by Dr. Goldsmith in the initial Bayesian Network.

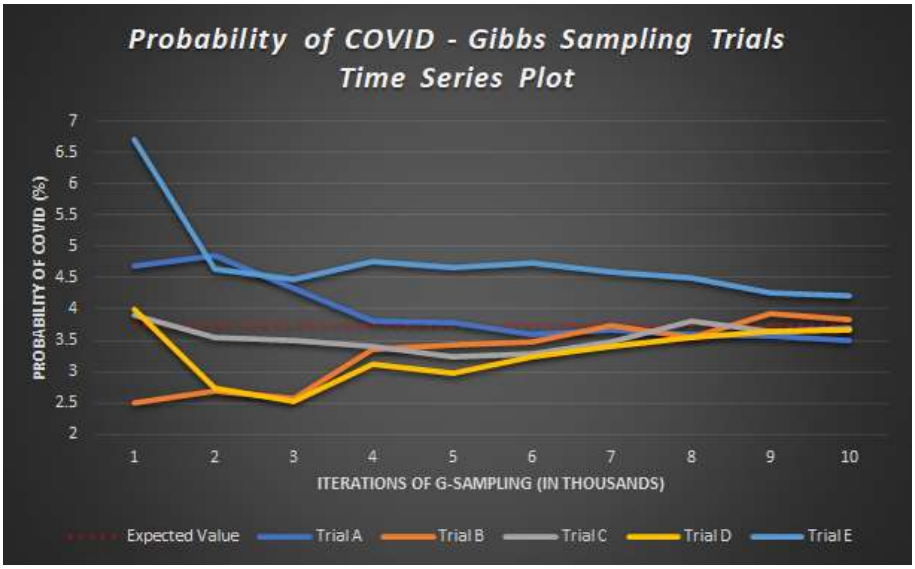
Graphs:

Time Series, Running Mean Plots:

The following graph shows the output of five trials of a Gibbs Sampling approximation of $P(C_{19} = T \mid F = F\&N = T)$, with values reported every 1,000 iterations over the course of 10,000 iterations each.

It is visually apparent that the values begin to converge around at 10,000 iterations, with a margin of error of approximately +/-1.5%.

These results are as expected, since our expected value of $P(C_{19} \mid F = F\&N = T)$ - derived from our state probabilities - is equal to approximately 3.736% of the sub-set $P(F = F\&N = T)$.

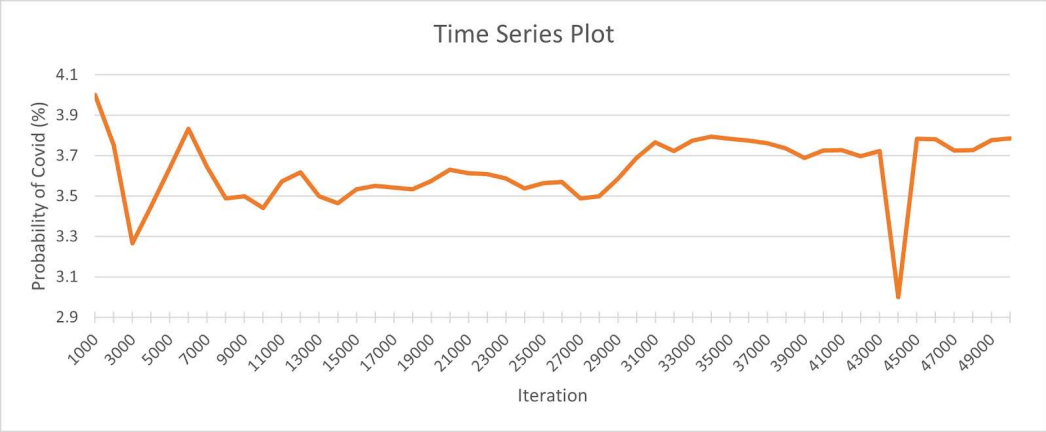


Relevant Data on the Next Page.

Relevant Data:

Iteration	Trial A	Trial B	Trial C	Trial D	Trial E
1000	4.7	2.5	3.9	4	6.7
2000	4.85	2.7	3.55	2.75	4.65
3000	4.333333	2.566667	3.5	2.533333	4.466667
4000	3.8	3.35	3.4	3.125	4.75
5000	3.78	3.44	3.24	2.98	4.66
6000	3.6	3.466667	3.283333333	3.233333	4.733333
7000	3.671429	3.742857	3.485714286	3.414286	4.6
8000	3.6	3.5375	3.8125	3.55	4.5
9000	3.566667	3.922222	3.644444444	3.644444	4.266667
10000	3.49	3.84	3.69	3.66	4.21

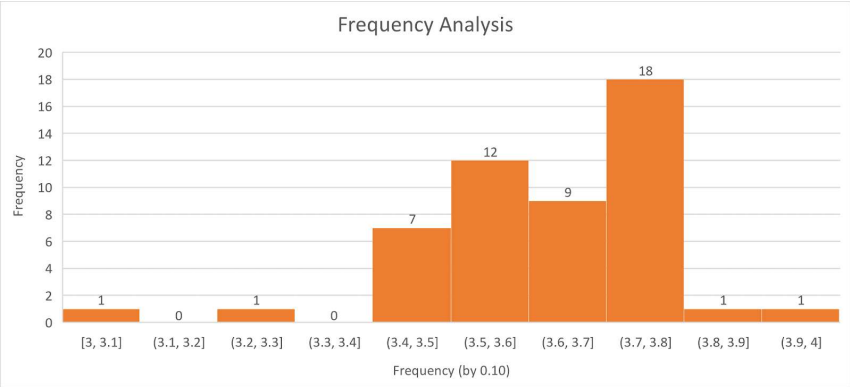
This next graph shows the approximated probability of $P(C_{19} = T \mid F = F\&N = T)$ over the course of 50,000 iterations. As you can see from the graph, the value converges much closer to our expected value than before - within +/- 0.5 percent.

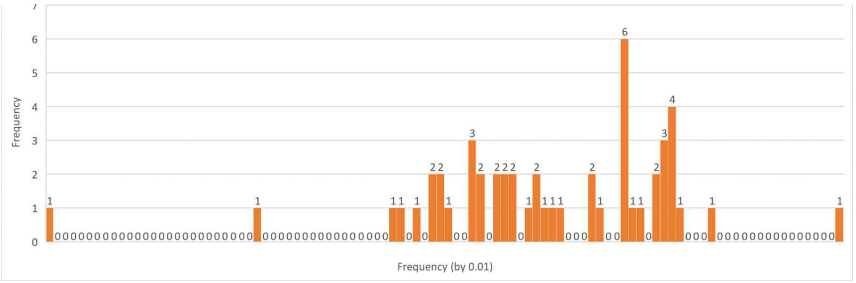


Tests of up to 500,000 iterations showed marginal improvement to the margin of error - now regularly within +/- 0.2 percent. Based on these results, we'd recommend using 50,000 or more iterations of this sampling methodology to approximate probabilities in this network. (Our state representation is so efficient that this takes only seconds more!)

Frequency Plot:

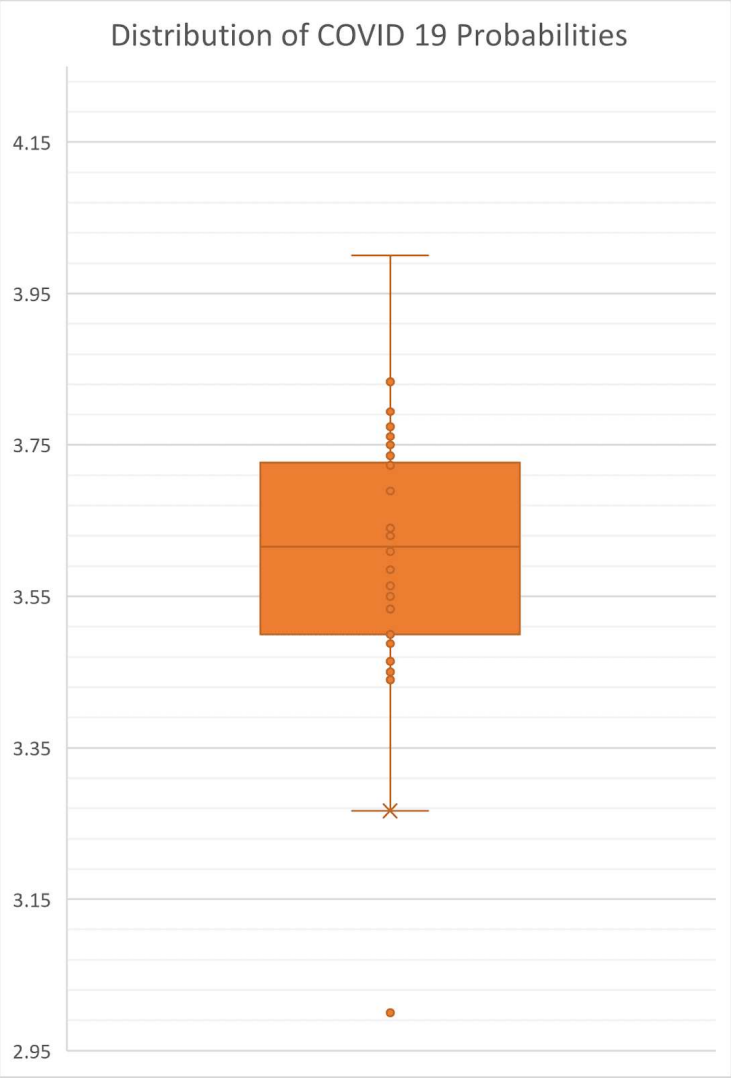
The following are a frequency analysis of the distribution of probabilities over 50,000 iterations. There are two graphs, the first separated into bins of width 0.1, and the second is separated into bins of 0.01. This gives a good picture of the exact distributions of the probabilities. The distribution is mainly centered between 3.4% and 3.8%.





Box and Whisker Plot:

This chart is another view of the distribution of the $P(C_{19} \mid F = F\&N = T)$ approximations at each iteration over our set of 50,000 iterations. The chart contains one outlier (~3%), while the rest of the data is normally distributed.



Description of Patterns Noticed:

On the Distribution of COVID Cases:

In the distribution of COVID, there was one outlier (3%). The rest of the data is normally distributed between 3.2% and 4.0%.

We also calculated the expected value for $P(C_{19} = T \mid N = T, F\&N = T)$:

$P(C_{19} = T \mid N = T, F\&N = T) = 0.03736099179066426$,
or approximately 3.736% of the sub-set.

This validated our Gibbs sampling output, providing us with a 1% MOE for all runs of 50,000 iterations or more.

The probability that COVID is true given all the child symptoms is true was higher than originally thought (69.49%), but given the simulated nature of the Bayseian network it would be expected that some derived values would be unexpected.

Learning Outcomes:

Lauren:

I learned about Monte Carlo chains and how they are implemented in estimating probabilities, and how to program probabilities in Python instead of R. My original approach to the project was to program using the MCMC package in R, but because we were given a list of probabilities instead of a dataset, I had a hard time using R programming. Additionally, I learned a lot about teamwork and communication as this project required LOTS of it. This was extremely challenging but ultimately rewarding. Stay away from Mr. Fish and his $15 massages.

William:

I learned about Bayesian networks and probabilities to a degree that was more complicated than the coursework I was required to take. (I only was required for stats at the 281 level, not 381 as part of grandfathering). I learned about utilizing errors in Python to my advantage, and assisted Javid in popping errors off of the call stack and utilizing exceptions to properly control flow in our program, which I moderately assisted in developing with the guidance of Javid Fathi. I also instructed Javid and other group members on utilizing git and github to collaborate.

This project required intense communication, and group-work. It was impossible to do without every single person listed here, and I strongly encourage you to consider that for students who worked in smaller groups. I learned to stay away from Mr. Fish and his $15 massages, and to in fact, warn other people to stay away from Mr. Fish and his $15 massages. Now you know.

Additionally, in the context of this assignment, I learned to utilize university resources at the graduate level, reaching out to and facilitating the reaching out to

multiple faculty for assistance; As admittedly, this assignment required help from multiple faculty and sources to finish, and required extensive collaboration. The challenge was extreme, but ultimately better prepared me to be an Engineer and utilize support structures properly in the workplace and in University.

Adam:

One of the biggest things I will take away from this project is how effective it can be to store data as a bitstring. Javid suggested this approach from the start and I believe it really helped me understand the project more, while also allowing me to refamiliarize myself with all of the bitwise operations. Although a lot of this was relearning, it was still super helpful for me. Another thing that I was able to brush up on during this project was my general knowledge of Python, since it has been a while since the last time I had used it. Finally, I believe that this project has also taught me a lot about communication and working as a team, which is something that I haven't had a lot of experience in in programming classes thus far. Stay away from Mr. Fish and his $15 massages.

Javid:

I learned a lot about Mr. Fish and his $15 massages. It'd be best if you stay away from them. On a more serious note, this was one of my favorite projects of the semester. I learned how to model Bayesian Networks using simple logic structures, guaranteeing efficient speed and memory output. I enjoyed getting to collaborate with my teammates - this was a difficult problem to solve, and after two weeks of work, the joy of solving it with other people was nearly euphoric. This was a great chance to test my knowledge of Bayesian statistics and C-style programming. I hope that you use this implementation in future demonstrations of the course!

Tian:

I learned about how conditional probability tables for Monte Carlo chains can be calculated from Bayesian Network. It is interesting to see that a computer can brute-force an estimated probability by running the simulation multiple times to create a probability space, then use this probability space to simulate the real world events. Also, I learned about how teamwork can help on a difficult project. Without my team, I can never get this done in time.

Stay away from Mr. Fish and his $15 massages.

People Consulted:

Dr. Judy Goldsmith, College of Engineering, University of Kentucky

Provided significant assistance in structuring assignment, calculating CPT probabilities.

Professor Kara Cook, College of Statistics, University of Kentucky -

Consulted for assistance in calculating CPT probabilities.

Dr. Brent Harrison, College of Engineering, University of Kentucky -

Provided basis evaluating CPTs, conditional probability using discrete state probabilities.

Isaac Rowe - Compared CPTs and pseudocode/methodology.

Seth Banauch - Compared CPT data.

Ngoc Phan - Discussed program methodology at length.

Mr. Fish - Provided $15 massages.

People Otherwise Assisted by Us:

- Blake Yates - Provided a copy of our CPT data upon request, as permitted by Dr. G.
- Maria Paoloni - Assisted in description of project and how to interpret probabilities
- Javid also produced resources (e.g.: notes, videos) for other students in the class, most specifically in the course Groupme. These resources can be reviewed in the class discussion and upon request.

Works Cited

Sinharay, S. (2003). *Assessing Convergence of the Markov Chain Monte Carlo Algorithms: A Review*. Education Testing Service. https://www.ets.org/Media/Research/pdf/RR-03-07-Sinharay.pdf